A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

Kongkun Charoenvisal

Dissertation proposal submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Architecture and Design Research

James R. Jones, Chair
Kihong R. Ku
Tanyel T. Bulbul
Elizabeth J. Grant
Scott A. Ragon

September 24th, 2013
Blacksburg, Virginia

Keywords: Building Information Modeling (BIM), Decision Support Systems (DSS), Vegetated Roofing Systems, Industrial Foundation Classes (IFC), Integrated Practice (IP), Sustainable Architecture

A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

Kongkun Charoenvisal

ABSTRACT

There is a body of evidence indicating that the implementation of current Architecture, Engineering, and Construction (AEC) industry business models and practices have caused negative impacts on global energy supply, ecosystems, and local or regional economies. In order to eliminate such negative impacts, AEC practitioners are seeking new business models in which the Building Information Modeling (BIM) technology can be considered an important technology driver. Despite the fact that the majority of AEC practitioners have used BIM tools for construction-level modeling purposes, some early adopters of BIM technology began to use BIM tools to better inform their design decisions. Corresponding to the increasing demand for decision support functionality, a number of studies showed that a part of BIM technology will be developed toward decision support and artificial intelligence domains.

The use of computer-based systems to support decision making processes can usually be found in the business management field. In this field, decision support and business intelligence systems are widely used for improving the quality of managerial decisions. Because of its theories and principles, Decision Support Systems (DSS) can be considered as one of the potential information technologies that can be applied to enhance the quality of design decisions. The DSS also has the potential to be constructed as a system platform for implementing building information contained in BIM models associated with other databases, analytical models, and expert knowledge used by AEC practitioners.

This study explores an opportunity to extend the capability of BIM technology toward the decision support and artificial intelligence domains by applying the theories and principles of DSS. This research comprises the development of a prototype BIM interoperable web-based DSS for vegetated roofing system selection. The prototype development can be considered a part of an ongoing research agenda focusing on the development of the integrated web-based DSS for holistic building design conducted within the College of Architecture and Urban Studies (CAUS), Virginia Tech. Through a post-use interview study, the developed prototype is used as a tool for evaluating the possibility for the DSS development and the usefulness of DSS in improving the quality of vegetated roofing system design decisions. The understanding gained from the post-use study is used to create a guideline for developing a fully functional DSS for holistic building design that will be developed in the future.

Dedications

To the memory of my grandmother, Mrs. Orla Thanyakij, and my grandfather,
Mr. Seubsawat Sakarin, whose love and kindness remain forever in my heart.

Acknowledgements

My inspired teacher, Dr. James Jones, encouraged me as a learner by sharing his own enthusiasm for learning. As a pupil under his exemplary guidance and consistent supervision, I felt that working on this dissertation was one of the most enjoyable learning experiences I have ever had. Dr. Jones shared with me the idea about shifting BIM from the "information" to the "knowledge" domain, which became the starting point of this dissertation. He also guided me through every step of the research and generously provided me with all the support I needed. I would like to take this opportunity to express my profound gratitude and deep regards to Dr. Jones for his time, patience, motivation, encouragement, generosity, and immense knowledge throughout the course of this dissertation.

This dissertation was able to be transformed from an abstract idea to a concrete piece of work because of valuable contributions from my committee members to whom I would like to express my deepest appreciation. Dr. Elizabeth Grant guided me through different steps of the research and supported me every way she could. Dr. Kihong Ku and Dr. Tanyel Bulbul tirelessly taught me about BIM from the basics to advanced levels. Dr. Scott Ragon helped me develop an in-depth understanding of software research and development, which was very important for DSS prototyping and evaluation.

Beside my advisory committee, I would like to thank Virginia Tech faculty members and staff for their contributions to this dissertation as well as my academic progress. Dr. Loren Rees dependably taught me about DSS for three consecutive semesters, which helped me develop necessary knowledge and skills for DSS design and development. Professor Steve Thompson and Professor Robert Schubert generously provided me with assistantship and research project opportunities. Craig Rick provided me with the server computer I needed and his IT support. Peggy Moles, Cindy Rubens, and Chriss Mattsson-Coon kindly assisted me in my academic progress.

I would also like to thank important contributors to the production of this dissertation. Carla Edwards devoted time out of her busy schedule at EDA English Lab to help me edit this dissertation. Steven Edwards and Nithiwat Kampanya assisted me in the deployment of the prototype DSS. Gade Kimsawatde gave her beautiful voice in five tutorial videos on the prototype DSS. As well, I would like to thank the research participants who provided valuable feedback on the prototype DSS.

I would like to thank all my colleagues and friends in the Ph.D. in Architecture and Design Research Program. Special thanks go to Bandar Alkahlan for always being available for me to exchange thoughts about the fundamental aspects of research, Daeung "Danny" Kim for helping me keep up with my research, and Jamal Alghamdi for helping me learn Revit.

Moreover, I would like to thank both the Charoenvisal and Suphanapong families for continuous support, help, and generosity throughout my doctoral journey. I would like to thank my wife, Sirirat, for her indispensable caretaking, support, and encouragement, which allowed me to focus on my studies. Lastly, I would like to thank my son, Don; watching his boundless love for learning has inspired me the most.

Table of Contents

List of Figures

List of Tables

List of Boxes

List of Equations

Chapter 1

1. Introduction

1.1. Overview

1.1.1. Building Information Modeling

In the world's developed countries and major cities, buildings can be considered to be the biggest source of energy consumption, greenhouse gas emissions, and material waste. According to the United States National Institute of Building Science (NIBS 2007), buildings consume about 40% of the world's energy and raw materials. In the United States, buildings consume 65.2% of total produced electricity. Buildings also contribute 40% of carbon emissions to the atmosphere and 20% of material waste to landfill. In 2008, the construction expenditures in the United States were approximately $1.288 trillion whereby 57% of this expenditure was wasted on non-value added efforts or waste in implementing current Architecture, Engineering, and Construction (AEC) business models. Such negative impacts on global energy supply, ecosystems, and local or regional economies have raised three significant movements in AEC industries: sustainable awareness, integrated practice, and building information modeling technology.

Building Information Modeling (BIM) has been considered a building information and database management technology that facilitates both environmental responsive and integrated practice paradigms. According to Suehiro (2008), BIM facilitates key design decisions concerning the lifecycle of a building for the sustainable paradigm. BIM can be used for strategy, planning, design, construction, operations and maintenance of sustainable buildings. Using BIM with spreadsheet programs can assist demand and capability analyses; allow for consideration of concept alternatives; create simulations of functional processes, and track performance metrics of sustainable design elements. For the Integrated Practice (IP) paradigm, BIM facilitates effective collaboration toward the common goals of IP. BIM virtually constructs the building and allows all design and construction disciplines to contribute diverse expertise to achieve project goals. BIM and IP in orchestration can improve the design process used in the current AEC business models. In Figure 1-1, BIM and IP together can shift the project team efforts from the construction documentation phase to the pre-design, schematic design, and design development phases whereby the changes made during these phases are more influential to the design and cost less than those made during the later phases. As represented by the preferred design process curve, team efforts can contribute to sustainability aspects of design projects.

The term "BIM" as coined by Eastman (1975) was recognized as the activities that were based on building information models. BIM was popularized later by Laiserin (2002) for a digital product based on this approach. According to the NIBS document, *United States National Building Information Standard*, 2007, BIM can be defined as a product, a collaborative process, and a facility lifecycle management tool. As a product, BIM is a digital representation of data about a capital facility. As a process, BIM covers

business drivers, automated process capabilities, and open information standards used for information sustainability and fidelity. As a facility management tool, BIM provides a repeatable, verifiable, transparent, and sustainable information-based environment for facility management throughout building lifecycle.

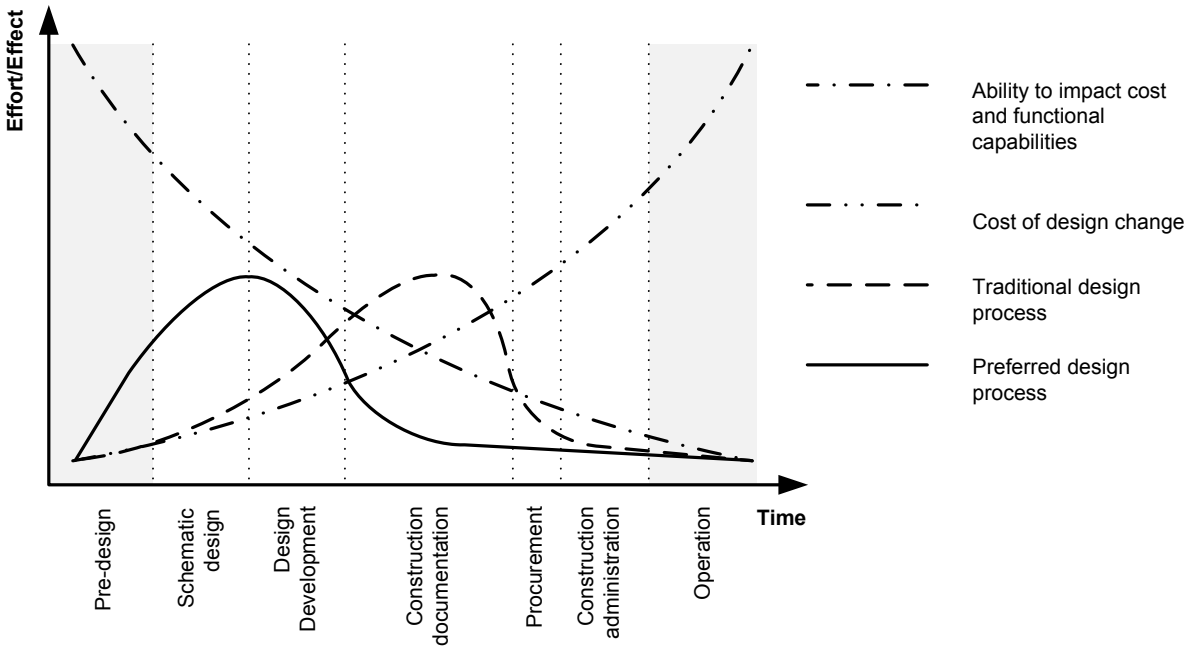


Figure 1-1: Value added, cost of changes, and current compensation distribution for design service, adapted from Pressman (2007)

A building information model such as a BIM model is a complete set of design documents in an integrated database. The BIM model is an object-based parametric model in which changes made to the model will be updated automatically. Derived from the object-based parametric model principle, building elements or objects are instances of model families or element classes (classifications). A model family is a set of relations and rules used to control the parameters in which element instances can be generated but will each vary according to their context (Eastman et al. 2008). Beyond a 2D/3D representation of the building which is normally found in CAD-based drawings, a BIM model is capable of storing information about an entire building in which all the information is parametric and thus interconnected. A BIM model is not only a center of collaboration among building project stakeholders, but also a rich resource for lifelong facility management. Founded on the BIM model principles, BIM can be considered as a relative conversion in design and documentation methodology in AEC industries (Krygiel and Nies 2008). Figure 1-2 demonstrates the potential use of BIM models.

From the architect and engineer's standpoints, BIM can be implemented in various design processes. In today's practice, BIM is mostly used in construction documentation because of BIM's base strength in

construction-level modeling. BIM can also be used for design-construction integration whereby the construction-level model can be extended to the fabrication-level model. In addition to detailed modeling applications, BIM can be used in conceptual design for generating the conceptual models and determining the underlying design concepts of building projects. Furthermore, in design and analysis of building systems, BIM can perform as a center of collaboration among design professionals such as consultant engineers, and enable the integration of analysis and simulation tools currently used by such professionals. As a result, BIM seems to be a promising technology that can be used to reinforce the quality of design decisions during design processes.

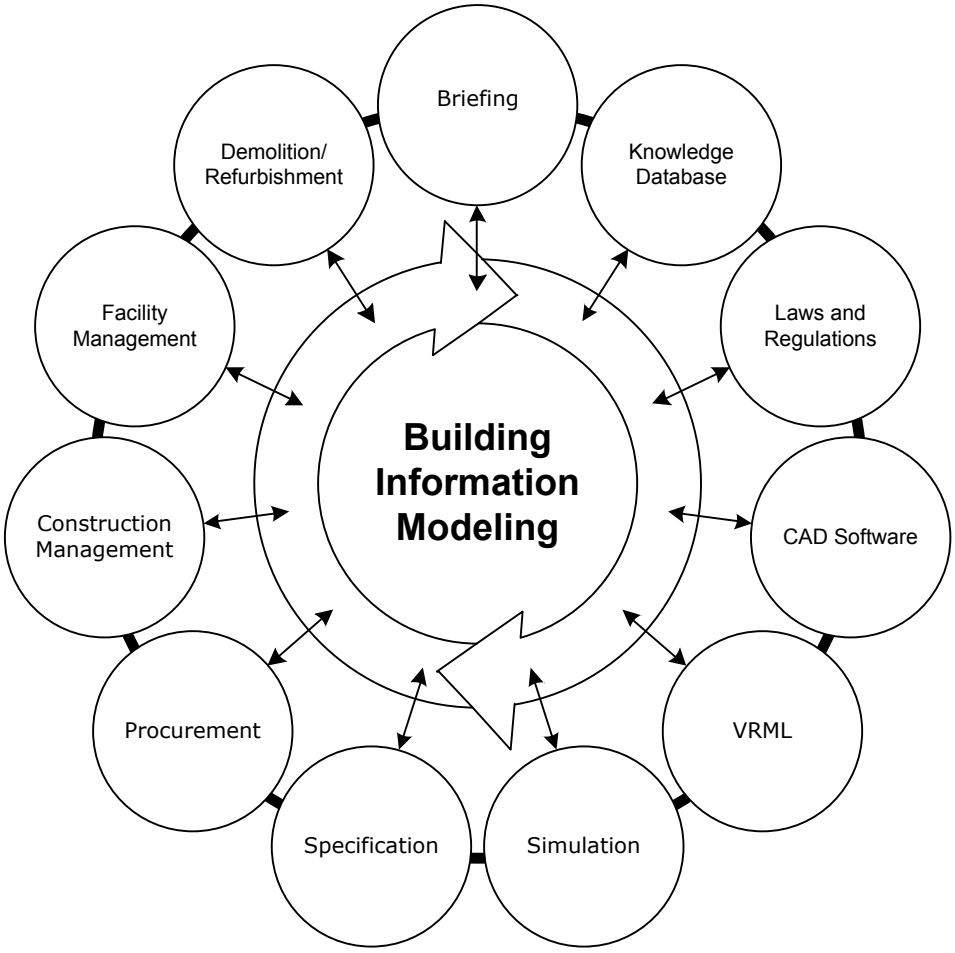


Figure 1-2: BIM utilization throughout building lifecycle based on the International Alliance of Interoperability (IAI) Nordic chapter 2000 BIM product Model, adapted from NIBS (2007)

Haymaker and Suter (2006) categorized reasoning in the AEC industry into two categories: manual and automated, based on the nature of dependency. Because of the unique nature of AEC projects, it is quite often that decision making cannot be algorithmically formulated. For this aspect, the decisions have to be manually made from the available information and a set of criteria used to evaluate such information. On the contrary, in many cases, computer algorithms already exist, which help AEC practitioner construct

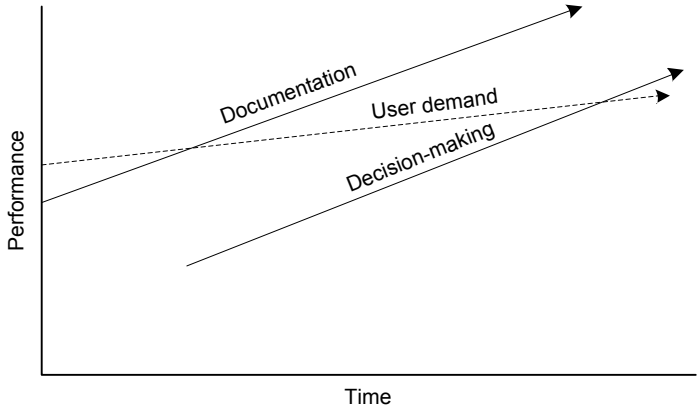
useful dependent information. Such algorithms allow AEC practitioners to easily define project-specific automated reasoning where possible. In today's practice, Haymaker and Suter pointed out that, AEC practitioners are using computer programs that can automatically construct useful task-specific dependent information from source information. The task-specific reasoning can be programmed using computer programming and query languages. However, the task-specific reasoning tools and generic query languages are not currently leveraged in AEC practices.

Referring to Haymaker and Suter's perspective, BIM seems to have the potential to be used for enhancing both manual and automated reasoning. However, BIM is not extensively used to support such activities. In 2007, a survey conducted by Joann Gonchar indicated that 74% of U.S. architectural firms used 3D modeling and BIM tools while 34% of firms use BIM for intelligent modeling. The findings showed that up to 25% of U.S. firms have used BIM beyond 3D modeling and/or documentation tools. However, based on a number of case studies on BIM implemented building projects, Eastman, et al. (2008) forecasted that, in 2012, the early adopters of BIM will move toward performance-based design by utilizing BIM tools to better inform their design decisions. Continually in 2020, BIM systems will be recognized as a substituted expression of artificial intelligence. In the near future, BIM tools will be convenient platforms for a regeneration of expert system developments for a wide range of purposes including, but not limited to, design guides and design wizards.

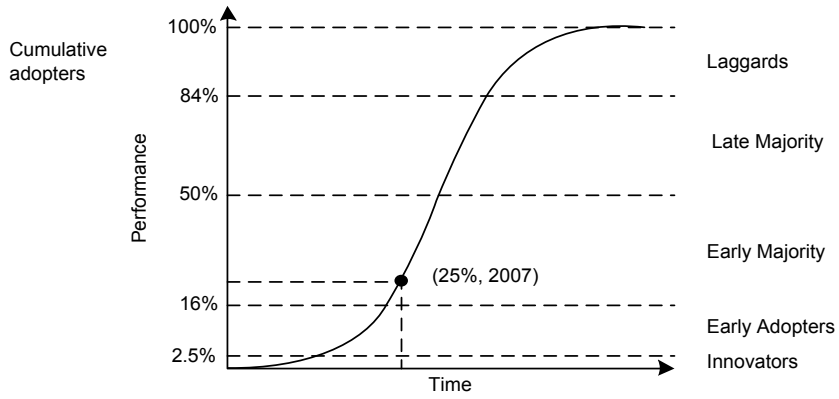
When contemplating the innovation in BIM technology using analysis tools such as the trajectories of technology improvement and trajectories of market needs, Figure 1-3a, introduced by Christensen (1997), and technology s-curve of cumulative adopters, Figure 1-3b, introduced by Rogers (2003), it is very convincing that a part of BIM technology is gradually extended toward the decision support and artificial intelligence domains. Based on Gonchar's survey in 2007, Figure 1-3a indicates that performance of modeling functions is likely to fulfill the demand of the AEC market. That is 75% of U.S. architectural firms which have used BIM tools for 3D and/or construction-level modeling. Consequently, there is a high probability for a new function of BIM to be pioneered in the market. Gonchar also found that 25% of the firms have used BIM for intelligent modeling. Corresponding to Gonchar's survey, Eastman, et al. also found that the early adopters of BIM technology tend to use BIM tools to inform their design decisions. Therefore, Gonchar and Eastman, et al.'s findings reinforce that the innovation in BIM technology is more than likely to be continued for the decision support and artificial intelligence functions.

Figure 1-3b reinforces the trend indicated by Figure 1-3a that a part of BIM technology will be extended toward the decision support and artificial intelligence domains. Based on Gonchar's study, 25% of U.S. architectural firms have used BIM for intelligent modeling where Gonchar stated that the use of BIM to perform such a function was rare during the five years before 2007. This information can be mapped onto the technology s-curve shown in Figure 1-3b. Figure 1-3b indicates that the innovation in BIM technology toward the decision support and artificial intelligence domains begin to be on the steepest gradient of the curve. Supported by Eastman, et al.'s 2012 and 2020 visions, the growth spurt period will be continued

until or beyond 2020. Furthermore, based on Figure 1-3b, the technology s-curve projection suggests that BIM, in the near future, will be adopted by the majority of AEC practitioners as a decision support or intelligent modeling tool.



(a) Trajectories of technology improvement and trajectories of market needs



(b) Technology S-curve of cumulative adopters

Figure 1-3: Trajectories of technology improvement and trajectories of market needs and technology S-curve of cumulative adopters

From underlying technology, current trends, and future potentials, it can be concluded that a part of BIM technology is progressively extended toward the decision support and artificial intelligence domains. The underlying object-based parametric model technology allows BIM models to capture the information about buildings as a whole. The information captured in BIM models has much potential to be reconstructed for enhancing the quality of design decisions. In addition, the current trends reflect that the use of BIM to better inform design decisions has gained more interest than ever before. In the near future, the decision support and artificial intelligence functions of BIM technology are more than likely to be adopted by the majority of ACE practitioners. However, the ongoing research conducted by both academic and nonacademic organizations within this area of interest is still limited in number. As a result, the use of BIM technology in decision support and artificial intelligence domains should be investigated.

1.1.2. Computer-Based Decision Support System

While a part of BIM technology tends to be developed toward the decision support and artificial intelligence domains, the use of decision support and artificial intelligence systems can be found in other fields, especially in the business management field. In order to perform managerial functions, business managers are usually involved in a continuous process of making decisions. In accordance with today's business environment where managers have to work under pressures and respond quickly to changing business factors, the decision making process has become more complex.

According to Simon (1977) and Turban et al. (2007), the decision making processes fall within a range from highly structured to highly unstructured decisions. Structured processes are routine and repetitive problems for which standard methods are available. Unstructured processes are fuzzy and complex problems for which there are no definite solution methods. The decision making process can be described with a four-phase process of intelligence, design, choice, and implementation. The intelligence phase includes searching for conditions that require decisions. The design phase includes inventing, developing, and analyzing possible alternative solutions or course of actions. The choice phase includes selecting a solution from the available alternative solutions. The implementation phase includes adapting the selected solution to the decision situation. Based on the four-phase decision making process, in structured problems, all four phases are structured. In unstructured problems, none of the four phases are structured. In between the two extremes, there are semi-structured problems that have some structured elements and some unstructured elements.

Based on the concepts articulated by Gorry and Scott-Morton (1971), the Decision Support System (DSS)[1] can be defined as the interactive computer-based system which helps decision makers utilize data and models to solve unstructured problems. In addition to this definition, the following definition of DSS was provided by Keen and Scott-Morton (1978):

> Decision support systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer-based support system for management decision makers who deal with semi-structured problems.

Referring to Sprague and Carlson (1982), the decision support system was considered a part of the decision making system. The decision making system as shown in Figure 1-4 consists of a manager or user who uses a decision support system to confront a task in an organizational environment.

Sprague and Carlson's basic architecture of DSS consists of three components: data subsystem, models subsystem, and dialog subsystem. DSS is operated on a complex software system that has three sets of capabilities: Database Management Software (DBMS), Model-Based Management Software (MBMS), and Dialog Generation and Management Software (DGMS). The data subsystem functions comprise data

[1] DSS is used as both singular and plural (system and systems), as are many other acronyms in this document.

creation–generation and restructure, update, and inquiry and retrieval. The data subsystem is managed through the DBMS. The models subsystem functions include model creation–generation, maintenance–update, and manipulation–use. The models subsystem is managed using the MBMS. The dialog subsystem is responsible for interacting with a DSS user through action and presentation languages. The dialog subsystem is managed through the DGMS. Figure 1-5 demonstrates the basic architecture of DSS.

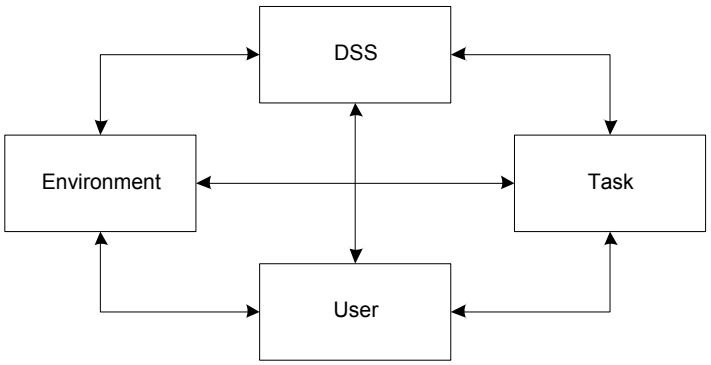


Figure 1-4: Decision making system, adapted from Sprague and Carlson (1982)

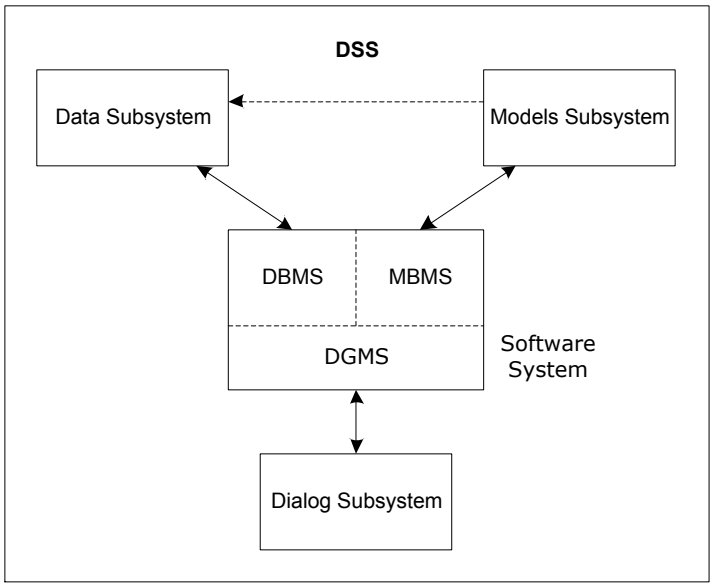


Figure 1-5: Components of the decision support system, adapted from Sprague and Carlson (1982)

In addition to Sprague and Carlson's basic DSS architecture, Turban et al. considered the term "DSS" as an umbrella term that could be used to described any computerized system that supports decision making in an organization. Turban et al. described that many unstructured and even semi-structured problems are so complex that their solutions require expertise. The required expertise can be provided by an expert system or another intelligence system. Hence, advanced DSS may be equipped with an

additional component known as a knowledge-based management subsystem. Figure 1-6 demonstrates Turban et al.'s high-level DSS architecture. In this architecture, the knowledge-based management subsystem can support any of the other subsystems or perform as an independent component. The knowledge-based management subsystem is an optional component that enhances the three major components: data management, model management, and dialog management subsystems, described by Sprague and Carlson.

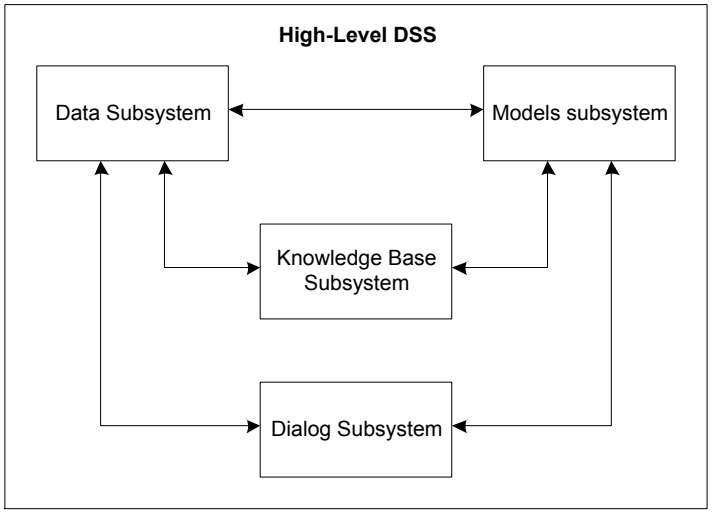


Figure 1-6: High-level architecture of DSS, adapted from Turban et al. (2007)

According to Turban et al., rapid growth of the Internet and Web technologies has brought considerable changes in how decision makers are supported. Technologies allow decision makers to access a vast body of data, information, and knowledge available around the world. Technologies provide a common, user-friendly Graphical User Interface (GUI) which is uncomplicated to learn, easy to use, and readily available. Also, technologies allow decision makers to effectively collaborate with remote partners. Through the Internet and Web, decision makers can use available intelligent search tools to quickly and inexpensively find needed decision support information. Using the Internet and Web technologies, decision makers are capable of accessing information at any time and from anywhere. Such changes have made the Internet and Web technologies to be one of the attractive platforms for DSS.

Figure 1-7 represents a possible web-based architecture of DSS demonstrated by Turban et al (2007). In this architecture, processing is distributed across several servers working in concert to solve complex analytical problems. Referring to Sprague and Carlson's basic DSS architecture, the data server can be considered as the DBMS component which optionally extracts data from the data warehouse or legacy DBMS. The optimization, simulation, or other sever can be considered as the MBMS component. The Web browser can be considered as the DGMS component that works in conjunction with the application sever through the Web server.

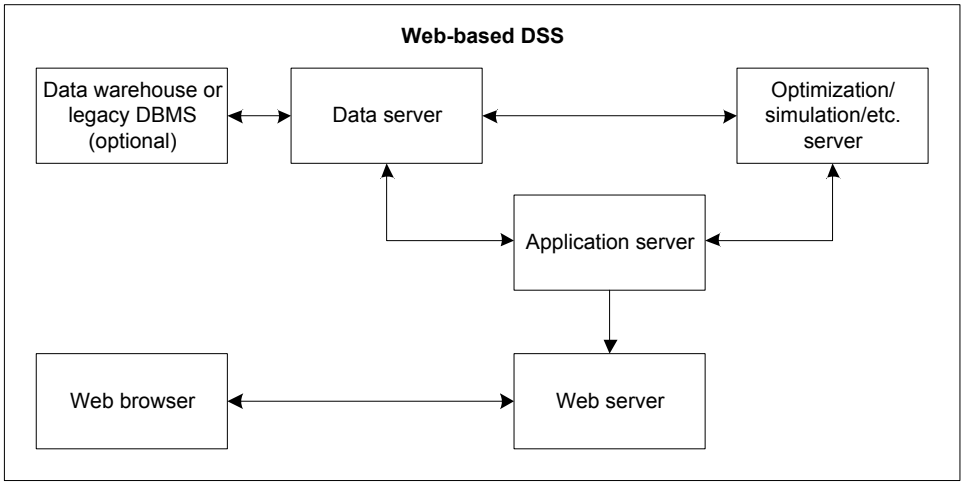


Figure 1-7: Multitier architecture for incorporating optimization, simulation, and other models into web-based DSS, adapted from Turban et al. (2007)

Decision support systems built upon previously described architectures can be categorized. According to the Association for Information Systems Special Interest Group on Decision Support Systems (AIS SIGDSS), the DSS are categorized into six categories as shown in Table 1.1. The AIS SIGDSS defines the first five major categories by the individual DSS component that drives each entire system, whereas the sixth category, combine or hybrid DSS, is a combination of two or more DSS major categories. Furthermore, the key characteristics and capabilities of the DSS are presented in Figure 1-8.

While, the term “DSS” is likely to be used in academia and research, the term “Business Intelligence (BI)” is likely to be used as a commercial term. However, Turban et al. suggested that both systems share some similarities, but they are different systems. Turban et al. defined the BI as a conceptual framework for decision support that combines architecture, databases (data warehouse), analytical tools and applications. Turban et al. suggested that DSS and BI are different in six aspects. First, DSS are appropriate to any type of organization when compared to BI which are more appropriate for large organizations due to system’s expensive data warehouses. Second, DSS are constructed to directly support specific decision making while BI are constructed to provide accurate and timely information and indirectly support decision making. Third, DSS are oriented toward analysts, whereas BI are oriented toward executives and strategic planning. Fourth, DSS are constructed to solve very unstructured problems where additional programming efforts may be required to customize the solutions, while BI are constructed with available commercial tools and components that are fitted to the needs of organizations. Fifth, DSS methodologies and tools are developed in the academic institutes whereas BI methodologies and tools are developed by software companies. Sixth, many DSS tools can be considered as the tools used by BI.

Table 1-1: Summary of the AIS SIGDSS classification for DSS

DSS Category	Description
Communications-driven and group DSS (GSS)	DSS in this category utilize computer, collaboration, and communication technologies to support groups in tasks that may or may not include decision making. Fundamentally, all DSS that support any group work fall into this category.
Data-driven DSS	DSS in this category process data into decision support information, and present the processed information to the decision makers. Many DSS developed in the Online Analytical Processing (OLAP) and data mining software systems basically fall into this category.
Document-driven DSS	DSS in this category use knowledge coding, analysis, search, and retrieval capabilities to support decision making. Essentially, all text-based DSS fall into this category.
Knowledge-driven DSS, data mining, and management Expert System (ES) applications	DSS in this category utilize the application of knowledge technologies for addressing specific decision support needs. In essence, all artificial intelligence-based DSS fall into this category.
Model-driven DSS	DSS in this category are developed around one or more optimization or simulation models that include significant activities in model formulation, model maintenance, model management in distributed computing environments, and what-if analysis. Many large-scale applications fall into this category.
Compound DSS	DSS in this category, also known as hybrid DSS, include at least two major categories from the five categories. For example, the ES can benefit by utilizing the model-driven and/or data-driven DSS to solve complex problems such as the DSS used by healthcare providers.

From its underlying theories and principles, the decision support system can be considered one of the potential information technologies that can be used for improving the quality of design decisions. The DSS also has potential to be used as a system platform that utilizes building information contained in BIM models together with other databases, analytical models, and practicing knowledge. As similar to business managers, design practitioners in the AEC industry are also involved in an iterative process of making semi-structured decisions. Based on today's industrial situations where design practitioners are seeking new business models that emphasize sustainable and integrated practices. Such decision making processes have grown to be more complex than the processes used in the past. The situations indicate that computer-based systems which can enhance the quality of design decisions have become increasingly important. Furthermore, there is a tendency to believe that a part of BIM technology will be extended toward decision support and artificial intelligence domains. Such tendency demonstrates that a BIM interoperable DSS has potential to be adopted by AEC practitioners. As a result, the development of DSS for AEC practitioners and the usefulness of such DSS in enhancing design decisions should be examined.

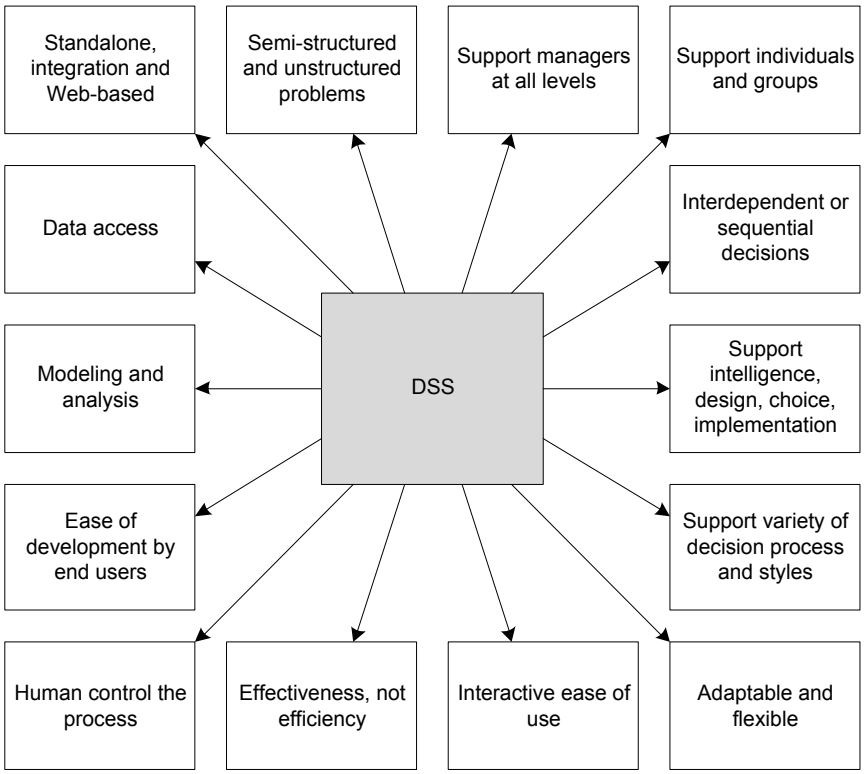


Figure 1-8: Key Characteristics and Capabilities of DSS, adapted from Turban et al. (2007)

1.2. Problem Statements

From current situations of the AEC industry, AEC practitioners are in search for new business models that help eliminate the negative impacts on global energy supply, ecosystems, and local or regional economies caused by contemporary business models. The Building Information Modeling (BIM) technology is thought to be a technology driver behind new business models. While the majority of AEC practitioners use BIM tools for construction-level modeling purposes, some early adopters of BIM technology tend to use the tools for decision support purposes. Corresponding to the increasing demand for decision support functionality, a number of studies showed that a part of BIM technology is likely to be developed toward decision support and artificial intelligence domains.

The use of computer-based systems to support decision making processes is existent in other fields, especially in that of business management. In the business field, the decision support and business intelligence systems are widely used to improve the quality of managerial decisions. From its theories and principles, DSS can be considered as one of the information technologies that are potentially capable of improving design decisions. The DSS also has the potential to be used as a system platform for implementing building information contained in BIM models associated with other databases, analytical models, and expert knowledge used by AEC practitioners.

The underlying background indicated that the development of DSS for AEC practitioners and the usefulness of DSS in improving design decisions should be investigated, and led to the two following research problem statements:

Problem 1 - Can a BIM interoperable DSS be developed to improve the quality of design decisions?

Problem 2 - Can a BIM interoperable DSS improve the quality of design decisions?

1.3. Objectives

This dissertation comprises two major objectives. The first objective is to develop a prototype of a BIM interoperable web-based DSS for AEC practitioners. The scope of DSS can be defined as the interrelationship between sustainability, integrated practice, and building information modeling technology paradigms as shown in Figure 1-9. From the sustainability aspect, the prototype is developed to enhance the quality of sustainable building design decisions. From the integrated practice aspect, the prototype is constructed upon the web-based DSS architecture in order to be platform independent as well as constructed to be used as a communication medium to support the communication between design team members. The prototype is also built to assert the legality of documenting design decisions which could be beneficial in the litigation between project members. From the building information modeling technology aspect, the prototype is structured to be capable of utilizing building information contained in BIM models together with other databases, analytical models, and practicing knowledge used by AEC practitioners.

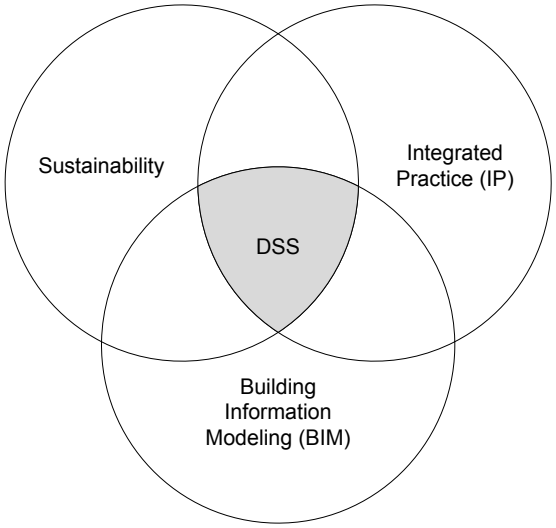


Figure 1-9: DSS as the interrelationship between sustainability, Integrated Practice, and Building Information Modeling technology

The proposed architecture of the prototype DSS is demonstrated in Figure 1-10. The architecture is a combination of the high-level and web-based DSS architectures. The components of this DSS can be

divided into two parts. The server-side components consist of the DBMS, MBMS, and KBMS located on an application server (or servers). The client-side component consists of the DGMS (Web browser) located on a local computer. Based on the AIS SIGDSS categories, the prototype DSS can be defined as a hybrid DSS in which the KBMS component is included to perform some tasks as an independent unit as well as to enhance other DSS components.

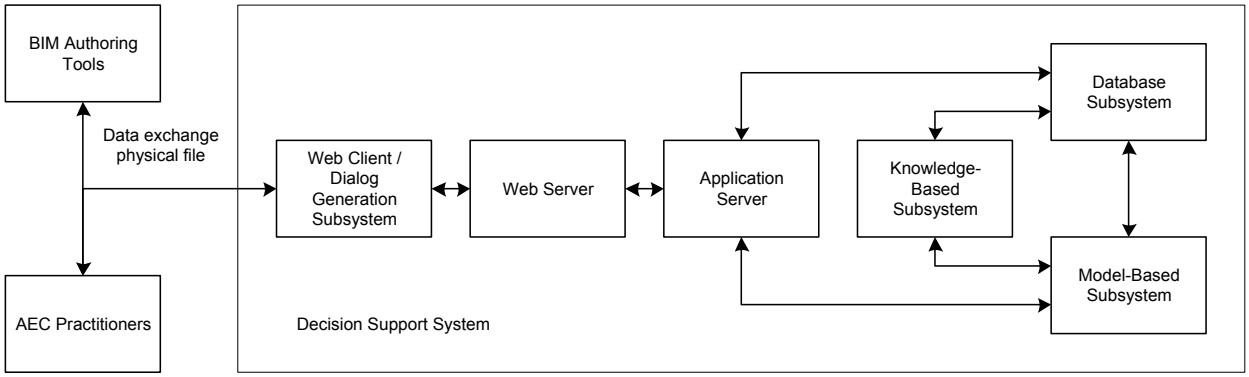


Figure 1-10: Proposed DSS Architecture

The second objective of this study is to determine the possibility that DSS can be developed and is useful for improving the quality of design decisions. The development possibility and usefulness of DSS can be determined by post-use information collected from a group of prospective users through a post-use interview study. The information includes the performance of DSS in fulfilling user requirements and in improving the quality of design decisions. In addition, the information about benefits and detriments of using DSS during design phases are also captured. The understanding gained from the post-use study is intended to be used as a guideline for developing the fully functional DSS for sustainable building design that will be developed in the future. The fully functional DSS is expected to be a platform in which multiple DSS applications similar to the DSS for vegetated roofing system selection can be combined to support various decisions involved in sustainable building design.

1.4. Methodology

Referring to Groat and Wang (2002), this study employs two research strategies: logical argumentation and qualitative research. The two strategies are combined using the two-phase design method which combines two strategies in a sequence of distinct phases. In the first phase, the formal mathematic/computer logical argumentation research strategy is used in the process of developing a prototype DSS. In the second phase, the qualitative research strategy is used to determine the performance of prototype DSS in fulfilling expected requirements and the usefulness of prototype DSS in improving the quality of design decisions.

The development of a prototype DSS utilizes the evolutionary process model suggested by Pressman (2010). The prototyping process model is an evolutionary process model that is suitable for this research

in several aspects. The prototyping process is usually used to help developers and other project stakeholders better understand the system that is to be built when requirements are fuzzy. The fuzziness of requirements may include the uncertainty of the efficiency of an algorithm, the adaptability of an operating system, or the form of interaction between user and system. Essentially, the prototyping process serves as a mechanism for identifying software requirements. The developed prototype can be considered a throwaway version which may or may not evolve into the actual system. Figure 1-11 represents the prototyping process model consisting of five phases: 1) communication, 2) quick plan, 3) modeling and quick design, 4) construction of prototype, and 5) deployment, delivery, and feedback. Since research conducted related to the DSS in the AEC industry are small in number, the characteristics and requirements of DSS are still uncertain. Furthermore, designers may not be familiar with DSS since they have not been widely used in the design process. Therefore, the prototyping process model can be considered the most appropriate approach for this study.

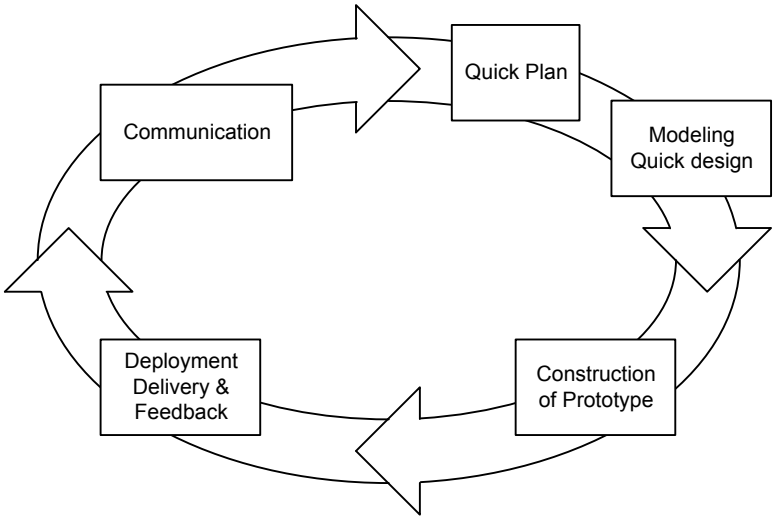


Figure 1-11: Prototyping paradigm, adapted from Pressman (2010)

While most of DSS projects are typically driven by "demand pull" or "visible, concrete needs" rather than "technology push" (Keen 1980), the prototyping process can be seen as a mechanism for assessing the needs of and obtaining feedback from working AEC practitioners. In general, the prototyping process highly encourages user participations, especially during communication and deployment phases of the process. As shown in Figure 1-11, the prototyping process allows users' needs be assessed during the communication phase and allows users' feedback to be collected during the deployment phase. For this research, direct communication methods such as project participant meetings can be used to help the researcher understand the needs of practitioners as well as obtaining their feedback.

After the prototype DSS is well developed, the post-use study founded upon the qualitative research strategy can be conducted. The study uses the personal and/or telephone interviewing tactics to understand the consensus of user opinion regarding the performance of the prototype DSS in fulfilling

user needs and in improving the quality of design decisions. The post-use interview study goal is to determine the possibility for DSS to be developed and the usefulness of DSS in improving design decisions, using the interpretive approach. The possibility of development can be determined by post-use information about the performance of prototype DSS in fulfilling expected requirements. The usefulness of DSS can be determined by post-use information about the performance of DSS in enhancing design decisions. In other words, the post-use interview study can be seen as a mechanism for gaining in-depth understanding of users' needs and detailed users' feedback. In addition, the benefits and detriments of using DSS during design phases are captured. The information collected from the post-use study is also used for developing a guideline which is a set of essential requirements that can be used in the development of the actual fully functional DSS.

1.5. Assumptions and Limitations

In order to promote the sustainable design paradigm and confine the boundary of research, this study intends to develop a prototype of a BIM interoperable web-based DSS for enhancing the quality of vegetated roofing (green roof) system design decisions. The use of green roof systems is one of the environmental responsive design strategies that have been widely adopted for decades in Europe and North America. However, systematic methods for selecting an appropriate green roof system for a particular building project have not been extensively used. Furthermore, there is no complete DSS that is capable of improving the quality of green roof design decisions.

Within the scope of this study, the prototype DSS is developed around a decision-making framework created by Elizabeth J. Grant (2007) in her dissertation, *A Decision-Making Framework for Vegetated Roofing System Selection*. This is because this framework represents a complete conceptual decision support framework that uses data, models, and expert knowledge to enhance design practitioner decisions. The framework has also been considered one of the most efficient tools for green roof system design, through the evaluations by comparing with other methods employed by green roof system design experts. By implementing this framework, this study can begin to search for appropriate technologies, designs, and attributes required for DSS prototyping, and then continue to develop the prototype DSS. The primary function of the prototype DSS is to help design practitioners select appropriate green roof systems for their projects. The design practitioners may comprise the potential users of the vegetated roofing selection framework including licensed architects, landscape architects, and roof consultants or engineers.

In addition to the limitation imposed by the implementation of the vegetated roofing system selection framework, the use of prototyping as a software engineering process tends to have inherent limitations, especially when it comes to software quality and maintenance. According to Pressman (2010), when developing a prototype, software engineers often make implementation compromises in order to get it working in a timely manner. Such compromises, which may include using inappropriate programming languages and implementing inefficient computational algorithms, often cause a significant impact on the

overall quality and/or long-term maintainability of the prototype. As a result, the prototype is frequently developed to serve as a mechanism for defining software requirements and may not be evolved into the full version of the system when the software development is geared toward quality and maintainability.

Because the prototype software apparently lacks overall quality and long-term maintainability, these issues need to be taken into consideration when developing the full version of DSS. Fundamentally, as part of the full system development, the domain engineering technique, which is increasingly used by software engineers to systematically share and reuse software components, can be implemented to help improve the quality and maintainability of the full system. The significant benefit of implementing this technique is that it helps reduce redundancy in software engineering work, either in building new systems or upgrading existing software systems. For the reason that this study concentrates on the development of a prototype DSS, the implementation of the software quality techniques such as the domain engineering technique is beyond the scope of this study.

Although the implementation of prototyping process may create some limitations concerning the quality and maintainability of the prototype DSS, it does not seem to have a significant impact on the system from an evaluation standpoint. This is because the evaluation of a DSS typically emphasizes the "effectiveness" of the decision produced rather than the computational "efficiency" of obtaining the decision (Turban et al. 2007). Based on this discernment, it can be considered that the prototype DSS is still valid if it can make end users believe that the fully released version of the DSS will do the job that they expected it to do. Therefore, it can be assumed that the prototype DSS developed as part of this study can be used as a tool to understand the usefulness of the fully released version of the DSS that will be developed in the future.

1.6. Hypotheses

In accordance with the underlying backgrounds and problem statements, this study examines the possibilities that a BIM interoperable web-based DSS can be developed and can be useful for improving the quality of green roof design decisions. Within this study, a prototype DSS is developed and used as a part of subsequent post-use interview study. The post-use study is conducted to understand the consensus of research participant opinion concerning the performance of the prototype DSS in fulfilling expected requirements and the usefulness of the prototype DSS in improving the quality of design decisions. The consensus of user opinion is summarized in a form of qualitative summary of participant responses as a "rich" story. The findings of this research can be hypothesized as follows:

Hypothesis 1 – The BIM interoperable web-based DSS can be developed to improve the quality of green roof design decisions.

Hypothesis 2 – The BIM interoperable web-based DSS can improve the quality of green roof design decisions.

1.7. Contributions

1.7.1. Theoretical Grounding

It has been recognized that the current business models and practices of Architecture, Engineering, and Construction (AEC) industries have caused negative impacts on global energy supply, ecosystems, and local or regional economies. In consequence, AEC practitioners are seeking new business models to overcome the shortcomings of current models whereby the Building Information Modeling (BIM) technology can be considered an important technology driver. Many scholars such as Eastman et al. (2008) and Krygiel and Nies (2008) consider BIM as a technology supporting a paradigm shift in AEC industries. Eastman et al. considered BIM a paradigm change that will have benefits for AEC industries as well as for society in general, because he believes BIM helps improve the quality of buildings. Integrating BIM with the new AEC business models can reduce building energy consumption as well as labor and capital resources. In addition to Eastman et al.'s opinion, Krygiel and Nies considered BIM a switch in design and documentation methodology in AEC industries. BIM has shifted how AEC practitioners view the entire process from preliminary design, through construction documentation, into actual construction, and further into post construction management.

In order to understand the current movements in AEC practices, it is interesting to discuss the definitions of the term "paradigm" and process of paradigm change at-large. According to epistemologist and historian of science, Thomas S. Kuhn (1970), the paradigm has two different definitions as follows:

> On the one hand, it stands for the entire constellation of beliefs, values, techniques, and so on shared by the members of a given community. On the other, it denotes one sort of element in that constellation, the concrete puzzle-solutions which, employed as models or examples, can be explicit rules as a basis for the solution of the remaining puzzles[2] of normal science[3].

In his book, *The Structure of Scientific Revolutions* published in 1970, Kuhn suggested that, in science, there are three distinct phases involved in scientific progress. The first phase is defined as the pre-paradigm. During this phase, there is no central or dominant paradigm, but there are potential paradigms proposed by a number of science schools who compete for the domination of a given field. After one of the candidate paradigms is accepted by the members of a given field, the second phase defined as normal science begins. In general, the selected paradigm can be limited in scope and precision at the time of its first appearance. Thus, during this phase, scientists mostly spend their efforts on solving the remaining puzzles within the context of the dominant paradigm. The normal science phase will continue until it reaches a crisis—a situation in which anomalies are discovered and accumulate to the point that they cannot be resolved within the context of the normal science. When the normal science phase

[2] Puzzles are special category of problems that can serve to test ingenuity or skill in solution (Kuhn 1970).

[3] The normal science is research firmly based upon one or more past scientific achievements, achievements that some particular scientific community acknowledges for a time as supplying the foundation for its further practice (Kuhn 1970).

reaches a crisis, the three-phase scientific progress enters its third phase defined as revolutionary science. During this phase, new possible paradigms are formulated whereby a candidate paradigm can be accepted as a new dominant paradigm. Within this phase, a paradigm change, later widely known as a paradigm shift, occurs when a new dominant paradigm is accepted by the members of a given field to replace an existing paradigm.

Inference to Kuhn's structure of scientific revolutions, the current business models and practices such as Design-Bid-Build (DBB) can be considered paradigms that have been employed by AEC practitioners for decades or even a century. Although there were a series of anomalies emerging during the time when the existing paradigms were employed, such anomalies either were resolved within the context of paradigms or disappeared on their own. For example, there was a major energy crisis in the U.S. in 1970s. In AEC industries, the energy crisis raised the awareness of energy use in buildings. However, the energy crisis did not stay long enough to cause a paradigmatic crisis to the AEC community, because the energy crisis was resolved externally by the increasing energy prices and economic recessions that decreased the overall national energy consumption (Randolph and Masters 2008). Since the 1970s, some AEC practitioners have continued to practice energy efficient buildings, even though this approach did not become a dominant paradigm.

According to Randolph and Masters, the reduction of U.S. energy consumption appeared for only a short period of time. Since the 1980s, the energy consumption has continually increased. In 2005, there was a body of evidence showing that the building industry alone consumed about half of U.S. produced energy and emitted about half of U.S. greenhouse gas emissions. In addition to energy and environmental dilemmas caused by the building industry, Eastman et al. suggested that the traditional AEC business models and practices tend to be a major cause of industrial losses and contribute unnecessary waste and errors due to the low-quality information flow and redundancy. Based on more accurate measures, it can be concluded that these discovered anomalies can lead to paradigmatic crises in AEC industries.

It is interesting to note that the described industrial puzzles represent paradigmatic crises that cannot be resolved within the context of existing paradigms. These crises stimulate a paradigm change in AEC industries wherein the AEC practitioners are in search for new business models to overcome the shortcomings of existing models. While a number of alternative business models and practices are explored, BIM tends to be one of the most promising technologies that can improve the quality of information flow and reduce redundancy. Furthermore, BIM can reduce the time used during the construction documentation phase. By implementing BIM, AEC practitioners can have more time to spend during the design phases which can be used to conduct a series of analyses emphasizing building energy and environmental performances. Therefore, BIM can be considered a potential puzzle-solution of the emerging AEC business models.

At a theoretical level, this study can be considered a paradigm shift as presented by Khun. During this fuzzy phase, there are a number of candidate paradigms (business models and practices) that have been

explored, but there is no particular paradigm that dominates the AEC fields. However, there is a body of evidence representing a pattern whereby BIM will become a foundation of the new dominant paradigm when a paradigm shift occurs. According to Kuhn, research under a paradigm must be a particularly effective way of inducing paradigm change. Therefore, this study is intended to focus on the extent of BIM technology since BIM can be considered a potential induction of paradigm change. This study is aimed to investigate the extent of BIM in knowledge capturing and supporting AEC practitioners' design decisions.

1.7.2. Research Contributions

As research in the revolutionary phase of AEC industries, this study is intended to serve as an induction of paradigm change by introducing the benefits of BIM in knowledge capturing and improving the quality of design decisions within the domain of sustainable architecture design. The outcomes of this research include a prototype of BIM interoperable web-based DSS and a guideline summarizing basic requirements for developing the fully functional DSS. As a part of this dissertation, a prototype of a BIM interoperable web-based DSS will be developed. The prototype development can be considered a part of an ongoing research conducted within the College of Architecture and Urban Studies (CAUS), Virginia Tech. The ongoing research emphasizes the development of an integrated web-based DSS for holistic building design. Through a post-use interview study, this research also comprises the possibilities that DSS can be developed and is useful for improving the quality of design decisions. Additionally, the benefits and detriments of using DSS during design phases are captured. The information collected from the post-use study is used for developing a guideline which is a set of essential requirements for the development of the actual fully functional DSS that will be developed in the future. In broader view, this study expands the knowledge necessary for the development of BIM technology toward decision support and artificial intelligence domains.

The further outcomes of this research as a fully functional DSS also could be beneficial to design education, professional practice, and legality of information sharing. For design education, the DSS could help design students faster constructed a well-organized way to make semi-structured decisions during the design process. For professional practice, the DSS could help expand the knowledge base of AEC practitioners in a variety of sustainable building design strategies such as high-performance building envelope, daylighting, and heating, ventilating and air-conditioning systems. For legality of information sharing, the DSS could help assert the legality of documenting design decisions which may be useful in the litigation between project parties.

1.8. Research Design Approach Summary

In summary, Chapter 1: Introduction has introduced an overview of this research and how this research is constructed. Chapter 2: Literature Review provides background information necessary to understand the contents of this research. Chapter 3: Methodology describes the methods used in prototype DSS development and post-use interview study. Chapter 4: Prototype of BIM Interoperable Web-Based DSS and Chapter 5: Post-Use Interview Study Data Analysis, Interpretation, and Results; then apply the

methods from Chapter 3 to generate the outcomes. Lastly, Chapter 6: Summary and Discussions discusses the outcomes, introduces the applications of this research, and recommends the continuing research that could be conducted in the future.

Chapter 2

2. Literature Review

2.1. Decision Support Systems in the Domain of Architectural, Engineering, and Construction Practices

2.1.1. Decision Support Systems and Architectural, Engineering, and Construction Practices

In order to effectively, efficiently, and economically develop a Decision Support System (DSS) within the domain of Architecture, Engineering, and Construction (AEC) practices, it is important to understand the relationship between DSS and AEC practices. Based on the introductory information discussed in Chapter 1, the pattern of new AEC business models represents the needs to increase the efforts of and corroboration among design practitioners and other project stakeholders during the design phases of a building project. Such patterns also represent the needs of sustainable building and integrated practices whereby the building information modeling technology tends to be an important driver of the new AEC practices. The understanding of this relationship can help determine the adaptability and benefits of using DSS to enhance design processes and design decisions during the design phases of the building lifecycle.

Based on the nature of design decisions and their impacts on the outcomes of a completed building project, DSS can be considered one of the potential support technologies that can help improve the quality of design decisions involving the design phases of new AEC practices. According to Gorry and Scott-Morton (1971) and Keen and Scott-Morton (1978), DSS can be defined as the interactive computer-based information systems that utilize data and/or models to help make semi-structured decisions. Based on this definition, the DSS tend to be highly adaptable to the design related phases of a building project due to the nature of decisions that design practitioners are involved. This is because the decisions frequently found during the design related phases including the pre-design, schematic design, and design development phases show the characteristics of semi-structured decisions whereby some or all phases of the four-phase decision making process of intelligence, design, choice, and implementation are not strongly structured, but not too weakly structured. However, the structure of decision making phases can be diverse depending on different design practices. For example, Vries and Wagter (1990) considered that the architectural design process is unlike design processes of other disciplines because its three characteristics: ill-structured, open-ended, and no fixed starting point. Thus, the architectural design process represents the four-phase decision making process that contains more unstructured elements than other design processes.

The decisions that AEC practitioners, especially designers, encounter during the design phases not only replicate the semi-structured decisions, but also have strong influences on the project and business success, according to Pollalis and Bakos (1996). For the project success, design phases can be considered the first phases in the building project lifecycle; in consequence, decisions made during design phases are critical to determine the cost, quality, and maintainability of the complete project. This is because such decisions restrain the feasible choices in the later phases of a building project. For the business success, based on today's economy, design has become increasingly important due to the rapidly changing business environmental conditions (e.g., consumer needs, competitive conditions, technological progress, and product requirements). The economic growth has raised standards of living beyond basic needs; therefore, design has become an important aspect of product differentiation. Since design is highly unstructured and can be considered an art rather than a science, design offers a large potential for differentiation and sustainable advantage.

Since decision support systems have significant potential to improve the quality of design decisions during the design phases of a building project, it is interesting to examine how such technology can be woven into design processes throughout the design phases. Von Wodtke (2000) considered that the general design process is derived from the creative process consisting of research, analysis, synthesis, and evaluation. It is interesting to note that, although using different terms, the four-phase decision making process also shares a similar pattern with the creative process. Based on the American Institute of Architects (AIA)'s architect basic services, Von Wodtke examined the creative process involved in the phases of architect services in architectural design projects. The phases include predesign, preliminary design, design development, construction documents, bidding, and construction administration. Von Wodtke suggested that the stages of the creative process and the phases of design are intertwined, for each phase of a design project involves the creative process. The focus in the initial phases is on design whereas the focus in the later phases is on production. Each phase in design consisting of creative thinking and design communication can benefit from digital tools with new media or computer-based systems.

Although computer-based systems can be used in every phase of a design project, the use of such systems for decision support may be different depending on design disciplines. For example, in their paper, A CAAD Model for Use in Early Design Phases published in 1990, Vries and Wagter suggested that, in the architectural design process, the early design phases are vaguer than those found in design processes of other design professions. Thus, Vries and Wagter introduced an alternative design process to the research-analysis-synthesis-evaluation process, which can be applied to the early design phases in architectural design projects. Vries and Wagter's design process consists of three cycles: decision, structuring, and development. In this process, the decision cycle is considered the innermost cycle that represents a simple synthesis-evaluation cycle. From their observations on design processes, Vries and Wagter found that design decisions are mostly made during this cycle and continue to influence the later cycles. During the decision cycle, the research and analysis phases are excluded since design

information is not sufficient and/or structured enough to be researched and analyzed. Continuing from the decision cycle, the structuring cycle is a step whereby multiple decision cycles are structured in order to deal with more complex decisions. The development cycle, the outermost cycle, is then responsible for controlling the development of the design. The development cycle can be considered the compilation of completed structuring cycles in which the information is sufficient and structured enough to perform the general design process. In their research, this alternative design process was transformed to a prototype program that is capable of handling unstructured design information during the early design phases.

In addition to Vries and Wagter (1990), Pollalis and Bakos (1996) studied the technology used during the design and construction phases of multiple bridge construction projects in their paper, Technology in the Design Process. Pollalis and Bakos categorized the technologies employed in bridge construction projects into two categories: support technology and implementation technology. The support technology serves to carry out symbolic problem-solving activities at a certain level of abstraction. Support technology tools include, but are not limited to, drawings, models, and computer graphics. The support technology also consists of knowledge components such as engineering methods, algorithms, and implementation techniques. The support technology facilitates design processes from the conceptual design stage, to design development and construction design. The implementation technology is an inherit part of the design process in which the ultimate goal of design is to produce an artifact. The implementation technology facilitates the physical realization of the design product. The implementation technology includes, but is not limited to, the use of materials, construction techniques, and construction tools and machinery. In design process, designers typically employ only support technology when products are symbolically represented. During design, the implementation technology involves as a series of assumptions that will not be tested until the construction begins. Pollalis and Bakos suggested that the right assumptions on the implementation technology during the design process considerably help improve the efficiency of design.

According to Pollalis and Bakos, the support technology is not limited to computer-based technologies, but computer-based technologies have opened new frontiers in the support technology. The term “new frontiers” used by Pollalis and Bakos was based on the cost-benefit characteristics of technology described in their paper, meaning that, when compared to existing non-computer-based technologies, computer-based technologies are capable of enabling designers to produce better outcomes while using fewer resources. In their paper, the computer-based technologies include the computer-based process and coordination technologies. The computer-based process technology is founded upon the use of computers to store, retrieve, and manipulate data. The technology allows quick alternative representations and feasibility studies of objects based on large sets of data. The coordination technology is founded upon the use of computers to facilitate the exchange of information between design phases, provide design information for the later phases of a building project, and improve the communication among project stakeholders. The technology enables the coordination of multiple tasks with a set of common goals. The computer-based process and coordination technologies in combination represent the

Information Technology (IT), which can be defined as the technology focusing on the use of electronic means to expand the limits of individual and organizational rationality.

Based on Groat and Wang (2002) suggestions on the uses of literature review, the literature review discussed in this subsection is used to understand an idea's genetic roots, focusing on the adaptability and benefits of using DSS to enhance design processes and decisions during design phases of the building lifecycle. The understanding gained from the literature review is summarized in Table 2-1. The literature review indicates that DSS have enormous potential in enhancing design processes and improving the quality of design decisions throughout the design phases of new AEC business models. Based on the nature of decisions found in design phases, DSS can provide the powerful computer-based processes building upon a large set of data, computational models, and organizational knowledge to enrich the design decisions. Additionally, DSS offer supports and services not only for information sharing between design phases and implementation phases of a building project, but also between building project stakeholders. Furthermore, in their paper, Pollalis and Bakos (1996) suggested that the technology cannot be separated from design intensions and the creative process. Despite the fact that design intensions influence the selection of appropriate technology, the available technology determines the feasibility of particular designs, which affects design processes and outcomes. Thus, the use of advance technology in the design process should result in better design. Based on Pollalis and Bakos's suggestion, DSS can be considered one of the available technologies that align with the intentions of new AEC business models emphasizing on the importance of decisions made during design phases. DSS are also capable of enhancing design processes and help produce better design outcomes.

Table 2-1: Summary of literature review on the decision support systems and architectural, engineering, and construction practices

Literature	Theory	Understanding
Gorry and Scott-Morton (1971)	Define the definition of computer-based Decision Support Systems (DSS)	Recognize that DSS are developed to help decision makers make unstructured decisions using data and/or models
Keen and Scott-Morton (1978)	Extend the definition of computer-based Decision Support Systems (DSS)	Recognize that DSS are also developed to help decision makers make semi-structured decisions using data and/or models
Vries and Wagter (1990)	Discuss three characteristics of the architectural design process in the early stages: ill-structured, open-ended, and no fixed starting point	Recognize that the architectural design process, especially in the early stages, encompasses more unstructured decisions than other design processes, which can benefit from DSS
Pollalis and Bakos (1996)	Discuss two technologies employed in bridge construction project: support technology and implementation technology Suggest that Information Technology (IT) is one of the most promising support technology that combines both computer-based process and communication technology	Can be confident that DSS as Computer-Based Information Systems (CBIS) can be implemented as support technology in the building industry
Von Wodtke (2000)	Suggest that any phase in design that encompasses of creative thinking and design communication can benefit from digital tools with new media or computer-based systems	Can be confident that DSS can be used as part of design processes employed by design practitioners in the building industry

2.1.2. Current State of Decision Support System Technologies in the Domain of Architectural, Engineering, and Construction Practices

In addition to the relationship between DSS and AEC practices, it is also interesting to review the current state of DSS technologies within the domain of AEC practices. Based on the objective of this study, the DSS technologies can be restricted to the AIS SIGDSS's knowledge-driven DSS category such as the knowledge-based DSS (KBDSS) and intelligent DSS (IDSS) as well as the compound DSS that combine the knowledge-driven DSS with other types of DSS. The intents of this review are to identify the commonality and understand the trends of DSS development especially for the sustainable building, integrated, and building information modeling practices.

Decision Support Systems and Sustainable Building Practice

In new AEC business models, sustainability is likely to become one of the common goals of industry-wide practices. The awareness of sustainability originally came from the notion of sustainable development described in the United Nations (UN)'s report, *Report of the World Commission on Environment and Development* (1987), as the development that implies meeting the needs of the present without compromising the ability of future generations to meet their own needs. In the building industry, the term "sustainability" can be considered a buzz word which spreads to include the practices of energy-efficient, green, sustainable, and regenerative buildings depending on the building performance, over the full lifecycle, concerning impacts on natural resources and surrounding environments. Concisely, based on ASHRAE (2006) and Stein (2006), energy-efficient buildings achieve high performance in reducing net negative energy impacts. Green buildings achieve energy-efficient building quality and high performance in minimizing net negative environmental impacts. Sustainable buildings achieve green building quality and high performance in producing no net negative environmental impacts. Regenerative buildings achieve sustainable building quality and high performance in generating a net positive environmental impact. In this dissertation, the term "sustainable building practice" is frequently used as an umbrella term for the practices of energy-efficient, green, sustainable, and regenerative buildings.

Within the scope of sustainable building practice, the accomplishments of a building project significantly rely on the building performance pertaining to impacts on natural resources and surrounding environments. Such performance-driven practice requires support technologies that can help designers acquire, consolidate, and manipulate assumptions about implementation technologies that influence building performance as well as help designers foresee the building performance resulting from their design decisions early during design phases. As a response to these needs, building performance simulation tools are increasingly adopted by AEC practitioners in design phases of sustainable building projects.

According to Malkawi (2004a), in his paper, Developments in Environmental Performance Simulation, performance simulation has been evolving in architectural design over the past decade. Since the beginning of development, simulation tools have been developed as single domain tools for predicting the

performance of the building design in areas including, but not limited to, thermal, fluid dynamics, lighting, acoustic and structures. Such tools have been developed in the procedural computational environments based on the maturing algorithms of particular domains. DOE-2 and BLAST are examples of single domain tools. With the advancements in technology, the computational environments have shifted from procedural to object-oriented environments. This shift causes the architectural change in building performance simulation environments. EnergyPlus, released in 1995 with the best features of DOE-2 and BLAST, is one of the tools developed upon the object-oriented approach. The greatest advantage of the object-oriented approach is that it enables the development of environments and frameworks that can achieve design analysis integration based on semantic representations which support object and data interaction. Based on this approach, maturing algorithms and data from multiple domains can be integrated in a single environment to support designer decisions concerning the whole building performance. This new environment represents the architecture of a decision support system. Since building performance DSS rely on simulation or optimization models, such DSS can be considered the model-driven DSS categorized by AIS SIGDSS.

When seeing the building performance simulation environments as decision support systems, the performance of simulation such as the accuracy of simulation can be potentially improved by incorporating other DSS components such as the knowledge-based and intelligent components into the simulation environments. In such compound or hybrid DSS, the knowledge-based and intelligent components are capable of improving the system functionality and performance of computational models and data queries. For example, in his paper, Virtual Agents as A Decision Support for Thermal Simulation, Malkawi (2004b) described the Artificial Intelligence (AI) technique used in a decision support system developed for building thermal simulation. The AI technique is the use of generic rules[4] as virtual agents[5] that can aid in detecting thermal problems and interact through a blackboard model. This AI component was developed to resolve the shortcomings of the systems that provide either rules or heuristic[6] knowledge for approximate evaluation that aids in the conceptual to intermediate design phases for building thermal problems. The shortcomings of such systems include the lacks of design aid for specific design problems and potential optimal design solutions as well as for the prediction of exact thermal design location.

[4] Rule-based system is a system in which knowledge is represented terms of rules such as a system based on production rules (Turban et al. 2007).

[5] Intelligent Agent (IA) is an expert or knowledge-based system embedded in computer-based information systems or their components to make such systems smarter (Turban et al. 2007).

[6] Heuristics is an informal, judgmental knowledge of an application area that constitutes the rules of good judgment in the field. Heuristics also encompasses the knowledge of how to solve problems efficiently and effectively, how to plan steps in solving a complex problem, how to improve performance, and so forth (Turban et al. 2007).

Figure 2-1 demonstrates the simplified framework of a problem detection model (AI subsystem) for building thermal problems using multiple knowledge reasoning agents described in Malkawi's (2004b) paper. In the framework, the model uses a set of inputs from a thermal simulation program which is considered a transit analysis program that provides hourly heating and cooling load estimates for residential and commercial buildings. The generic rules take inputs from the simulation program and identify the contribution of loads on the total building performance. The rules are divided into three groups to identify the contribution of loads in three areas: 1) the worst zone of the building, 2) the major element or systems of potential problems, and 3) the precise elements or systems, taking into consideration the first two groups of rules. After identifying the contribution of loads, the generic rules then activate the visual agents. The agents are fabricated from eight knowledge sources, a blackboard, and a controller. The knowledge sources are building elements and systems that affect the thermal behavior of the building. The knowledge sources include 1) walls, 2) glazing, 3) roofs, 4) ventilation, 5) infiltration, 6) occupancy 7) lighting, and 8) equipment. The blackboard is an area of working memory reserved for the dynamic portion of the cache and static facts. The controller of agents monitors and performs reasoning process between the blackboard and the knowledge sources. Based on this framework, virtual agents can provide problem identification without the use of rules of thumb. Furthermore, the uses of virtual agents also allow the development of a robust DSS that could identify potential building problems based on thermal simulation.

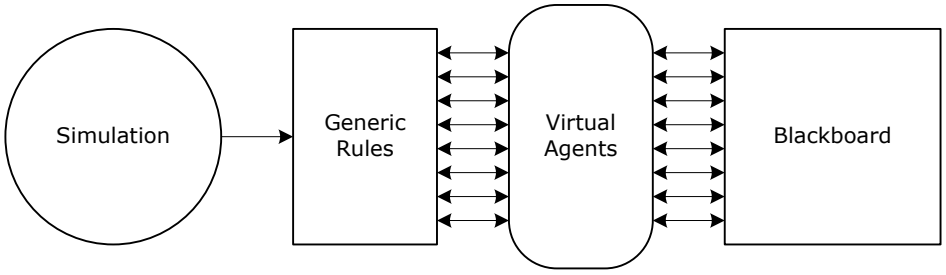


Figure 2-1: Simplified framework of agents, adapted from Malkawi (2004b)

In addition to Malkawi's (2004b) study, there are a few studies that also used KBDSS and IDSS to improve the performance of simulation or optimization models in model-driven DSS such as in the building performance simulation environments. For instance, Malkawi et al. (2005) studied a decision support evolution model using Genetic Algorithm[7] (GA) as the evolution algorithm and Computational Fluid Dynamics (CFD) as the evolution mechanism. The system uses an interactive approach that allows design to be evaluated using CFD analysis automatically to maximize several thermal and ventilation criteria. In addition to this generic-driven object and data integration approach, there is another approach for improving the performance of building simulation described in Malkawi's (2004a) paper as the

[7] Genetic algorithm is a software program that learns in evolutionary manner, similar to the way biological systems evolve (Turban et al. 2007).

process-driven integration approach. This method focuses on the effective use of performance simulation tools in the design process with full participation from the design team including the expert consultants. For example, Flager et al. (2008) investigated an application of the Process Integration and Design Optimization (PIDO) technique used in the aerospace industry to an AEC case study: the Multidisciplinary Design and Optimization (MDO) of a classroom building for structural and energy performance. In this study, Flager et al. employ Phoenix Integration's ModelCenter, process integration software used in the aerospace industry, to automate the process of design codes typically used during the design lifecycle. The software automatically transfers design data from a geometric design program to an energy analysis program to structural analysis programs to design exploration and optimization tools, thereby allowing architects and engineers to concentrate on design analysis, instead of the complexities of data integration. Similar to Malkawi et al. 's (2005) study, this study also applied the GA for design exploration and optimization. These examples indicate that there is a growing interest in utilizing KBDSS and IDSS as part of DSS in the context of sustainable building practice.

According to Keysar and Pearce (2007), sustainability is not only one of the common goals, but also can be considered an innovation in AEC practices based on Rogers's (2003) principles of diffusion of innovations. In their paper, Decision Support Tools for Green Building: Facilitating Selection among New Adopters on Public Sector Projects, Keysar and Pearce recognized that there is an increasing number of private AEC firms who participate in public building projects. While building projects in the public sector increasingly require the sustainable building practice, the large majority of private firms lack experience with sustainable building projects, expertise in sustainable building design support and implementation technologies, and past satisfactory building performance outcomes. Keysar and Pearce used an analysis tool introduced by Rogers to examine the current state of sustainability adoption throughout the industry. The results of their analysis show that the majority of firms fall within the segment of the adoption curve beyond the innovators and early adopters. The results also indicate that, when considering each individual deficiency of private firms, this group of firms may fall within the early majority, late majority, or even laggard category of adopters. The results imply that such firms are getting behind in their business competition especially in the public sector and sustainable building practice. In consequence, there are the needs to incorporate sustainability principles as part of the design of capital projects which can be facilitated by the appropriate adoption of tools to support effective decision making during design (Keysar and Pearce 2007).

In their paper, Keysar and Pearce proposed a database tool to help the majority of AEC practitioners select appropriate tools to improve their design processes and decisions toward the sustainable building design. In order to develop the database, more than two hundred decision support tools including, but not limited to, performance simulation, lifecycle assessment, and education and professional development tools used in LEED[8]-based green building projects were reviewed and classified. The tools were

[8] LEED is an abbreviation of the Leadership in Energy and Environment Design.

categorized based on the hybrid organizing framework by topical area, purpose, and tool type described in their paper. Keysar and Pearce also characterized these tools to rate each tool according to a LEED-based operationalization of the relative advantage and trialability. The relative advantage was defined as the tool's ability to yield immediately practicable information to support determination of compliance with LEED credits, or to provide information essential to support documentation of that compliance. The trial ability was defined as the initial cost of the tool to the user. The characterization of tools indicates that, within 275 reviewed tools, there are only 17% of the tools, around 47 tools, rated high in both relative advantage and trialability.

Although Keysar and Pearce's database tool does not represent the knowledge-driven DSS, the approach that they used for conducting the characterization of green building decision support tools is very interesting to discuss. In their paper, Keysar and Pearce implemented Rogers's (2003) characteristics of innovations to determine the attributes of tools and create the rating system along with a LEED-based operationalization. The characteristics of innovations include 1) relative advantage, 2) compatibility, 3) complexity, 4) trialability, and 5) observability. These five characteristics of innovations provide a useful framework not only for evaluating and comparing candidate tools, but also for predicting their likelihood of success or adoptability. Keysar and Perce suggested that developers who develop new tools should consider these attributes when designing tools to appeal to the market niche of tool adopters new to green or sustainable building.

Sustainability is soon-to-be one of the common goals and can be considered one of the innovations in AEC practices. As a goal in AEC practices, the building performance concerning impacts on natural resources and surrounding environments tends to be one of the most important aspects of the sustainable building practice. In order to improve design processes and decisions, AEC practitioners can benefit from incorporating the support technologies, such as building simulation environments into their design processes throughout the design phases of the building lifecycle. From the technology standpoint, the object-oriented technology enables the integration of design support tools, replicating the framework of DSS. Such DSS can be enriched by including the knowledge-based DSS as part of the systems to provide knowledge support for designers, as well as to improve the accuracy and efficiency of computational models and data queries. As an innovation, sustainability has been increasingly adopted by the majority of AEC practitioners. However, this group of practitioners lack experience with sustainable buildings, expertise in sustainable building design support and implementation technologies, and past acceptable building performance outcomes. This group of practitioners, about half of the industry, can certainly benefit from employing appropriate decision support tools to help improve their design processes and decisions toward the sustainable building practice. In order to effectively develop decision support tools or systems to serve this group of practitioners, developers should take into account the five attributes of innovations to increase their likelihood of success or adoptability.

Decision Support Systems and Integrated Practice

In addition to the sustainable building practice, the integrated practice also has significant potential to benefit from the use of decision support systems. According to the American Institute of Architects (AIA 2009), Integrated Practice (IP), also known as Integrated Project Delivery (IPD), is a process that enables project stakeholders including, but not limited to, owners, architects, engineering consultants, and contractors to work collaboratively throughout the project lifecycle. As one of the innovative AEC business models, IP has been rapidly adopted in many recent building projects because it is proven to reduce building design and delivery waste together with promote sustainability. IP influences early contributions of knowledge and expertise which allows all project team members to better realize their highest potentials while expanding the value they provide. This communications-driven practice can benefit from the utilization of computer, collaboration, and communication technologies to support groups in tasks such as decision making tasks. In consequence, the technologies that fall within the category of communications-driven DSS or group DSS defined by the AIS SIGDSS can be considered the prospective technologies for enhancing the collaboration between AEC practitioners in the context of integrated practice.

In order to achieve the goals of IP, AEC practitioners are seeking the interactive and integrated environments to support collaboration and communication among practitioners. As stated by Haymaker and Suter (2006) in their paper, Communicating, Integrating, and Improving Multidisciplinary Design and Analysis Narratives, building projects in today's business environment have become increasingly complex and are required to achieve a collective number of economy, ecology, and equity goals. Such situations indicate the needs for the Multidisciplinary Design and Analysis (MDA) which demands AEC practitioners to understand the interdependencies and make tradeoffs between their discipline-specific goals and the goals of other disciplines. In order to achieve their goals, AEC practitioners are expected to develop narratives, collections of information representations with explicit dependencies among them, for their works in their own field and interweave them with narratives of practitioners in other fields. However, from their observations on MDA processes across case study projects, Haymaker and Suter found that these narratives are often not effectively represented or managed which make multidisciplinary projects time consuming and error-prone. Therefore, Haymaker and Suter propose a visual language and framework, called Narratives, for constructing and managing AEC information representations and dependencies between these representations. The framework was formulated upon the methods used to construct information and specify its dependency on other information and methods to control the integration of this information as the projects progress. Such methods can be categorized into three categories including representation, reasoning, and management methods. Narratives can be constructed using a tool, called Narrator, which is a Java desktop application designed to help quickly and visually compose Narratives. In the paper, Haymaker and Suter also discussed about their ongoing investigation on the development of Narrator in three areas including enabling distributed representation and reasoning using web-based technologies, incorporating any data types and reasoning, and improving user interface and immersive information environment.

Narratives and Narrator tool introduced by Haymaker and Suter can be considered one of interesting examples demonstrating the interactive and integrated group support environments. The framework represents a communications-driven or group DSS that utilize computer, collaboration, and communication technologies to support groups of AEC practitioners in tasks. This system has considerable potential to help improve the quality of multidisciplinary design decisions by providing interdependent information representations and reasoning produced by area-specific experts that can be managed in one place. Based on the reasoning method used to formulate the framework, Narratives can also benefit from utilizing the knowledge-driven DSS as part of the communications-driven DSS. As stated by Haymaker and Suter, reasoning in the AEC industry can be categorized into two categories: manual and automated, based on the nature of dependency. Within Narratives, practitioners should be able to specify when the nature of the reasoning is manual and to specify tools or data formats necessary for constructing the dependent information. Furthermore, practitioners should be able to easily define project-specific automated reasoning where possible and to use existing reasoning tools for automated reasoning. The demands of manual and automated reasoning in the integrated AEC business processes such as described in the Haymaker and Suter case indicate the potential implementation of compound DSS consisting of communications-driven and knowledge-driven DSS in the integrated practice domain.

Other than Haymaker and Suter's study, Boddy et al. (2006) studied a compound DSS that utilizes knowledge-driven DSS in conjunction with communication-driven DSS. In their paper, Knowledge Informed Decision Making in the Building Lifecycle: An Application to the Design of a Water Drainage System, Boddy et al. presented a methodology to use a knowledge management environment, C-Sand, and a decision support tool, Advanced Decision Support (ADS). This methodology provides designers with an environment that supports sustainability informed decision making, and bring immediate sustainability knowledge management into the application used in design offices. In this system, C-Sand is employed to promote and disseminate knowledge about sustainable products, techniques, and practice in the construction industry. C-Sand was developed based on the method of intra and inter-organizational knowledge sharing via a set of web-based services, developed using Java Remote Method Invocation (RMI), with a portal interface. Besides C-Sand, ADS is employed for viewing and recording the evolution of project information over time to better support effective decision making. The main output from ADS is a proof of concept demonstration platform based on an integration of the ADS prototype and CAD application, Bentley Systems' Microstation/J. The integration between C-Sand and ADS creates a Knowledge Informed Decision Making (KIDM) environment that brings together information and people who use it. KIDM, when fully developed, is expected to be capable of assisting practitioners in their decision making by taking advantage of the existing and available knowledge. Within this environment, designers' decisions will be supported and informed by knowledge resources, with the reasons for these decisions made by designers feeding back to the body of knowledge. This example reveals that knowledge-driven DSS can be used in conjunction with communication-driven DSS to improve the quality of group decisions in integrated work environments.

Figure 2-2 represents the management of sustainable construction knowledge in C-Sand. The C-Sand services are first used to create representations of real-world resources, shown as rectangular shapes, defined as Knowledge Representations (KR), shown as hexagon shapes. KR can then be linked to other KR using a construct characterized as Knowledge Representation Links (KRL) represented by the arrow lines. As stated by Boddy et al. (2006), once a resource is known to the system, it will be stored as a KR and available as a search result to end users. C-Sand has ranking and relevance feedback mechanisms on search results in a process similar to bookmarking. The system allows a user to choose the search results that best suit their requirements and rate the real-world resources on a four-point scale. The system takes the rated search results to create a user interest and stores as another KR. The KR of a user interest represents a set of resources that a particular user takes in to consideration as a useful set of reference resources in a particular knowledge domain. A useful set of references is then defined as the initial search query in a particular domain. The knowledge representations of interests are also public by default which makes them accessible for other users. When a user finds a KR of interest as a search result, such user can simply connect to peers who have such similar interest to create physical social networks outside the C-Sand environment. For example, in one project, an architect may be assigned to work on the layout of a graywater drainage system. This architect may layout the system based on the KR that represents the most useful set of resources used by project engineers. Later, in another project, another architect may be assigned the similar task in designing the graywater drainage system. The later architect can then use resources linked through the KR of the previous architect to design the system. In some cases, the architect who works in the later project may encounter design problems that are different from the architect who worked in the previous project and may need to consult that architect. The later architect can then find the contact of the former architect or even the contacts of the project engineers and request for their assistance. Based on the described principles, the knowledge management environments such as C-Sand have considerable potential to help improve the quality of multidisciplinary design decisions within integrated building projects.

In the domain of integrated practice, decision support systems can be considered one of the information technologies that have a large potential to be adopted by AEC practitioners because of their capabilities to support multidisciplinary or group decisions. According to Fjermestad and Hiltz (2000) cited by Boddy et al. (2006), in an analysis of 54 implementations of communications-driven DSS or group DSS, it was shown that over 80% of organizations using this type of DSS, showed improved performance. Communications-driven DSS such as discussed in the Boddy et al. example also can be used in combination with knowledge-driven DSS to improve the quality of multidisciplinary decisions based on organizational knowledge resources. From the technological point of view, the web-based technologies, data integration techniques, and visual-based user interfaces can be considered major complimentary technologies for DSS development within the IP domain. It is interesting to note that, when developing a DSS to support multidisciplinary projects, there are some issues which should be taken into consideration. Based on Boddy et al. (2006), the issues may include, but are not limited to, the records of

intent behind decisions, ownership, rights, and, responsibilities, the dependencies of information, the notification of information changes, the information version control, and the evolution of schema throughout the course of a project.

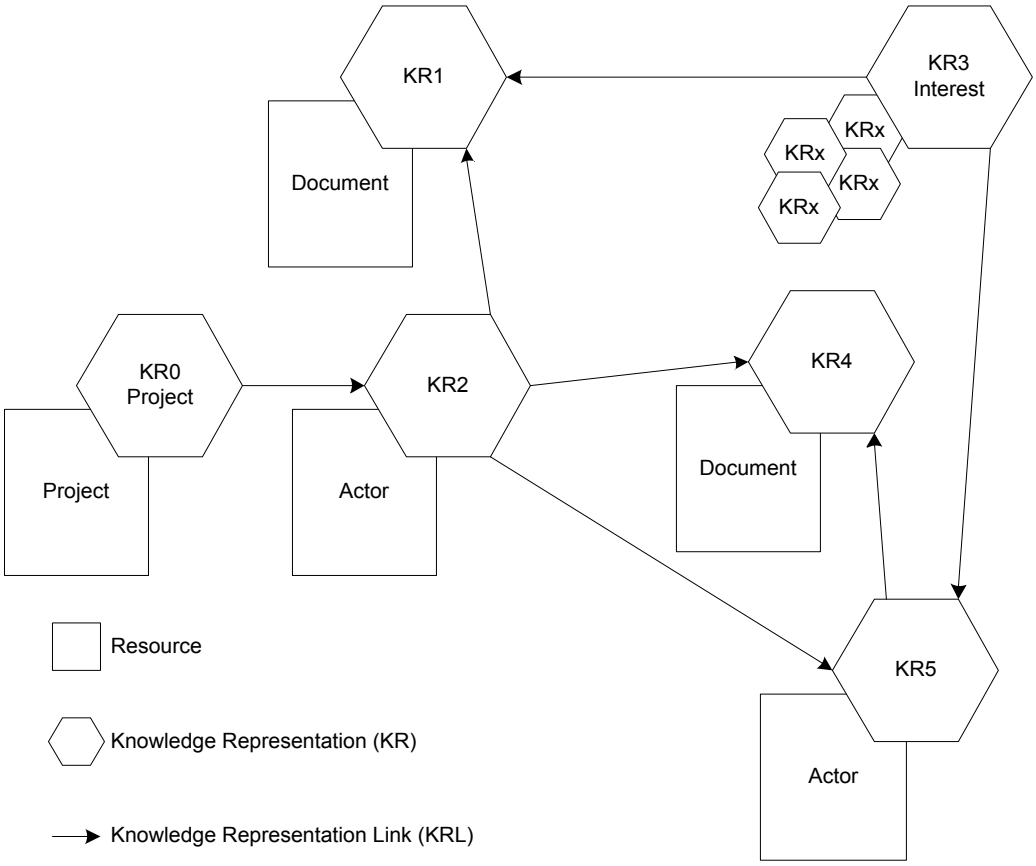


Figure 2-2: Knowledge management in C-Sand, adapted from Boddy et al. (2006)

Decision Support Systems and Building Information Modeling Practice

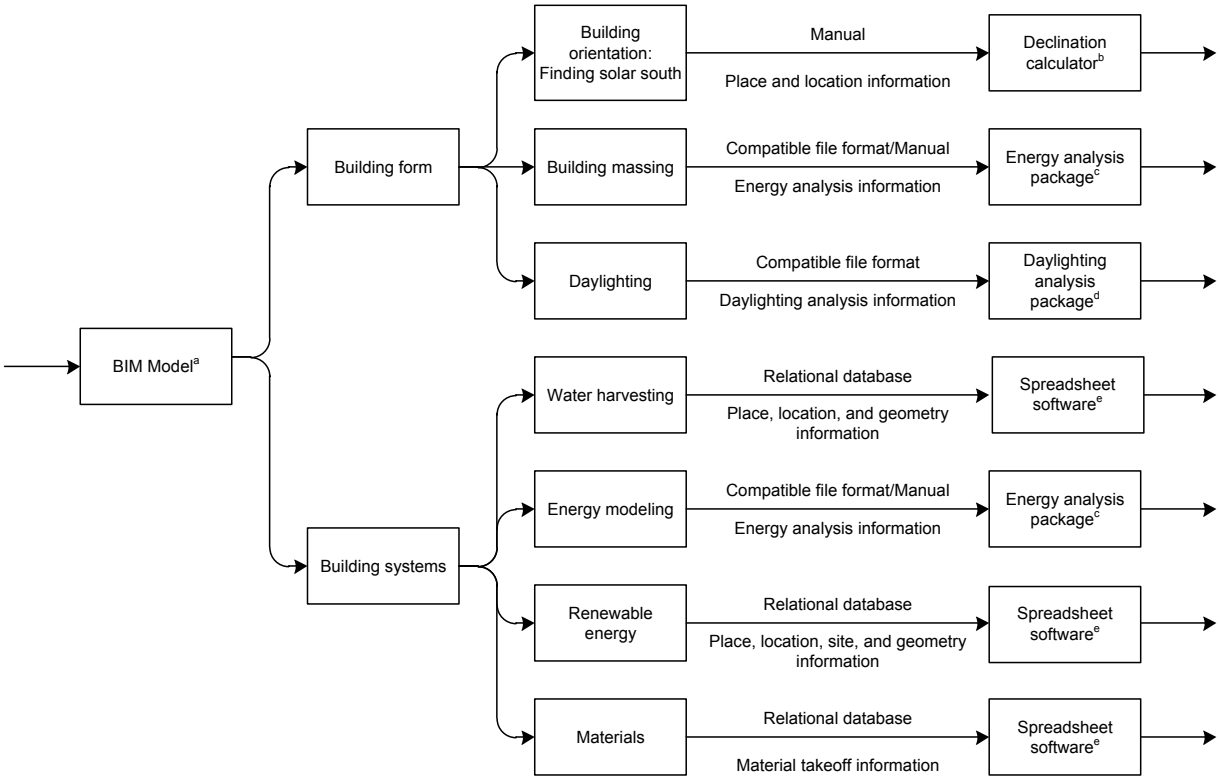
Referring to the information discussed in Chapter 1, Building Information Modeling (BIM) technology has been recognized as one of the innovations in Computer-Aided Architectural Design (CAAD) technology. BIM, as a collection of tools and processes, is a disruptive technology to existing CAAD technologies such as Computer-Aided Drafting (CAD) technology employed by AEC practitioners during the past decades (Onuma 2009). Based on Schilling's (2008) types of technological innovation, BIM can be considered an architectural innovation that changes not only how the overall system is designed, but also how the system components interact with each other. The new system architecture built upon the object-based parametric model principles enables BIM models to provide collective multidisciplinary information about buildings in the way that they are designed, constructed, operated, and demolished rather than the abstract representations of buildings normally found in CAD documents. As stated by Onuma (2009), information captured using BIM tools, the initial "I" in "BIM", is the greatest value of BIM implementation. Information contained in BIM models has considerable potential to be restructured and used as decision

support information for improving the quality of design decisions in AEC practices. From the innovation adoption point of view, the capability of BIM in providing decision support information has been increasingly recognized by the early majority group of AEC practitioners after BIM was adopted by the innovator and early adopter groups of practitioners for decision support and intelligent modeling. This state of technology diffusion indicates the growing demand in using BIM to improve the quality of design decisions; in consequence, it is interesting to examine the applications and current state of BIM technology in this area of interest.

In today's practice, there are two major approaches in which BIM can be used to improve the quality of design decisions during the design phases of the building lifecycle. One approach is to use BIM as a center of design activities in cooperation with other task-specific tools. For example, in their book, *Green BIM: Successful Sustainable Design with Building Information Modeling* published in 2008, Krygiel and Nies demonstrated an interesting framework whereby BIM can be used as a center of sustainable building design activities. The framework represents an order of operations including 1) understanding climate, culture, and place, 2) understanding the building type, 3) reducing the resource consumption need, 4) using free/local resources and natural systems, 5) using efficient manmade systems, 6) applying renewable energy generation systems, and 7) offsetting remaining negative impacts. In the document, BIM tools were seen as building information and database management tools whereby they can be used for 1) exporting building geometry, 2) counting building elements, 3) sorting building elements, 4) performing basic calculations based on building elements, and 5) producing reports for information sharing and communication. It is interesting to note that these capabilities represent the basic capabilities of the Database Management Subsystems (DBMS) used in decision support systems. With these capabilities, BIM tools can be considered data-driven decision support tools that are capable of reorganizing raw data into decision support data used for supporting design decisions. Nevertheless, BIM tools alone are not likely to be used throughout the framework. Therefore, Krygiel and Nies used BIM tools in conjunction with a set of analysis tools to better inform their design decisions. In the framework, BIM tools are used together with a set of analysis tools to support design decisions mainly on the building form and building systems. For the building form, the tools are used for assisting design decisions concerning building orientation, building massing, and daylighting. For the building systems, the tools are used for supporting design decisions for water harvesting, energy modeling, renewable energy, and sustainable materials.

Figure 2-3 demonstrates the flow of information within Krygiel and Nies's framework. In the figure, there are three methods used to transfer building information from a single BIM model to task-specific applications. First, the BIM model can be used to provide specific information for task-specific applications that cannot directly process information contained within the BIM model. In this case, the information is required to be manually input to the task-specific applications. For example, the BIM model can be used to provide information about building location including building latitude and longitude to help designers find the solar south through the solar declination calculator such as the one provided by the National

Geophysical Data Center (NGDC). Second, information contained in the BIM model can be exported to specific file formats supported by task-specific applications. For example, the BIM model created in Autodesk's Revit Architecture can be exported to the 3D Studio (.3ds) file format to perform daylighting analysis in Autodesk's 3D Max. Third, BIM model can be directly linked to the task-specific applications through the data connectivity middleware such as Microsoft's Open Database Connectivity (ODBC). For example, in order to perform an analysis on the water harvesting, building information contained in the BIM model is linked to Microsoft Excel through ODBC connection using Excel Macro. In this case, the spreadsheet software is used for estimating the monthly water saving from rainwater and graywater harvesting based on building data from the BIM model and external data from the Internet such as the local rainfall data. Krygiel and Nies' example represents an interesting implementation of the BIM-centric approach that use BIM tools in conjunction with different task-specific tools to effectively support design decisions during the design phases of the building lifecycle.



a. A single BIM model is created using Autodesk's Revit Architecture.
b. National Geophysical Data Center (NGDC)'s declination calculator (http://www.ngdc.noaa.gov/geomagmodels/Declination.jsp)
c. Integrated Environmental Solutions Virtual Environment (IES<VE>, http://www.iesve.com), Ecotect (http://ecotect.com), eQuest (http://www.doe2.com/equest), or Green Building Studio (GBS, http://www.greenbuildingstudio.com)
d. IES<VE> (http://www.iesve.com), Daysim (http://irc.nrc-cnrc.gc.ca/ie/lighting/daylight/daysim_e.html), or Autodesk's 3ds Max
e. Microsoft Excel

Figure 2-3: Data flow within Krygiel and Nies's framework

It is interesting to note that the BIM model centric approach involves a high degree of data exchanges between the central BIM model and task-specific applications and in consequence creates a high possibility of data lost, can be potentially error prone, and makes multidisciplinary data integration difficult. Based on the current state of BIM technology, there are two technological improvements aimed to resolve the shortcomings of this approach. One improvement focuses on the interoperability between BIM models and task-specific applications. According to Eastman et al. (2008), conventionally, interoperability has relied on file-based exchange formats. There are four types of exchange formats that can be found in today's AEC practices including 1) Direct, proprietary links between specific BIM tools, 2) Proprietary file exchange formats, 3) Public product data model exchange formats, and 4) XML-based exchange formats. Among these four exchange formats, the interoperability improvement tends to move toward the public domains which include the public data model and XML-based exchange formats due to the need for industrial-wide data integration.

As stated by Eastman et al., the development of data models began in the late 1980's to support product and object model exchanges within different industries, led by the ISO-STEP international standard effort, using the EXPRESS data modeling language. EXPRESS is machine-readable and has multiple implementations, including a compact text file format, SQL and object database implementations and XML implementations. Built upon the ISO-STEP technology, the Industrial Foundation Class (IFC) is currently the only public, non-proprietary, and well-developed AEC product model published in the late 1990's by the International Alliance of Interoperability (IAI), a non-profit industry-led international organization. At the current state of BIM technology, the IFC is more than likely to become the international standard for data exchange and integration within the building industries (Eastman et al. 2008).

In addition to the public product data model exchange formats, XML-based exchange formats are other public domain exchange formats that can be used for exchanging business data between two applications set up for such exchanges. According to Rob and Coronel (2009), XML, the abbreviation of eXtensible Markup Language, is a metalanguage used to represent and manipulate data elements published and standardized by the World Wide Web Consortium (W3C) in the late 1990's. XML is developed for facilitating the exchange of structured documents over the Internet. XML documents are structured by XML Schema which is an advanced data definition language that is used for describing the structure of XML data documents. IFCXML, a subset of the IAI's IFC data model mapped to XML, is an example of XML Schemas used in AEC practices. When they are fully developed, both public product data model and XML-based exchange formats are more than likely to become the solutions for the interoperability problems of the BIM model centric approach.

In addition to the ongoing developments of industrial standard data exchange formats, the developments of BIM model repositories can be considered another technological improvement aimed to resolve the shortcomings of the BIM model centric approach. While the interoperability between BIM tools and task-

specific applications is important for the BIM centric model approach, the management of versions, updates, and changes of data has become another important challenge. This challenge has considerable potential to be overcome by the implementation of BIM model repositories. As stated by Eastman et al., a BIM model repository is a database system whereby its schema is based on a published object-based format. BIM model repositories facilitate query, transfer, updating, and management of individual project objects from a diverse set of applications. In AEC industries, IFC is the only broad building level schema in which an IFC repository can support integration of data generated by multiple applications for use in other applications, support iterations on parts of the design, and track changes at the object level. When the technology is fully developed, it is expected to provide important services including dataset preparation for multiple analyses, bills of material and procurement tracking, construction management, building commissioning, facility management and operations (Eastman et al. 2008).

The technological improvements on industrial standard data exchange formats and BIM model repositories can be considered important drivers behind another approach in which BIM can be used for improving the quality of design decisions. Instead of using BIM as a center of design activities, this alternative approach utilizes information contained in BIM models in conjunction with information from other sources such as from task-specific tools to support design decisions within a single integrated environment. This approach implements the industrial data exchange format and data repository technologies to provide the interfacing capability to connect information from a variety of tools used by AEC practitioners to one information system. The Onuma Planning System (OPS), developed by Onuma Inc., is one of the information systems developed around this approach. According to Onuma (2009), BIM can rather be seen as a process operating around a set of software products than a software product itself. This perspective makes BIM-based systems such as OPS become more like information systems that harmoniously support AEC business processes throughout the building lifecycle. OPS is a web-based application that links BIM design and other tools with a central database built upon open source environments including MySQL and Apache (Eastman et al. 2008). Built upon open architecture, data mining, and BIM technologies, OPS is capable of supporting higher quality design of buildings, promoting higher quality and interoperable information about building environments, and creating transparent information that is then made available to support integrated decision making (Onuma 2010). OPS has been used by the United States Coast Guard (USCG) for managing its large scale facilities located on the Coast Guard Island in Alameda which covers thirty-five facilities totaling 700,000 square feet. According to Eastman et al. (2008), the implementation of OPS in the USCG case can help reduce 98% of facility data updating efforts and reduce the time required to edit a single data point from 2 to 0.4 seconds.

Figure 2-4 demonstrates the flow of information within the OPS framework. The figure shows that the initial information about building can be input to the OPS through Microsoft Excel whereby the data are exchanged using the Comma-Separated Values (CSV) format. OPS has a variety of design and decision support features. An interesting example of OPS's features is the BIMBlobs feature used for automatically generating mass-level building model based on the data imported from the excel file. The model can be

exported to Google Earth using the Keyhole Markup Language (KML), an XML schema used for expressing geographic annotation and visualization on Google Earth. The similar model can also be exported to BIM authoring tools including, but not limited to, Google SketchUP for schematic design and Autodesk's Revit Architecture or Graphisoft's ArchiCAD for design development and construction-level documentation using IFC. Furthermore, the same model can be exported to the analysis tools such as Data Design System's DDS-CAD for engineering system analyses and Autodesk's Ecotect for building performance simulations using IFC. It is interesting to note that the data from analysis tools and Google Earth tend to flow in one direction due to the limited outputs of such applications. However, the model modified using BIM authoring tools based on the analysis results from the analysis tools can be fed back to OPS. The bidirectional communications between OPS and BIM authoring tools allow building information to be incrementally stored and managed in one place via OPS. OPS allows designers to retrieve data stored in the system's data repository and turn stored data into decision support data which can be used to improve the quality of design decisions at hand.

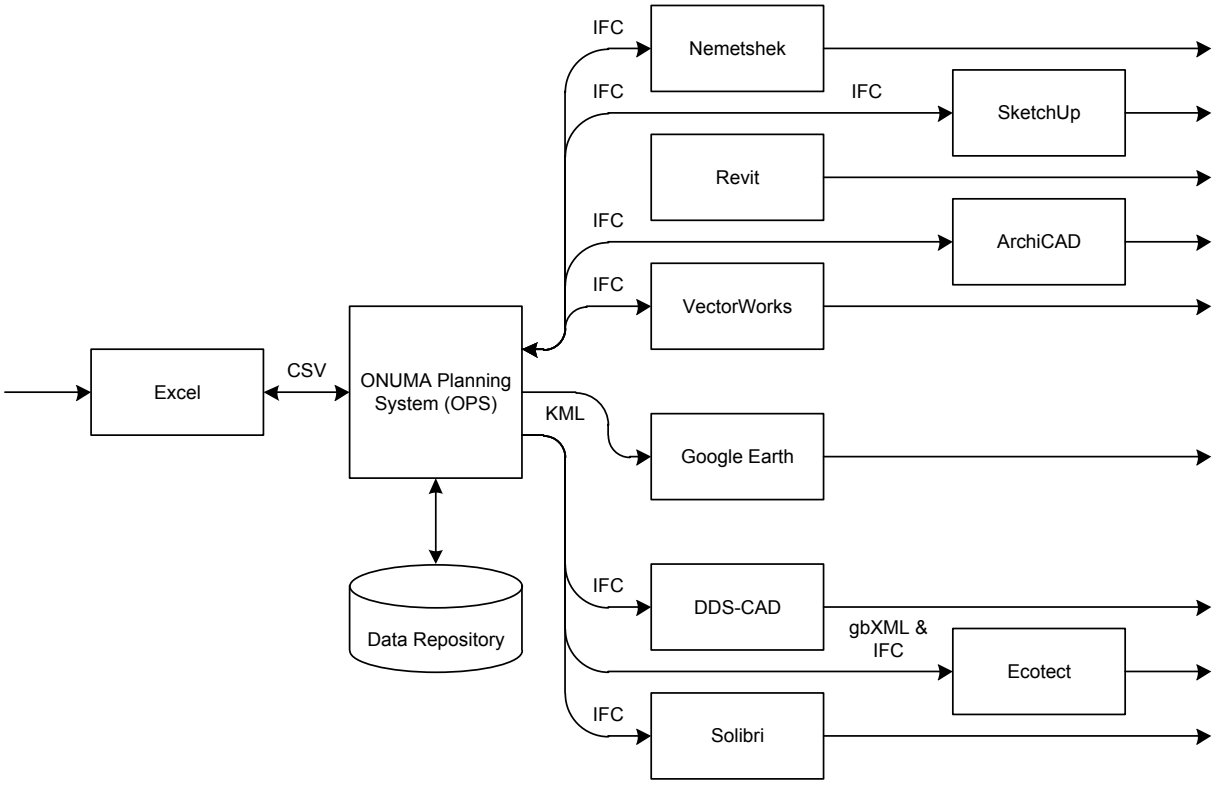


Figure 2-4: Data flow within the ONUMA Planning System Framework

From the current state of technology, BIM can certainly be used as part of AEC business processes for improving practitioners' decisions throughout the building lifecycle. Based on the use of BIM in Krygiel and Nies example, BIM tools provide certain basic capabilities that are similar to the capabilities provided by DBMS. BIM models contain rich building data that can be transformed to decision support data by

such database management capabilities. In practice, BIM can be used as a center of design activities and collaborations in conjunction with task-specific tools employed by diverse experts who participate in the building project. Influenced by the improvements in public domain data exchange format and data repository technologies, BIM tends to be seen as part of information systems that are capable of supporting AEC business processes in every phase of the building lifecycle. BIM-based information systems such as OPS represent the use of BIM as part of data-driven DSS described by the AIS SIGDSS. The BIM-based data-driven DSS can be used to process data into decision support information and present processed information to decision makers. For building project data at the large scale such as the case of USCG's project, the data-driven DSS can be enhanced by data mining systems built upon the similar techniques and tools used in knowledge-based and intelligence systems including, but not limited to, statistical methods, decision trees, and genetic algorithms. BIM is not only a technological innovation of CAAD technology, but also can be considered a compliment technology that enables the product and process innovations in AEC industries whereby the sustainable building and integrated practices tends to be the innovations that highly benefits from BIM-based DSS.

The current state of DSS technologies in the domain of AEC practices indicates that DSS can be used as part of AEC business processes, throughout the building lifecycle, for enhancing practitioners' decision making activities. Within the scope of sustainable building, integrated, and BIM practices, the advantages of implementing DSS to help improve the quality of design decisions during the design phases of building projects have been increasingly recognized as valuable by AEC practitioners. For the sustainable practice, due to its performance-driven nature, design practitioners can benefit from the implementation of compound DSS, which may be composed of model-driven, knowledge-driven, and group DSS in their design processes. Such DSS can extend the capabilities of the tools used in the sustainable building practice, such as building performance simulation tools to generate more accurate simulation results. For integrated practice, the communications-driven DSS play an important role in supporting multidisciplinary or group design decisions both in real-time and when the project is continually evolving. The communications-driven DSS can be used in conjunction with the knowledge-driven DSS to help improve the quality of design decisions based on the organization or team's collective knowledge. For the BIM practice, there is a body of evidence which indicates that BIM shares the similar capabilities with database management systems and can provide rich data about building for the data-driven DSS. For the large data set, data-driven DSS can implement the data mining techniques to effectively, efficiently, and intelligently construct decision support information. It is interesting to note that, while each area of the contemporary practices seems to have a dominant category of DSS, the knowledge-driven DSS can be used to enhance the performance of such dominant DSS.

According to Groat and Wang (2002) suggestions on the uses of literature review, this literature review is used to understand the current conceptual landscape, concentrating on the contemporary research on the development of computer-based, knowledge-driven decision support tools or systems in the AEC domain. Based on the scope of this dissertation which emphasized on the interrelationship between the

sustainable building, integrated, and BIM practices, DSS can be considered the information systems that fall within the scope of this research. DSS has considerable potential to be constructed as a system platform for implementing building information contained in BIM models associated with other industrial databases, analytical, optimization, and simulation models, and expert collective knowledge developed and used by AEC practitioners. DSS also has significant potential to be constructed as a system platform for multidisciplinary collaboration and communication. From the DSS development standpoint, it is important to take the diffusion of innovation principles into consideration throughout the development lifecycle, especially the five attributes of innovations. Because the development of DSS may be involved in the integration practice, it is important to consider the method, right, and litigation issues associated with the information sharing. Furthermore, in order to be highly compatible with the existing BIM and task-specific tools, it is important to consider the implementation of public domain data exchange formats such as the Industrial Foundation Classes (IFC) and IFC eXtensible Markup Language (IFCXML). While this state of DSS technology review highlights the possible trends and research gaps, focusing on the development of DSS within the domain of AEC practices as summarized in Table 2-2, the detailed information about DSS development will be extensively discussed in the next section of this chapter.

Table 2-2: Summary of literature review on the current state of decision support system technologies in the domain of architectural, engineering, and construction practices

Literature	Study	Research gap
Malkawi (2004a)	Use generic rules as virtual agents to detect thermal problems based on thermal simulation	Use the knowledge-based management subsystem to enhance the model management subsystem rather than use it as a primary component to provide possible solutions
Malkawi et al. (2005)	Develop a decision support evolution model that uses genetic algorithm as an evolution algorithm and Computational Fluid Dynamics (CFD) as an evolution mechanism, which allows design to be evaluated using CFD analysis automatically to maximize several thermal and ventilation criteria	Use the knowledge-based management subsystem to enhance the model management subsystem rather than use it as a primary component to provide possible solutions Use the genetic algorithm, which tends to be less trustable than the rule-based knowledge model as viewed by some DSS experts
Boddy et al. (2006)	Present a methodology to use a knowledge management environment, C-Sand, and a decision support tool, Advanced Decision Support (ADS)	Can be seen as a computer-based system that can be used for acquiring knowledge and creating knowledge representations Has considerable potential to be developed further in the form of Expert System (ES)
Haymaker and Suter (2006)	Propose a visual language and framework, called Narratives, for constructing and managing AEC information representations and dependencies between these representations	Can be seen as a communication-driven DSS that allows real human experts to be part of decision-making activities Could have some problems associating with using human experts, for example, they are rare and expensive
Flager et al. (2008)	Apply the Process Integration and Design Optimization (PIDO) technique used in the aerospace industry to an AEC case study: the Multidisciplinary Design and Optimization (MDO) of a classroom building for structural and energy performance.	Use a DSS generator, which requires a comprehensive skill set Use the genetic algorithm, which tends to be less trustable than the rule-based knowledge model as viewed by some DSS experts
Krygiel and Nies (2008)	Demonstrate a framework whereby BIM can be used as a center of sustainable building design activities	Use a BIM authoring tool as a DBMS Use a set of BIM support analysis tools, which require a comprehensive skill set Develop a decision-making work flow that allows different BIM support applications to blend in, which has considerable potential to be used as a basis for user-oriented DSS development
Onuma (2010)	Develop and market the Onuma Planning System (OPS), a combination of data-driven DSS and data mining system, built upon open-source technologies such as Apache and MySQL	Integrate a data mining system as an intelligent system element of the knowledge-based management subsystem, which demonstrates a potential use of knowledge-driven DSS in real world application Adopt Industry Foundation Classes (IFC) as an open standard for data exchange in real world application

2.2. Decision Support System Development

2.2.1. Decision Support System Development Approaches

Development of a DSS can be considered a complex process, which considerably differs from the development of other computer-based systems. This is because DSS require a symbiosis between the users and the system in order to be effective (Sprague and Carlson 1982). To make this symbiosis possible, users such as problem solvers and decision makers must understand the capabilities of DSS, and system developers must be able to integrate the DSS technologies into the decision making process. Development of a DSS also frequently involves diverse issues including, but not limited to, technical, organizational, and behavioral issues. In order to effectively develop a DSS, it is important to examine

several methodologies that have been proved to help professional DSS developers develop effective DSS.

From a holistic view, Sprague and Carlson (1982) introduced a conceptual framework for the development of decision support systems in their book, *Building Effective Decision Support Systems*. The framework aims to help DSS developers organize a complex subject, identify the relationships between the parts, and reveal the areas in which further developments will be required, as well as to be used as a communication mechanism for all the people who must work together in building DSS. The framework considers three important aspects of DSS development consisting of three levels of technology, all of which have been designated DSS, the roles of several key types of people in the building and use of DSS, and the development approach that is evolving for the creation of DSS.

As stated by Sprague and Carlson, there are three levels of hardware/software technologies involving in DSS development. The technologies are typically used by people who have different levels of technical capability, and differ in the nature and scope of tasks to which such people can be applied. The three technology levels are the specific DSS, DSS generator, and DSS tools. Specific DSS are the hardware/software systems that allow a group of decision makers or an individual decision maker to work on specific sets of related problems. Specific DSS are information system applications that actually interact with end users and accomplish the work. Specific DSS represent the first level of DSS technology. DSS generators are hardware and software packages whereby a particular package provides a set of capabilities to quickly and easily build specific DSS. Microsoft Excel is a good example of DSS generator used for developing Excel-based DSS applications. DSS generators represent the second level of DSS technology. DSS tools are hardware or software elements which facilitate the development of specific DSS or DSS generators. DSS tools represent the third and most fundamental level of DSS technology applied to the development of DSS.

In addition to the description of three technology levels involved in DSS development, Sprague and Carlson also described the relationships between the three technology levels. Specific DSS can be developed using either DSS generators or DSS tools. DSS generators can be developed using DSS tools and used for the development of specific DSS. DSS tools can be used directly for developing specific DSS as well as for developing DSS generators. Unlike conventional computer-based applications, DSS applications may need to be changed more often over their lifetime corresponding to the decision making task or problem, the user's approach to the problem, and the organizational environment in which the user faces the problem. As a result, it can be considered that the ability to accommodate change is a major value of DSS. Although DSS tools are widely used in the development of DSS applications, the use of DSS tools could be difficult when dealing with the constant change and flexibility. In consequence, DSS generators are widely used for the development of DSS applications that are frequently changed.

For the reason that DSS require the symbiosis between users and DSS to work effectively, it is important to consider people who are involved in building DSS as part of the framework. Referring to Sprague and

Carlson, the people who are normally involved in building DSS can be divided into five different roles in which a spectrum of these roles spread across the three technology levels. The five roles are the manager or user, intermediary, DSS builder, technical supporter, and toolsmith. The manager or user is the person who takes action on making decisions and is responsible for the consequences of the decisions. The intermediary is the person who provides immediate supports for the manager such as interacting with and making suggestions about the DSS. The DSS builder, also called as facilitator, is the person who works with the user or intermediary to assemble the necessary capabilities from the DSS generator to configure the specific DSS. The technical supporter is responsible for developing additional information system capabilities or components when they are needed as part of the generator. The toolsmith is responsible for developing new technology, new languages, and new hardware and software, as well as improving the efficiency of linkage between subsystems. Within the framework, a person may assume several roles or may work with a group of people to fulfill one role. Figure 2-5 represents the relationship between the three technology levels and the roles of people who are involved in building DSS.

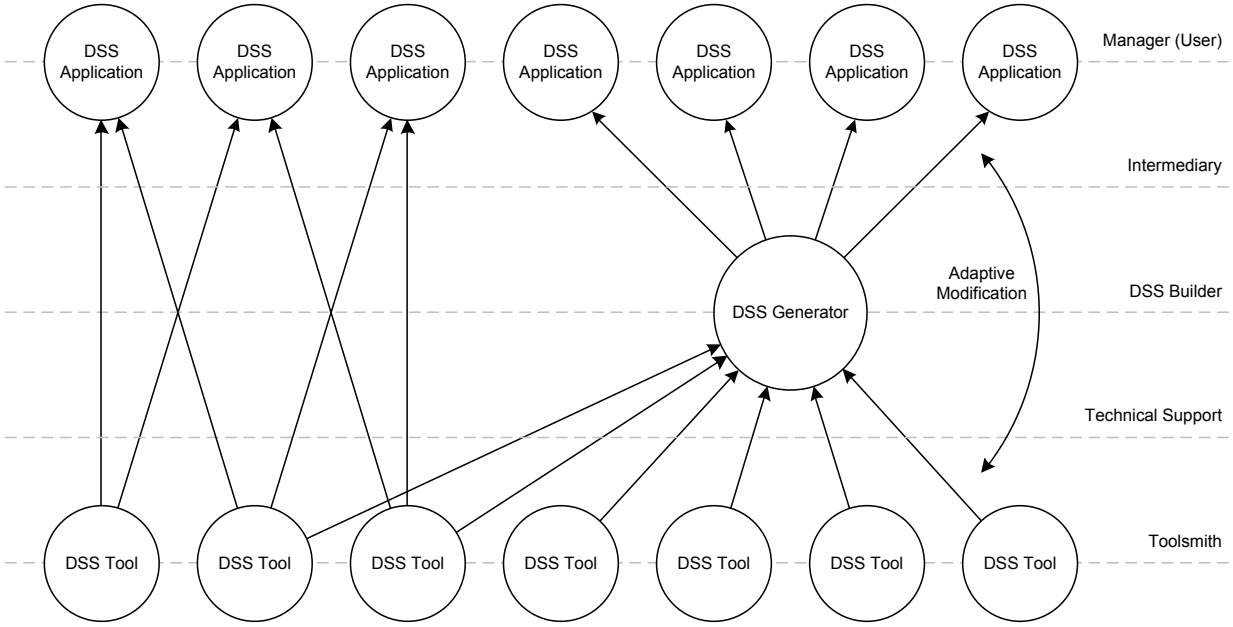


Figure 2-5: Three technology levels and the roles of people who involve in the development of DSS, adapted from Sprague and Carlson (1982)

According to Sprague and Carlson, there are certain issues that make the development approach used for developing DSS different from the approaches used for developing conventional computer-based systems. One issue is that there is no comprehensive theory of decision making. Another issue is the rapidity of change in the conditions that decision makers face. Furthermore, in almost all cases, it is very difficult for DSS designers or developers to determine the initial functional requirements of developing DSS at the beginning of the development project. As a result, DSS require a unique development approach to overcome these issues.

In their book, Sprague and Carlson proposed a unique approach for DSS development consisting of systems analysis, iterative design, and adaptive system. For the systems analysis, Sprague and Carlson suggested that DSS need to be independent of any imposed process because decision making processes can vary depending on different decision makers. In consequence, the user-oriented paradigm can be considered the most appropriate paradigm for articulating system performance requirements. For the iterative design, DSS frequently need to be built within short, rapid feedback from users to ensure that development moves in the right direction. It is also important to note that DSS are developed to permit change in a timely manner without any difficulty. As a result, the analysis-design-construction-implementation approach normally used for developing conventional computer-based systems is needed to be combined into a single step which is iteratively repeated as well as to be able to accommodate rapid feedback from users. For the adaptive system, DSS are adaptive systems by their nature. DSS consist of all three levels of technology in place, operated by the participants with different roles, with the technology adapting to change over their lifetime. Based on Simon (1980) cited by Sprague and Carlson (1982), an adaptive system adapts to changes of several kinds over three time horizons. First, the system allows search for answers within a relatively narrow scope in the short run. Second, the system learns by modifying its capabilities and activities in the intermediate time horizon. Third, the system evolves to accommodate much different behavior styles and capabilities in the long run.

While Sprague and Carlson's approach can be used as a basis for DSS development, it is also interesting to examine other major DSS development approaches that have been recently employed by today's DSS developers. Based on Power (2000) and Turban et al. (2007), there are three major approaches or methodologies involved in today's DSS development. The three approaches are the system development life cycle, prototyping, and user-based DSS development approaches. It is interesting to note that each particular approach seems to inherit certain characteristics from a distinct process model used in software engineering and likely to be described differently by experts in the DSS field.

The System Development Life Cycle (SDLC) is a systematic, sequential software development approach built upon the linear process model defined by Pressman (2010). The widely known waterfall process model, also called classic life cycle, is a great example of linear process models. The SDLC represents a process flow wherein development activities systematically flow from one to another. In each increment, the SDLC usually begins with well-defined system requirements and ends the process with new released software.

Prototyping is an evolutionary software development approach or methodology founded upon the evolutionary process model also defined by Pressman. The prototyping approach represents a development cycle that repeats itself until the developing systems sufficiently satisfy user needs. Unlike the SDLC, in each cycle, the prototyping approach normally starts with ill-defined system requirements and ends the process with a more comprehensive version of software.

The user-based, also referred to user-oriented, DSS development represents the specialized process model described by Pressman. Specialized process models undertake certain characteristics of one or more conventional process models including, but not limited to, linear, iterative, and evolutionary process models. Specialized process models appear to be employed when specialized or intently defined software engineering approach is selected. For the user-based DSS development, either the SDLC or prototyping approach can be applied depending on user's preference. Table 2-3 summarizes the key characteristics of particular DSS development approaches.

Table 2-3: Characteristics of three major DSS development Methodologies

Development Approach	Characteristic
System Development Life Cycle (SDLC)	System requirements must be solid since the beginning of DSS development, suitable for large-scale and/or complicated systems such as enterprise-wide DSS, widely used for the development of data-driven DSS, and inflexible to change
Prototyping	System requirements can be fuzzy, suitable for the development of model-driven and knowledge-driven DSS, user's participation is encouraged, and flexible to change
User-based DSS development	Appropriate only for small and simple DSS, both SDLC and prototyping can be used as a basis of DSS development depending on user's preference, and flexible to change

In order to effectively develop a DSS for a business organization, it is very important for a DSS design and development team to select an appropriate development methodology for a DSS development project. In his online book, *Decision Support Systems Hyperbook*, Power (2000) suggested a useful process developed to help a DSS design and development team select a proper approach for the development of a DSS. Figure 2-6 demonstrates the process recommended by Power. In a business organization, a DSS development project typically initiates by an executive sponsor who provides funding for the project. After a DSS development project is initiated, a DSS development team led by a DSS project manager begins to conduct the decision-oriented diagnosis. The diagnosis of current decision making and the specification of changes in decision processes are the activities that provide the key input to the design of the DSS. As stated by Power, a related diagnosis activity is to conduct a DSS audit. The DSS audit is an activity that helps the development team identify opportunities to redesign business processes and include new decision aids and DSS in business processes. Continued from the decision-oriented diagnosis, the DSS development team should prepare a feasibility study of the technical and economic prospects related to the development of a DSS. For the development of large-scale DSS, the feasibility study report preparation can be a lengthy and complicated process ranging from the scope of DSS to the benefits, risks, and mitigating factors. In smaller projects, the preparation can be much simpler and less comprehensive than in larger projects. Afterward, the outcomes of the feasibility study are then used to determine whether the organization should move to in-house or outsourced development. According to Power, in reality, purchasing packaged DSS applications and outsourcing the development and maintenance of DSS are quite versatile and usually less expensive to implement than in-house development.

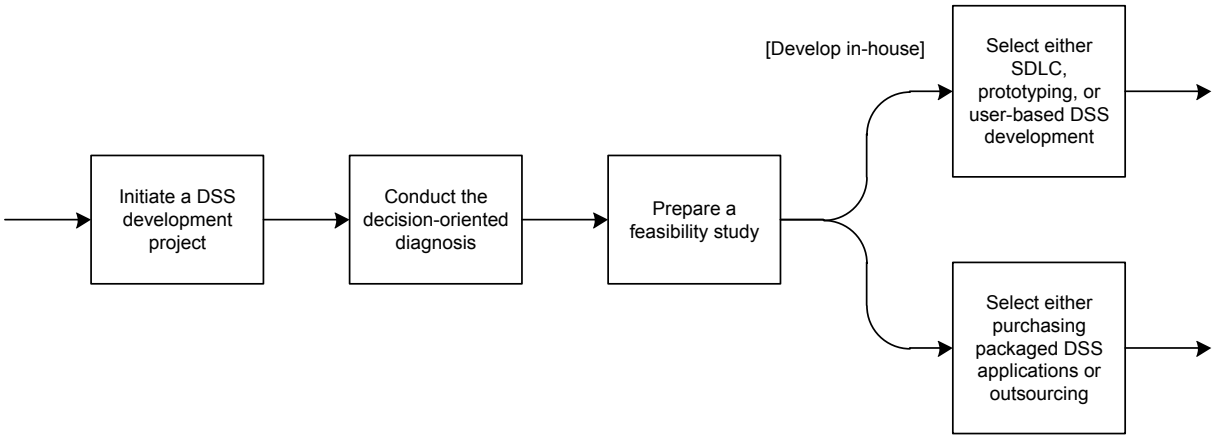


Figure 2-6: DSS development approach selection process, adapted from Power (2000)

Corresponding to Power (2000), Turban et al. (2007) also discussed whether an organization should develop DSS in-house or acquire DSS solutions through outsourcing. According to Turban et al., there are three development options for in-house DSS development including building from scratch, building from components, and integrating applications. Building DSS from scratch can be expensive and time consuming, but provides the most suitable solutions conforming to an organization's needs. This option should be considered only for developing specialized DSS whereby premade components are not available. The building from components option allows a development team to reuse in-house developed components as well as combine them with commercial packaged components, which make the in-house DSS development affordable and flexible enough to serve an organization's needs. Similar to the building from components option, the integrating applications option enables a development team to integrate DSS at the application level using a business-to-business approach between several business partners. In general, building DSS from components and integrating applications are more preferable than building DSS from scratch. Turban et al. suggested that insourcing can be considered a challenging task which requires specialized IT procedures and resources. Therefore, most organizations tend to acquire commercially-packaged DSS applications or totally outsource the development and maintenance of their DSS.

Based on Power and Turban et al., business organizations who can afford to develop DSS in-house usually prefer to take advantages from available DSS components or packaged DSS applications rather than build DSS from scratch. Additionally, small- and medium-sized organizations with small IT staffs and budgets are more than likely to outsource the development and maintenance of their DSS to outside contractors. Because a business organization may have a tendency to acquire DSS solutions through outsourcing, it is interesting to examine several methods normally used for acquiring DSS solutions from an organization's external sources.

As stated by Turban et al., DSS solutions can be acquired using methods including buying, leasing, software-as-a-service, and utility (on-demand or cloud) computing. The buying method is usually found in the in-house DSS development where by a development team purchases DSS components or packaged DSS applications from software venders. In the outsourced development, instead of buying, business organizations normally establish contracts with outside contractors, also referred to as Application Service Providers (ASP), for complete DSS solutions. The leasing method can be found in both in-house and outsourced developments whereby an organization can lease DSS products and/or services from software venders as well as from ASP. In recent years, software-as-a-service method is usually found when business organizations pay subscriber fees to use web-based DSS applications instead of buying or leasing DSS applications and installing such applications on their computers. This method has become increasingly attractive to business organizations due to the advancement in the Internet and World Wide Web technologies. The utility computing, also referred to as cloud computing, method provides unlimited computing power, storage capacity, and accessibility for both in-house and outsourced DSS development. This method allows business organizations to use their DSS applications on-demand in the pay-per-use manner.

Power's process for DSS development is not only useful for the development of DSS in business organizations, but also has considerable potential to help clarify certain aspects of the DSS development conducted as part of this dissertation. This dissertation can be considered a study in a series conducted within the College of Architecture and Urban Studies (CAUS), Virginia Tech. This series of research was initiated by the Center of High Performance Environments (CHPE) aimed to broaden a body of knowledge concerning energy efficiency, sustainability, systems integration, and integrating appropriate technology into indoor environments of the 21st century. In this research series, there are a number of studies emphasized on the processes of decision making involving in the lifecycle of high performance buildings. A Ph.D. dissertation, *A Decision-Making Framework for Vegetated Roofing System Selection*, conducted by Grant (2007) can be considered one example of such studies. Grant's study shares certain characteristics with the decision-oriented diagnosis described by Power. In her study, Grant identified decision making processes normally used for selecting a vegetated roofing system for a building project then proposed an alternative process that helps improve the quality of practitioner decisions concerning the vegetated roofing system selection. It is interesting to note that the decision-oriented diagnosis can be a body of research by its very nature. Thus, in order to confine the boundary of research, this dissertation intends to use the findings of former studies such as Grant's study as the key input to the design of the DSS.

In Power's process for DSS development approach selection, preparing a feasibility study can be considered a critical step to decide whether a business organization should move to in-house or outsourced DSS development. Nevertheless, it is important to note that the development of DSS conducted as part of this dissertation is not aimed to serve users or decision makers in a particular organization, but rather to serve diverse practitioners who tend to work in different organizations

including, but not limited to, architectural, engineering consultant, and contractor firms. Consequently, preparing a comprehensive feasibility study for industrial-wide users can become a challenging task and can be a body of research by its own right.

Fortunately, there are a few ongoing studies conducted in the area of Information Technology (IT) investment in AEC firms such as those conducted by Marsh and Flanagan (2000), Ekström and Björnsson (2003), and Love, Irani, and Edwards (2004). These studies represent the efforts to develop practical methods for evaluating the costs, benefits, and risks of IT investments especially in construction-oriented companies located in the United Kingdom, United States, and Australia. Apparently, these studies indicate that business organizations in the building industry are lagging behind business organizations in other industries in integrating IT into their business processes and not likely to invest considerably in IT. In addition, it can be considered that the majority of AEC firms are similar to small- and mid-size business organizations in other industries in which they tend to choose not to fully develop DSS inside their organizations. Although preparing a comprehensive feasibility study can be difficult because of the discussed barriers, a simple, but comprehensive enough, feasibility study should still be prepared as part of the DSS development in this dissertation.

Based on the situation in which AEC firms are unlikely to be capable of developing in-house DSS, it is appropriate to consider the development of DSS in the role of a software vender or an application service provider in the context of this dissertation. According to Power and Turban et al., whether the development of DSS will be conducted inside or outside a business organization, prototyping seems to be the most practical approach for the DSS development, especially in the early phases of DSS development projects. This is because the prototyping methodology is an evolutionary process whereby the changes on system requirements can be accommodated quickly and flexibly. According to Power, the prototyping methodology is highly effective to be used in the development of model-driven and knowledge-driven DSS as well as can be used in the development of other types of DSS to certain extent. As stated by Turban et al., if the prototyping methodology is conducted correctly, when the requirements and functions of DSS become stable, the development of DSS can then be migrated to the more formal development approach such as the SDLC approach to be better integrated with the enterprise-wide information systems.

In conclusion, based on Groat and Wang (2002) suggestions on the uses of literature review, this literature review is used to understand an idea's genetic roots, focusing on methodological approaches for DSS development. The understanding gained from the literature review is summarized in Table 2-4. The literature review indicates that the framework for DSS development suggested by Sprague and Carlson is considerably advantageous to be used as a basis of DSS development conducted as part of this dissertation. In addition to Sprague and Carlson's framework, Power (2000) suggested a process for DSS development approach selection which is also useful in the context of this study. This research can be considered a study in a series initiated by the CHPE. From the decision-oriented diagnosis standpoint,

this study aims to leverage the findings of former studies, also initiated by the CHPE, which were emphasized on the processes of decision making involving in the lifecycle of high performance buildings. From the feasibility study standpoint, a simple, but comprehensive enough, feasibility study report will be prepared as part of the DSS development. The objectives of the feasibility study include, but are not limited to, summarizing the background and definitions and determining the technical and financial feasibility of the DSS development. The feasibility study will be discussed further in Chapter 3: Methodology. Furthermore, from the development approach standpoint, this study will use the prototyping approach for the development of DSS in the role of a software vender or application service provider. The prototyping development approach will be discussed further in the following subsection of this chapter as well as in the following chapter.

Table 2-4: Summary of literature review on decision support system development approaches

Literature	Theory/methodological approach	Understanding/Implementation
Sprague and Carlson (1982)	Introduce a conceptual framework for the development of DSS Discuss three levels of DSS technology: specific DSS, DSS generator, and DSS tools Discuss five evolving roles in DSS: manager, intermediary, builder, technical supporter, and tool smith Discuss three techniques for DSS development: system analysis, interactive design, adaptive system	Intend to use the framework as a guideline for DSS development Consider developing the DSS as a specific DSS Intend to involve in the DSS project as a builder and/or technical supporter Intend to adopt the techniques for DSS development throughout the project lifecycle
Power (2000)	Introduce a process for selecting a DSS development approach Discuss three major in-house DSS development approaches: system development lifecycle (SDLC), rapid prototyping, and user-based DSS development Discuss two DSS acquisition options: purchasing and out sourcing	Adopt the DSS development approach selection process to help select an appropriate development approach View rapid prototyping as a promising process for developing DSS
Marsh and Flanagan (2000)	Discuss IT investment in the UK construction industry	Can be implied that UK companies may not be interested to develop DSS in house
Ekström and Björnsson (2003)	Discuss IT investment in the US construction industry	Can be implied that US companies may not be interested to develop DSS in house
Love, Irani, and Edwards (2004)	Discuss IT investment in the Australia construction industry	Can be implied that AUS companies may not be interested to develop DSS in house
Turban et al. (2007)	Discuss three development options for in-house DSS development: building from scratch, building from components, integrating applications Discuss five methods used in in-house DSS development: system development lifecycle (SDLC), rapid application development (RAD), prototyping, buying applications, leasing applications, and software-as-a-service (SaaS) Discuss three outsourcing development options: outsourcing, application service providers, utility (on-demand) computing	Aim to play the service provider role in developing DSS
Pressman (2010)	Discuss four prescriptive software engineering process models: waterfall, incremental, evolutionary, and concurrent.	Intend to use the evolutionary process model to guide DSS development

2.2.2. Prototyping: An Evolutionary Process Model for Decision Support System Development

Prototyping can be considered the most applicable and practical approach or methodology for the development of DSS conducted as part of this dissertation; thus, it should be examined in greater details.

As discussed earlier, the prototyping approach implemented in DSS development can be varied depending on different DSS experts and tends to inherit certain characteristics from the evolutionary process model used in software engineering. Therefore, in order to better understand the DSS development approach used for the DSS development in this dissertation, the evolutionary process model, some variations of the prototyping methodology, as well as benefits and shortcomings of the prototyping approach should be carefully studied.

Over the past decades, computer-based systems have become increasingly sophisticated and complex due to dramatic improvements in hardware performance, profound changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options (Pressman 2010). Sophistication and complexity not only produce impressive results when a system succeeds, but also bring massive problems for people who are involved in building sophisticated and complex computer-based systems. Consequently, in order to build and deliver quality software to the market, Pressman suggested that software engineers or developers should adapt a mature software process that is appropriate for the products and demands of the marketplace.

As stated by Pressman, a software process is a framework for the activities, actions, and tasks that are required to build high-quality software. In a process framework, there are two set of activities. The first set comprises framework activities that are applicable to all software projects, regardless of their domains, size, complexity, and degree of rigor with which software engineering is to be applied. Each activity in a particular framework composes of a set of actions whereby each action encompasses a set of tasks that produce a major work product. Each task set in a particular action aims to identify the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress. In a generic framework, there are at least five framework activities consisting of communication, planning, modeling, construction, and deployment.

The second set of activities in a process framework consists of umbrella or complementary activities that are applicable across the entire software process. Umbrella activities help a software development team manage and control progress, quality, change, and risk. In a generic framework, umbrella activities typically include project tracking and control, risk management, quality assurance, technical reviews, measurement, configuration management, reusability management, and work product preparation and production.

Referring to Pressman, within a process framework, the framework activities and their associated actions and tasks are organized by a process flow with respect to sequence and time as well as supported by the utility activities throughout the process. The framework activities organized by different process flows represent different software process models. Based on this principle, a generic evolutionary process model typically consists of the generic framework activities organized by an evolutionary process flow. A generic evolutionary process model is demonstrated in Figure 2-7. As shown in Figure 2-7, the

evolutionary process model executes five framework activities in a circular manner in which each circuit through the activities result in a more complete version of the software.

Based on the evolutionary process model described above, a prototyping process model consisting of prototyping activities ordered by an evolutionary process flow can be demonstrated in Figure 2-8. According to Pressman, the prototyping process begins with communication. Communication actions and their associated tasks such as conducting stakeholder meetings aim to define the overall objectives for the software, identify known requirements, and outline areas where further clarification is required. Subsequently, an iteration of prototyping is quickly planned, and then modeling in the form of a quick design takes place. A quick design mainly emphasizes on a representation of certain aspects of the software that will be visible to end users such as human interface layout and output display formats. Next, the prototyping process moves on to the prototype construction focusing on design implementation. After the prototype was constructed, it is deployed and evaluated by stakeholders, who provide feedback that is used for refining requirements to a greater extent. Afterward, the prototype is evolved into the next prototyping cycle to be fine-tuned to satisfy the needs of various stakeholders, while simultaneously enabling a developer team to understand more about software requirements.

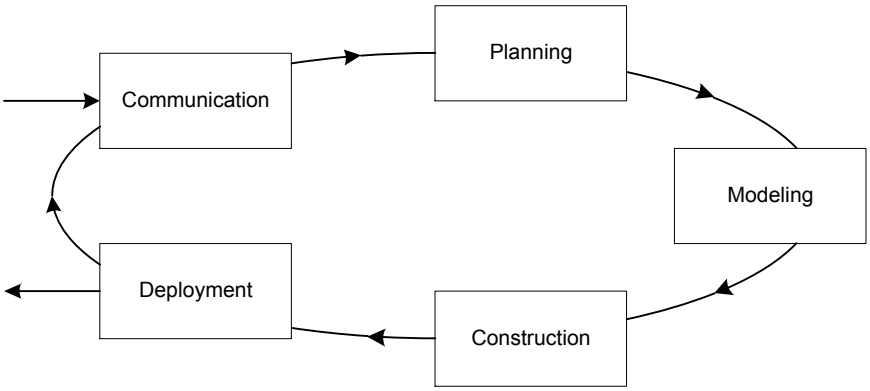


Figure 2-7: Evolutionary process model, adapted from Pressman (2010)

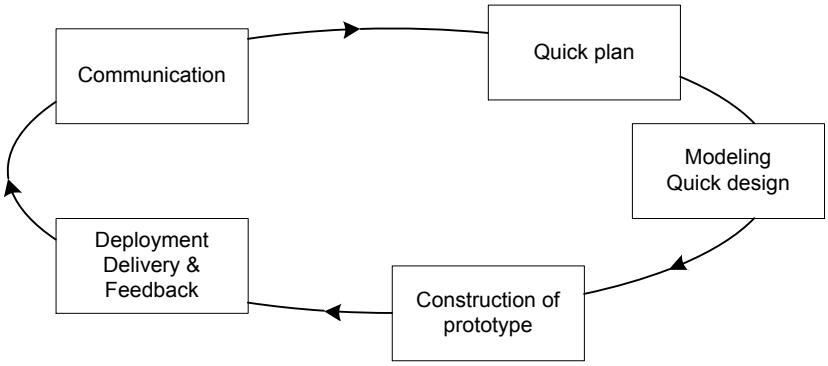


Figure 2-8: Prototyping process model, adapted from Pressman (2010)

In DSS development, the prototyping process model tends to be implemented as part of different DSS development processes described by experts in the DSS field. For example, prototyping can be implemented in rapid prototyping processes such as those described by Power (2000). Rapid prototyping processes used in DSS development typically consist of five steps. The first step is to identify user requirements, which is similar to the communication activity in the prototyping process model. The second step is to develop a first iteration DSS prototype, which represents the flow of activities from planning to deployment, delivery and feedback found in the prototyping process model. Next, the third step is to evolve and modify the next iteration DSS prototype, which repeats the development cycle with the more comprehensive understanding of user requirements. Subsequently, the fourth step is to test the DSS prototype and return to step 3 if needed. Afterward, the last step is to perform full-scale implementation using one of the formal development approaches such as the SDLC.

In addition to the rapid prototyping methodology described by Power, Turban et al. (2007) described another interesting example of how the prototyping process model can be used in DSS development. This example represents a prototyping development process used in the development of a real-world DSS for IMERYS Crop. The prototyping development process is illustrated in Figure 2-9. Figure 2-9 shows that the prototyping process starts with planning and analysis in which users, managers, and/or executive sponsors must be involved. One major goal of planning and analysis is to identify a sub-problem for which the initial DSS prototype is to be implemented. Subsequently, the analysis, design, and prototype implementation phases are iteratively performed until a small prototype is adequately developed. Afterward, the final implementation of this small part of the system goes on. At the same time, additional iterations take place in the loop of analysis-design-implementation-prototype while other subsystems or capabilities are included into the deployed system until a fairly stable, comprehensive system evolves. It is interesting to note that, in this example, the prototyping approach is used as part of a formal development approach called the Rapid Application Development (RAD). The RAD is normally used in the development of web-based DSS using Web programming tools.

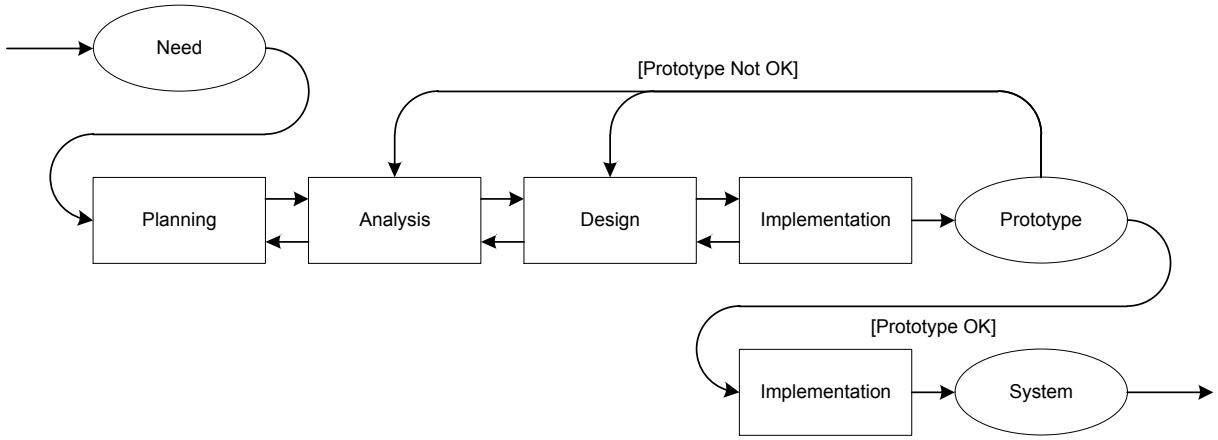


Figure 2-9: Prototyping development process, adapted from Turban et al. (2007)

Turban et al. also described a variation of their prototyping development process as shown in Figure 2-10. Figure 2-10 represents a throwaway prototyping development process. In this process, a particular prototype resulting from each analysis-design-implementation cycle evolves into the next development cycle. However, unlike the previous methodology, DSS prototypes resulting from evolutionary development cycles are usually developed as pilot tests and often thrown away rather than evolved into the full-scale design and implementation activities. Pilot tests are normally conducted in simpler development platforms or environments such as spreadsheet software until user requirements and the final system to be deployed are well-understood. When the pilot test is successful, the prototype is discarded and a preliminary design is then created for the real system. Later, the DSS can be completed through a selected formal development approach such as the SDLC or RAD using one of the programming languages.

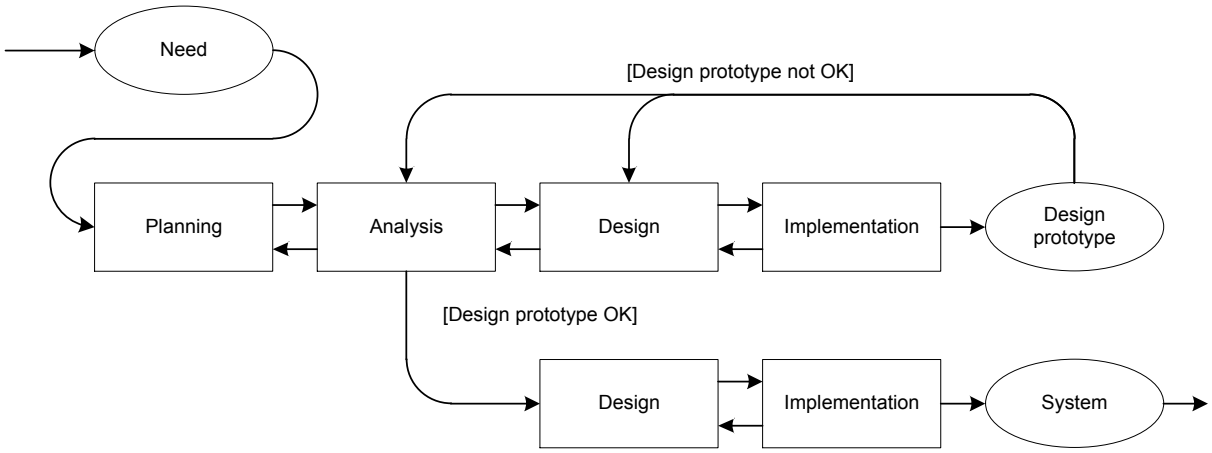


Figure 2-10: Throwaway prototyping development process, adapted from Turban et al. (2007)

According to Pressman, a prototyping paradigm can be considered the best approach to be used when software requirements are not well-defined. In a software project, requirements can be equivocal because a customer often defines a set of general requirements for software, but does not identify detailed requirements for software functions and features. Software requirements also can be ambiguous because a developer team may be unconfident about the efficiency of an algorithm, the adaptability of an operating system, or the form of interaction between a user and the developing system. In order to overcome these issues, prototyping is a software process that essentially serves as a mechanism for identifying software requirements. Prototyping helps assist developers and other stakeholders to better understand the software that is going to be built when requirements are fuzzy. Prototyping can be used not only as a stand-alone process model, but also, as a technique that can be implemented within the context of other process models.

In DSS development, Power suggested that prototyping appears to improve user-developer communication when compared with the more formal DSS development approaches such as the SDLC. Prototyping introduces deliberate flexibility and responsiveness into the process of DSS development.

Prototyping development processes can also accommodate change flexibly and effectively. Furthermore, the DSS developed using a prototyping development process tends to serve user needs better than a DSS developed through the formal development approaches. Besides Power, Turban et al. also suggested that prototyping allows a system to be quickly provided to users, even when it is not yet ready for formal use. Prototyping also requires short development and user-reaction time. Prototyping helps improve user understanding of the system as well as its capabilities and information needs. Furthermore, prototyping can be conducted at low cost.

Although the prototyping paradigm seems to be an appealing methodology for software engineering, prototyping have some limitations that should be taken into consideration. According to Pressman, software engineers often make implementation compromises in order to get a prototype working in a timely manner. Compromises may include using an inappropriate operating system or programming language simply because it is available and familiar and implementing an inefficient algorithm simply to demonstrate software capability. After some time, software engineers may feel comfortable with the compromises and overlook the reasons why such compromises were inappropriate. As a result, inappropriate compromises many times are included as an integral part of the system. Furthermore, due to the reason that it is haphazardly developed, prototype software apparently lacks overall quality or long-term maintainability. Prototype software often has to be rebuilt to be able to maintain high-levels of quality. In consequence, stakeholders who are unaware of this issue frequently feel disappointed, especially when they have to wait quite a while after seeing what appears to be a working version of software. Although prototyping can be problematic because of these reasons, it can still be an effective paradigm for software engineering. When using prototyping as a software engineering approach, it is important to acknowledge all stakeholders that the prototype is developed to serve as a mechanism for defining requirements and could be partially or totally discarded when the software development is geared toward quality.

In DSS development, prototyping can also be problematic for several reasons. According to Power, prototyping can extend the development schedule if it is improperly used. Many times, DSS users and developers tend to fiddle with a DSS and make changes that do not significantly improve the usability of the product. As a result, DSS development efforts must be well managed and controlled in order to ensure that the system being developed is useful and can be completed within project given deadlines. In addition to Power, Turban et al. also pointed that, when a prototyping approach is used, the gains acquired from particular development activities have considerable potential to be lost. These gains may include, but are limited to, an in-depth understanding of the information system's benefits and costs, a detailed description of the business's information needs, an easy-to-maintain information system design, a well-tested information system, and well-prepared users. Nevertheless, Turban et al. suggested that this problem could be solved by using a Computer-Aided Systems Engineering (CASE) tool to carry out consistency. Moreover, both Power and Turban et al. suggested that prototyping is not appropriate for the development of large scale data-driven DSS due to the scale, resource planning efforts, and levels of

experience of designers and implementers. Consequently, most of DSS prototypes are usually not fully attached to any large scale database.

While the use of a prototyping methodology in DSS development can be problematic, prototyping can be very effective if done carefully and with good design practices. A well-developed DSS using a prototyping approach is more than likely to allow more complex plans and more functionality to be added over time. In order to implement the prototyping process model effectively, Turban et al. suggested several critical success factors for prototyping. First, users and managers should be involved in every phase and iteration. Second, learning should be explicitly integrated into the design process. Third, prototyping should essentially bypass the formal information requirement definition in the formal development approaches such as the SDLC. Fourth, the interval between iterations should be as short as possible. Furthermore, the initial prototype must be low cost.

To summarize, this literature review is also used to understand an idea's genetic roots, focusing on the DSS prototyping process. The understanding gained from the literature review is summarized in Table 2-5. Based on its principles, prototyping can be considered the most appropriate methodology for the development of DSS in the context of this dissertation in certain ways. As mentioned earlier, this research aims to leverage the findings of former CHPE's initiated studies aimed to improve the quality of design processes and decisions involving in the lifecycle of high performance buildings. It has been recognized that the outcomes of such studies include, but are not limited to decision making matrixes, processes or frameworks, and algorithms can be seen as rich resources for the development of a DSS for holistic high performance or sustainable building design. However, such outcomes represent only a set of generic requirements for the DSS. In order to develop a DSS, the detailed requirements for DSS functions and features are also needed to be identified. Hence, prototyping can be significantly used as a mechanism for identifying the unknown requirements. In addition to the requirement issues, a prototype developed as part of this dissertation also represents part of an ongoing project aimed to develop a DSS for holistic building design. Thus, the prototyping process model such as described by Pressman is practical and applicable to be used as a basis of prototype development because it can be implemented as a standalone process as well as implemented as part of other DSS development approaches. In this study, the prototype development aims to emphasize on one design process or set of algorithms whereby the successful prototype can then evolve into the fully functional, high quality DSS being developed using a formal, high investment development approach. The prototyping methodology specifically used in this study will be discussed further in Chapter 3.

Table 2-5: Summary of literature review on the DSS prototyping process

Literature	Theory/methodological approach	Understanding/Implementation
Power (2000)	Discuss five rapid prototyping steps: identify user requirement, develop a first iteration DSS prototype, evolve and modify next iteration of DSS prototype, test DSS and return to the previous step if necessary, full-scale implementation Discuss advantages and disadvantages of rapid prototyping	Recognize the rapid prototyping process as well as its benefits and detriments
Turban et al. (2007)	Discuss the use of prototyping in the RAD Discuss the use of throwaway prototyping in the RAD Discuss prototyping collaboration, evaluation, and implantation Discuss advantages and disadvantages of prototyping	Understand more about how prototyping can be integrated into a more formal DSS development process such as RAD Recognize the issues of prototyping collaboration, evaluation, and implantation as well as its benefits and detriments
Pressman (2010)	Discuss two set of activities in software engineering process: framework activities and umbrella/commentary activities Discuss five framework activities: communication, planning, modeling, construction, and deployment Discuss eight umbrella activities: software project tracking control, risk management, software quality assurance, technical reviews, measurement, software configuration management, reusability management, and work product preparation and production Discuss prototyping paradigm in software engineering Discuss advantages and disadvantages of prototyping	Decide to implement the prototyping software engineering process to develop the DSS, because it can be used as a mechanism to define the requirements of system functions and features, which are not well understood Recognize the prototyping software engineering process activities

2.2.3. Technological Aspects of Decision Support System Development

Over the past decades, the dominant information technology platform in business organizations has been changing from mainframes and LAN-based, client-server systems to the Internet and World Wide Web (WWW) or Web technologies (Power 2000). This technological movement has influenced the paradigm change in DSS development. As stated by Power, the Internet and Web have become the platform of choice and new frontier for innovative DSS. The innovative web-based examples of all AIS SIGDSS's categories can be increasingly found and more innovative DSS of a particular category are gradually being developed. This emerging technology platform can possibly be an ideal platform for DSS development within the AEC domains, seeking for the information sustainability throughout the building lifecycle. The Internet and Web technologies enable AEC practitioners to exchange building related data, communicate, and work without limitations caused by computer hardware and software systems. As a result, in order to develop a DSS for AEC practitioners effectively, the technological background, data interoperability issues, hardware and software trends, advantages, and disadvantages of the Internet and Web technologies used in DSS development should be examined.

The Internet was named after a dominant network technology known as the Internetworking technology. The technology allows data to be transferred between computer networks through a collection of hardware and software. In general, the collection of hardware and software can be described by the ISO Open Systems Interconnect Reference Model (ISO/OSI RM), an international standard network model/architecture initiated by the International Organization for Standardization (ISO) and the

International Consultative Committee in Telephony and Telegraphy (CCITT). The ISO/OSI RM consists of seven network layers, which can be listed from the bottom layer whereby the data are electronically transmitted to the top layer that interfaces with a user as follows: Physical, Data Link, Network, Transport, Session, Presentation, and Application. The Internet is constructed upon the network architecture know as TCP/IP. The Transport Control Protocol (TCP) is a piece of software in the Transport layer responsible for ensuring the safe and reliable transfer of the data. The Internet Protocol (IP) is a piece of software in the Network layer responsible for setting the rules for data transfer over the network. The Internet can be seen as the infrastructure that enables data transfer and global networking.

While many people might think that the Internet and the Web are similar technology, they are actually not the same technology. The Internet allows any data such as email, fax, video, voice, and webpage data to be electronically transmitted over TCP/IP. For example, the Internet allows email data to be exchanged using the Post Office Protocol (POP) and Simple Mail Transfer Protocol (SMTP), the Application layer protocols used for receiving and sending email data over TCP/IP network protocol. In contrast, the World Wide Web or the Web allows only hypertext pages to be transmitted over TCP/IP. Hypertext is the Web content that can be viewed by a browser and transferred using the Hypertext Transfer Protocol (HTTP) in the ISO/OSI RM Application layer. In brief, the Web technology can be considered a subset of the Internet technology. In DSS development, a web-based DSS can be built upon the Web whereby its contents can be transferred using the HTTP over TCP/IP network architecture.

At the time of this study, Web 2.0 is the current state of the Web technology. Since 2003, Web 2.0 has noticeably shifted how people and business used the Web and developed web-based applications. According to Deitel and Deitel (2008), Web 2.0 provides new opportunities and connects people and content in unique ways. Web 2.0 embraces architecture of participation, which is a design that encourages user interaction and community contributions. This architecture has significantly changed the way software is developed. Open Source Software, Collective Intelligence, Rich Internet Applications (RIA), and Software as a Service (SaaS) are the emerging software development paradigms influenced by Web 2.0. Based on its capabilities, Web 2.0 creates new opportunities for DSS to be developed, distributed, and used as well as has become one of appealing platforms for DSS.

In Web 2.0, webpage contents are increasingly described by the eXtensible HyperText Markup Language (XHTML). According to W3Schools (2010), XHTML is a combination of the HyperText Markup Language (HTML) and the eXtensible Markup Language (XML). HTML is a markup language, a set of predefined markup tags, used for describing webpages. XML is a markup language designed to carry data and to be self-descriptive. Unlike HTML, XML was not designed to display data, and XML tags are not predefined. In the XHTML combination, HTML is responsible for displaying data whereas XML is responsible for describing data.

Because XML is a part of XHTML that deals with the exchange of data on the Web, it is interesting to discuss further about XML and some of its related technologies. XML is a strict markup language which

has to be syntactically correct, well-formed, and valid. According to Deitel and Deitel, an XML document can be processed using an XML parser which makes the document's data available to applications. If an XML parser can read an XML document, such a document is considered syntactically correct and well-formed. An XML document can reference a Document Type Definition (DTD) or an XML Schema that defines the proper structure of the XML document. An XML document can be validated using a validating parser against the reference DTD or an XML Schema. If the validating parser can read an XML document, such a document is considered valid. XML documents are highly portable, which can be viewed and modified using any text editor that supports ASCII/Unicode characters. In addition, XML documents can be formatted and manipulated using the eXtensible Stylesheet Language (XSL) which is a group of three technologies. First, XSL Formatting Objects (XSL-FO) is a vocabulary used for specifying the format of an XML document. Second, XML Path Language (XPath) is a string-based language of expressions used for locating structures and data in an XML document. Third, XSL Transformations (XSLT) is a technology used for transforming XML documents to other types of documents. XSL documents help programs determine how XML document data should be rendered.

As stated by Deitel and Deitel, XML can be considered a widely supported open technology for describing data and a standard format for data exchanged between applications over the Internet. XML enables document authors to create markup languages for describing any type of data in a way that both human beings and computers can understand. XML-based markup languages, called XML vocabularies, allow particular types of data to be described in standardized and structured ways. For example, MathML is an XML vocabulary created for describing mathematical formulas. Custom XML vocabularies have been increasingly developed by various industry groups in most major industries to allow business applications to communicate in unified languages. OpenGIS GO, gbXML, aecXML, IFCXML, and BLIS-XML are some examples of XML vocabularies that can be found in AEC practices in which many of BIM-based tools employed by AEC practitioners can generate XML documents based on these XML vocabularies. These vocabularies have considerable potential to be used for carrying and exchanging data between DSS and other applications used in AEC practices over the Internet.

The advancement in hardware, software, and communications technologies apparently extend the capabilities of the Internet and Web technologies. On the hardware side, the capabilities of computer hardware have changed dramatically from the mainframe computing age to the distributed computing age. Based on Moore's Law, the power of hardware doubles every two years, while the price remains essentially the same (Moore 1965). According to Deitel and Deitel, while hardware capabilities have increased immensely, hardware costs have decreased rapidly over the past decades. This phenomenon has also occurred in the communications field. With the high demand for communications bandwidth and tremendous competition between providers, communications bandwidth has increased quickly, while prices have decreased rapidly. The advancement in hardware and communications technologies has significantly influenced the wireless technology whereby portability has become a major focus for the computer industry. With the faster data-transfer speeds of wireless networks and the higher capabilities of

mobile computing devices, DSS users or decision makers can better operate DSS, share decision support information, and connect to the Internet wirelessly through available wireless networks using various types of portable devices such as laptop computers, handheld computers, and advanced mobile phones.

On the software side, the technology change from the procedural programming paradigm to the object-based programming paradigm has profoundly influenced how web-based applications were developed over the past decades. According to Deitel and Deitel, the object-oriented programming has become widely used since the 1980s. Before object technology, software applications were developed using procedural languages such as FORTRAN, Pascal, BASIC, and C. Programming languages as such focused on actions, normally represented by verbs, which quite unnaturally reflects the way people perceive the World. This issue made the programming of sophisticated and complex software highly difficult. In order to overcome this issue, the object-based programming using programming languages such as C++, Java, Visual Basic, and C# were developed and became widely used. In the object-oriented programming, a software object is an abstract representation of a real-world object such as a person. Such an object has its properties, for example, a person has weight. An object also can perform actions, for instance, a person can walk. Objects are instances of a class, also known as classification. Classes are types of related objects. For example, all persons belong to "person" class, although individual persons have different weights, heights, and hair colors. In brief, object technology is a packaging scheme for creating meaningful and reusable software units.

Object technology enables the development of sophisticated and complex software applications, including web-based or Web applications. Most of Web applications available today are more than likely being developed using object technology. Based on object technology, Web applications can be developed using client-side scripting languages such as JavaScript individually or together with server-side scripting languages such as PHP and Ruby or with high-level programming languages[9] such as Java, Visual Basic, and C#. In addition to object technology, Deitel and Deitel also suggested other new software technologies that have considerably influenced the development of Web applications including, but not limited to, agile software development, refactoring, design patterns, game programming, open source software, Linux, and Ruby on Rails. Some of these emerging technologies will be discussed further in this dissertation. Together with the advancement in hardware and communications technologies, new software technologies create new opportunities for Web based applications, including web-based DSS to have similar user experiences with desktop-based applications as well as to be operated through mobile computing devices via the Internet.

The Internet and Web technologies have brought considerable benefits to the development, operation, and maintenance of DSS. As stated by Power, the Internet and Web technologies have increased access

[9] The high-level programming languages are normally used as part of the Web application development frameworks, such as JavaServer Faces (JSF) and Microsoft's Active Server Pages .NET (ASP.NET).

to DSS. The technologies also help increase the use of a well-designed DSS in a business organization. The Internet and Web technologies provide a way to manage a company's knowledge repository and to bring knowledge resources into the decision making process. The technologies have reduced some of the problems caused by the competing "thick client" enterprise-wide DSS design that required special software to be installed on a manager's computer, which leads to the reduction of IT management, IT support, and end user training costs. In addition to Power, Turban et al. also suggested that the Internet and Web technologies provide easy access to a vast body of data. The technologies increase the availability of intelligent search tools that enable managers to find the needed information quickly and inexpensively. The technologies also provide a common, user-friendly Graphic User Interface (GUI) that is easy to learn, easy to use and readily available. Furthermore, the technologies provide a high level of ability to collaborate with remote partners.

While the Internet and Web technologies seem to be a promising platform for DSS, there are some issues associated with the technologies that should be discussed. Referring to Power, one problem that can be found in web-based DSS is that users may have unrealistic expectations, especially about the amount of information in which the Web can provide. In addition, there may be technical implementation problems, especially the peak demand and load problems. It is also quite expensive to train decision support content providers and to provide such providers with sufficient tools and technical assistance. The Internet and Web technologies can raise additional security concerns. Furthermore, using the Web for decision support could cause the accumulation of outdated materials, especially management reports and documents. However, these problems can be lessened by applying appropriate software engineering processes.

In summary, this literature review is used to understand an idea’s genetic roots, focusing on the technological aspects of DSS development. The understanding gained from the literature review is summarized in Table 2-6. The literature review shows that the Internet and Web technologies have considerable potential to be implemented as a platform for the prototype DSS developed as part of this dissertation. The technologies provide tremendous opportunities for the DSS to be developed, operated, and maintained with less technological barriers and at affordable costs. The technologies make the DSS freely accessible, always available, and allow AEC practitioners to work collaboratively with other project stakeholders without the limitation of computer operating systems and software applications as well as from remote locations around the globe. The DSS also has considerable potential to benefit from the existing XML vocabularies created for carrying and exchanging data among AEC applications, especially the BIM-based tools. Furthermore, the advancement in hardware, software, and communications technologies allows web-based applications to have similar user experiences with the desktop-based applications and to be used wirelessly from mobile computing devices through the Internet. Although the technologies have some associated problems that should be taken into consideration, such problems can be reduced by applying proper software engineering processes. As a result, in this study, the prototype DSS will be developed as a web-based DSS application on the Internet and Web platform.

Table 2-6: Summary of literature review on the technological aspects of DSS development

Literature	Theory/methodological approach	Understanding/Implementation
Moore (1965)	Discuss the inverse relationship between hardware capabilities and costs	Recognize that, while hardware capacities have increased immensely, hardware costs have decreased rapidly over time
ISO and CCITT	Develop and standardize the ISO Open Systems Interconnect Reference Model (ISO/OSI RM) Define ISO/OSI RM seven layers: physical data link, network, transport, session, presentation, and application	Recognize the TCP/IP protocol, which is the network architecture that the Internet is built upon
Power (2000)	Suggest that the Internet and Web have become the platform of choice and new frontier for innovative DSS	Intend to use the Internet and Web as a platform to develop the DSS
Deitel and Deitel (2008)	Discuss Web 2.0 and hardware technologies and trends emphasizing the mobile, wireless technology Discuss Web 2.0 and software technologies and trends emphasizing the object technology	Aware that the DSS may be used by mobile, wireless devices such as tablet computers and smartphones
Eastman et al. (2008)	Summarize XML vocabularies used in the AEC industry as follows: OpenGIS GO, gbXML, aecXML, IFCXML, and BLIS-XML	Recognize the XML vocabularies used in the AEC industry, which have considerable potential to be used for exchanging data between BIM authoring tools and DSS
W3Schools (2010)	Provide the definition of HTML, XML, and XHTML	Understand the definition of HTML, XML, and XHTML

2.2.4. Decision Support System Architecture and Components

Software architecture is a system that is defined in terms of a collection of components and interactions among those components whereby a particular system can also engage as an element in a larger system (Shaw and Garlan 1996). This abstract definition of software architecture likewise can be used as a basis for describing the architecture of decision support systems. Referring to Sprague and Carlson (1982) and Turban et al. (2007), the components of DSS typically include the data management, model management, and user interface subsystems and optionally include knowledge-based management subsystems. In the Internet and Web environment, these system components interact through the Internet and Web infrastructure, which forms the web-based DSS architecture.

Data Management Subsystem

As defined by Turban et al., the data management subsystem includes a database that contains relevant data and is managed by the Database Management Software (DBMS). According to Turban et al., the data management subsystem is typically composed of a DSS database, data directory, query facility, and DBMS. The structure of the data management subsystem is demonstrated in Figure 2-11. The data management subsystem can be interconnected with the corporate data warehouse which is a repository for pertinent decision-making data. Because of the advancement in the Internet and Web technologies, data are typically stored or accessed via a database Web server.

A database is a collection of interrelated data, organized to meet the needs and structure of an organization that can be used by more than one person for more than one application (Turban et al. 2007). In small-scale DSS, data can be directly entered into models and sometimes can be extracted from larger databases. In large-scale DSS, data can be further extracted from fully integrated, multiple-source DSS databases. The data in a DSS database can be extracted from three sources including

internal data, external data, and personal data sources using the operation or process called extraction. Extraction essentially consists of the importing of files, summarization, standardization filtration, and condensation of data, which is also known as Extraction, Transformation, and Load (ETL) process.

In a database, the data definitions are contained in the data directory which can be seen as a catalog of all the data. The directory's main function is to answer questions about the availability of data items, their source, and their exact meaning. The directory also supports the addition of new entries, deletion of entries, and retrieval of information about specific objects.

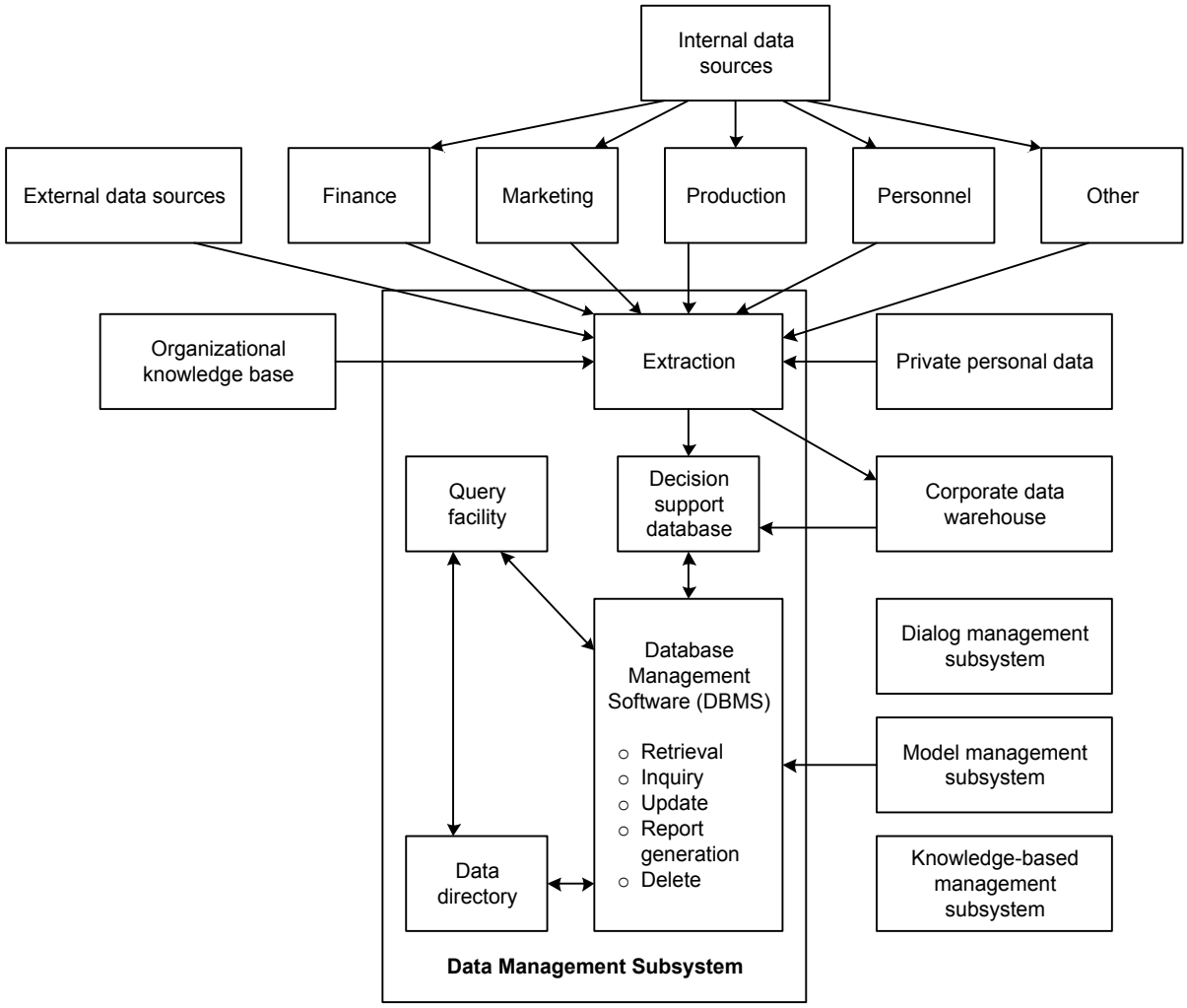


Figure 2-11: Data management subsystem structure and components, adapted from Turban et al. (2007)

In order to build and use DSS, it is necessary to access, manipulate, and query data from a database. The query facility performs these tasks by accepting requests for data from other DSS components, determining how the requests can be filled, formulating the detailed requests, and returning the results to the issuer of the requests. The query facility includes a special query language such as the Structured

Query Language (SQL), which can also be directly utilized by the programing languages such as Java, Visual Basic, and C#.

The data management subsystem can be managed through the DBMS. The DBMS is capable of capturing or extracting data for inclusion in a DSS database; updating data records and files; interrelating data from different sources; retrieving data from the database for queries and reports; and performing complex data manipulation tasks based on queries. The DBMS is also responsible for providing comprehensive data security, handling personal and unofficial data, tracking data use within the DSS, and managing data through a data dictionary. DBMS can be either purchased from major venders such as IBM, Microsoft, and Oracle or acquired from low cost, open-source providers such as MySQL and Apache Software Foundation.

Referring to Turban et al., there are key database and database management system issues that should be taken into consideration. The first issue is the quality of data. Turban et al. suggested that the United States firms routinely lose over $600 billion per year because of poor data quality. Within the scope of the data management subsystem, the data quality problems probably can be lessened by utilizing data quality cleanup tools. The second issue is the data integration. In order to support decision makers' decisions, most of the time data and information tend to be gathered from various sources. For data integration, Enterprise Information Integration (EII) products such as DB2 Information Integrator and XML standardization can help improve the quality of data integration. Another issue is the scalability. The scalability problem is usually found in large databases in which their processing times and storage space grow dramatically due to the increasing size of data to be stored and accessed. This issue probably can be resolved by the improvement of the DBMS in the areas of data models, access methods, query processing algorithms, concurrency control, recovery, query languages, and user interfaces to the DBMS. The last issue is the data security. The data security issue can be reduced by utilizing tools developed for monitoring database activity and providing audit trials.

Model Management Subsystem

The model management subsystem is a software package that includes quantitative models, appropriate software management, and modeling languages (Turban et al. 2007). The model management subsystem is managed by the Model Base Management Software (MBMS) and can be connected to corporate or external storage of models. Model solution methods and management systems can be implemented in Web development systems to run on application servers. Based on Turban et al., the model management subsystem can be composed of the model base, model directory, model execution, integration, and command processor, and MBMS. The structure of the model management subsystem is demonstrated in Figure 2-12.

As stated by Turban et al., a model base contains routines and special statistical, financial, forecasting, management science, and other quantitative models that provide the analysis capabilities in a DSS. In a DSS, the number of models can be varied depending on different decision problems. In general, the

models in the model base can be categorized into four categories including strategic, tactical, operational, and analytical models. Strategic models are used to support top managers' strategic planning responsibilities. Tactical models are used to assist middle managers in allocating and controlling the organization's resources. Operational models are used to support the day-to-day working activities of the organization. Analytical models are used to perform analysis on data. The models in the model base alternatively can also be classified by functional areas such as product control models or by disciplines such as management science allocation models.

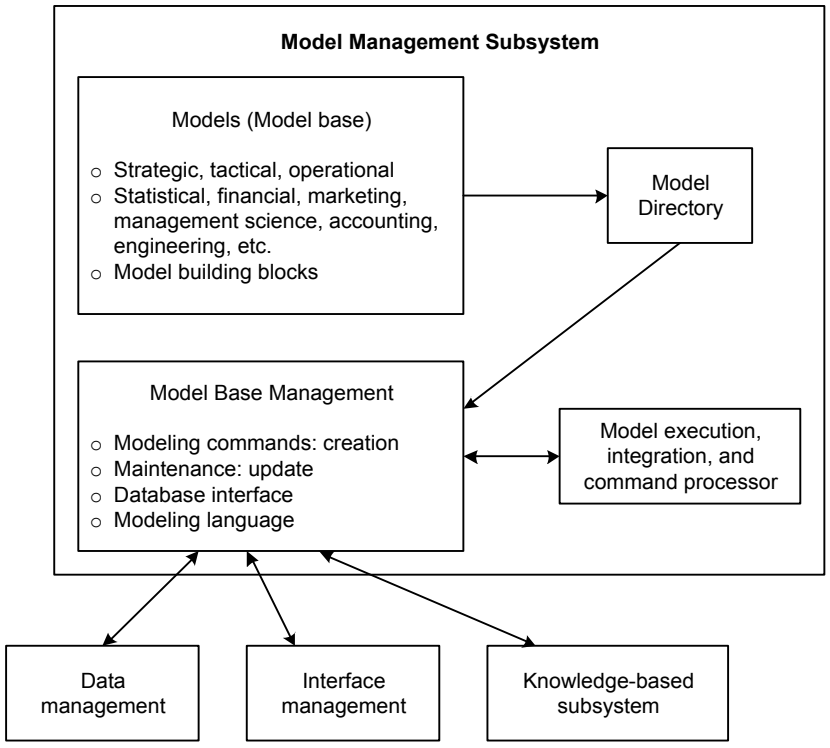


Figure 2-12: Model management subsystem structure and components, adapted from Turban et al. (2007)

In addition to models, the model base can also contain model building blocks and routines. Model building blocks and routines are software elements used for building computerized models. A particular model building block or routine can be used as a standalone model or as part of larger models. Examples of model building blocks and routines may include a random number generator routine, a curve- or line-fitting routine, and a present value routine.

At a higher level than building blocks and routines, Turban et al. also suggested that it is important to take into consideration the different types of models and solution methods needed in the DSS. DSS development systems frequently provide one or more modeling components for building DSS. In general, there are three types of components provided by the DSS development systems. First, the deterministic modeling components include those for linear programming, dynamic programming, modeling languages, and so on. Second, the probabilistic modeling components include those for forecasting, Markov modeling, queuing, simulation, and so on. Third, the modeling components that can be considered either

deterministic or probabilistic include those for vehicle routing and scheduling, and product planning, and so on; which each of them contains a number of modeling components.

Based on the nature of DSS in which they normally have to deal with semi-structured or unstructured problems, most of the time the models contained in a model base are needed to be customized. In consequence, besides model building blocks and routines and modeling components, it is also important to consider modeling tools and languages used for customizing models. Some examples of widely used modeling tools and languages are .NET Framework languages, C++, and Java. In addition to these tools and languages, some DSS development systems may have their own proprietary modeling tools and languages.

Quite similar to the database directory, the model directory can be seen as a catalog of all the models and other software in the model base. According to Turban et al., the main function of the model directory is to answer questions about the availability and capability of the models based on model definitions contained in the directory.

In addition to the model base and model directory, it is important to discuss the model execution and integration which are the activities controlled by model management. As stated by Turban et al., model execution is the process of controlling the actual running of the model. Also stated by Turban et al., model integration involves combining the operations of several models when needed or integrating the DSS with other applications. In order to control these activities, a model command processor is used to accept and interpret modeling instructions from the user interface component and route them to the MBMS, model execution, or integration functions.

The model management subsystem is managed by the MBMS. The MBMS allows users to create models from scratch, from existing models, or from the building blocks as well as to manipulate the created models. The MBMS is capable of storing, retrieving, and managing a wide variety of different types of models in a logical and integrated manner. The MBMS is also capable of accessing and integrating the model building blocks. The MBMS enables users to use multiple models to support problem solving as well as to interrelate models with appropriate linkages with the database and integrate them within the DSS. The MBMS has capabilities to catalog and display the directory of models for use by several individuals in the organization. Furthermore, the MBMS is also responsible for managing and maintaining the model base with management functions and tracking model data and application use.

User Interface Management Subsystem

The user interface subsystem is part of DSS that directly interact with the user or decision maker. Based on Sprague and Carlson's (1982) decision making system, the user who communicates with and commands the DSS through the user interface is considered part of the system. According to Turban et al., it can be considered that some of the unique contributions of DSS are derived from the intensive interaction between the computer and the user. The user interface management subsystem is managed

by the Dialog Generation and Management Software (DGMS), also known as the User Interface Management System (UIMS). Similar to the data management and model management subsystems, the user interface subsystem also benefits from the advancement in the Internet and Web technologies. The Web provides a set of uniform user interface objects that is easy to learn to use and readily available.

According to Turban et al, the user interface subsystem is composed of user interfaces, DGMS, and natural language processor. The schematic structure of a user interface process is demonstrated in Figure 2-13. User interfaces allow the user to interact with DSS through action and display languages processed by the DGMS. The DGMS enables the user to interact with the model management and data management subsystems. In advanced systems, the user interface component includes a natural language processor or provides standard objects such as pull-down menus and buttons through a Graphic User Interface (GUI).

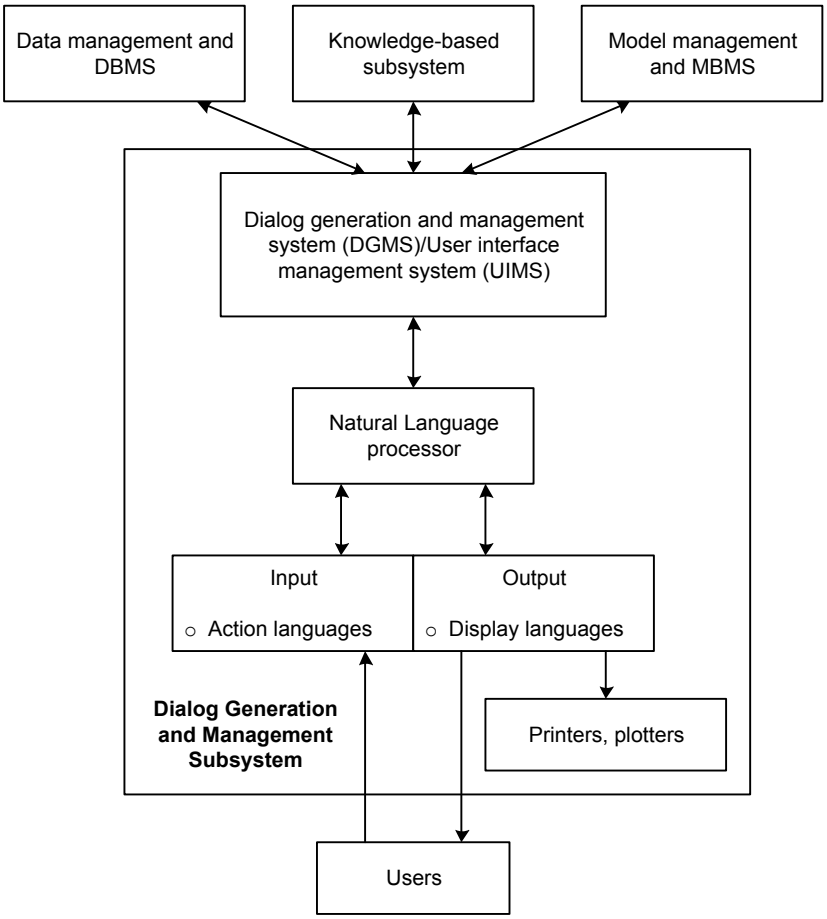


Figure 2-13: Dialog generation and management subsystem structure and components, adapted from Turban et al. (2007)

Knowledge-Based Management Subsystem

In addition to the major three DSS components, the knowledge-based management subsystem is a DSS component that provides intelligence to augment the decision maker's own (Turban et al. 2007). This

subsystem can support any of the other subsystems or perform as an independent component. The knowledge-based management subsystem can be interconnected with the organization's knowledge repository, which is also known as the organization knowledge base. Due to the advancement in the Internet and Web technologies, knowledge is increasingly provided via Web servers. According to Turban et al., many artificial intelligence methods have been implemented in Web development systems such as Java and .NET Framework and are easy to integrate into the other DSS components.

The knowledge component consists of one or more intelligent systems. Knowledge components may be provided by expert system, neural networks, intelligent agents, fuzzy logic, case-based reasoning systems, and etc. Similar to database and model management software, Knowledge-Based Management Software (KBMS) provides the necessary execution and integration of the intelligent system.

Web-Based Decision Support System Architecture and Development Tools

Based on DSS components suggested by Sprague and Carlson and Turban et al., a schematic view of DSS consisting of data management, model management, user interface, and knowledge-based management subsystems can be illustrated as shown in Figure 2-14. In addition to the interactions between DSS components and between DSS and their users described earlier, DSS can also interconnect with other computer-based systems as well as the Internet, intranets, and extranets. It is interesting to note that this schematic architecture of DSS can also be applied to web-based DSS except that the DSS components in web-based DSS may be distributed throughout the Internet and Web. In web-based DSS, the DSS components interact via the Internet and Web infrastructure.

According to Power (2000), most web-based DSS are built using a three- or four-tier[10] architecture. A basic web-based DSS architecture consisting of data management, model management, and user interface management subsystems is demonstrated in Figure 2-15. Based on this architecture, a user or decision maker sends a request through the Hypertext Transfer Protocol (HTTP) to a Web server using a Web browser. Next, the Web server processes the request using a program or script. The script then may implement or link to a model, process database request, or format a document. Afterward, the results are returned to the user's web browser for display via the Web browser. As stated by Power, the application code usually sits on a remote server and the user interface is presented at the client's WWW browser. Web applications are designed to allow any authorized user to interact with them through a WWW browser and an Internet connection.

[10] A tier is a partitioning of functionality that may be allocated to a separate physical machine (Bass, Clements, and Kazman 2003).

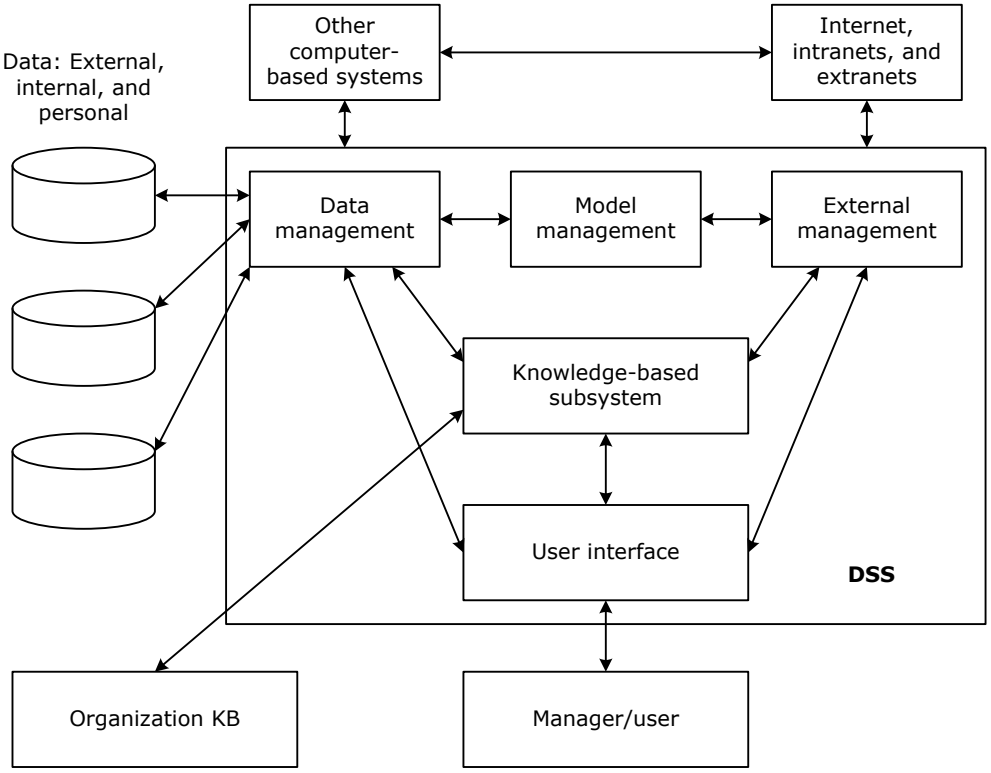


Figure 2-14: Schematic view of DSS, adapted from Turban et al. (2007)

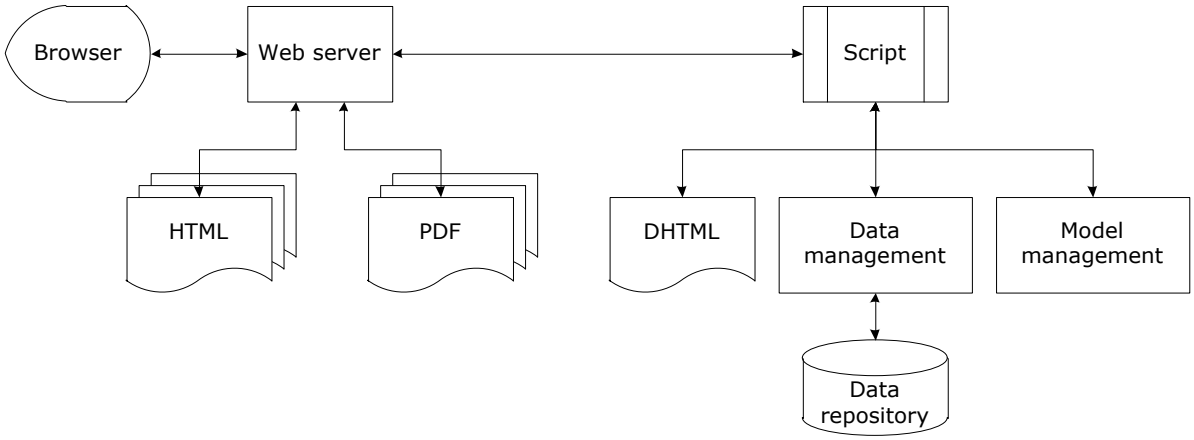


Figure 2-15: Web-based DSS architecture, adapted from Power (2000)

In addition to the basic architecture of web-based DSS demonstrated in Figure 2-15, Power also suggested some widely used tools for building web-based DSS. The tools include Hypertext Markup Language (HTML), eXtensible Markup Language (XML), Common Gateway Interface (CGI), JavaScript code in HTML pages, Java applets, and ActiveX components.

HTML and XML are markup languages used over the Web. HTML is designed to specify the logical organization of a document with hypertext extension for hypertext links and user interaction. For DSS,

HTML can be used for receiving input and showing output from a decision aid programmed in a programming language such as Java or JavaScript. XML is a general syntax for describing hierarchical data. XML is applicable to a wide range of DSS applications, including applications with databases, web documents, and searching (Power 2000). In combination, HTML is responsible for displaying decision support data whereas XML is responsible for describing such data.

CGI, JavaScript, Java applets, and ActiveX components increase the interactive capabilities of web-based DSS. CGI applications are server-executed programs used for creating Dynamic HTML (DHTML) documents, for taking values from Web forms, and for providing web-based interface to other applications. CGI programs provide the back-end processing for many web-based Decision Aids and DSS. JavaScript is a programming language that is highly integrated with web browser objects. JavaScript is downloaded as part of an HTML page and the Web browser processes it after it is received. JavaScript programs consist of functions that are called as a result of Web browser events. Java is a high-level general-purpose programming language. Because complied Java code is architecture-neutral, Java applets/applications are ideal for diverse operating system environments like the Internet. Java provides a powerful addition to the DSS development tools for programmers. Similar to Java applets, ActiveX controls are reusable software components developed by Microsoft. These controls can be used to quickly add specialized functionalities to Web sites, desktop applications, and development tools. Related to ActiveX, VBScript enables one to embed interactive elements into HTML documents.

According to Power, there are some issues that should be taken into consideration when developing DSS on the Internet and Web infrastructure. The issues include peak load problems, the stateless nature of the Web, and the rapid change of Web technologies. For the peak load problems, it is important for developers to consider the scalability of the hardware and software as well as future expansion planning. The stateless nature of the Web can lead to securities issues when business organizations would like to make sensitive internal data accessible to users. User authorization and authentication have become challenging in the Web environment due to the large number of potential users. Web technologies have been rapidly changing overtime. Therefore, it is important for a development team to keep up with new developments as they occur.

In conclusion, this literature review is used to understand an idea's genetic roots, concentrating on the DSS architecture and components. The understanding gained from the literature review can be summarized as shown in Table 2-7. In general, the understanding of software architecture helps increase the quality of software development as well as the effectiveness of software maintenance (Shaw and Garlan 1996). For the development of DSS, DSS architecture helps describe high-level relationships between DSS subsystems including data management, model management, user interface, and knowledge-based management subsystems. DSS architecture also helps determine high-level relationships between DSS and related computer-based systems. The detailed understanding of DSS architecture can also help a development team to make principled choice among design alternatives. An

architectural system representation is also important for the analysis and description of the high-level properties of a complex DSS. For system maintenance, it can be considered that most of DSS development environments tend to provide reusable DSS building components, especially in the web-based environment. Based on their nature, DSS can be seen as living mechanisms which are needed to be changed very often over their lifetime. Such reusable components have significant potential to help accommodate those required changes. In the context of this study, the DSS architectural representations demonstrated in Figure 2-14 and 2-15 will be used as a basis for the architectural design of the prototype web-based DSS.

Table 2-7: Summary of literature review on the DSS architecture and components

Literature	Theory/methodological approach	Understanding/Implementation
Shaw and Garlan (1996)	Provide the abstract definition of software architecture	Recognize that software architecture is a collection of components and interactions among them in which a particular system can also engage as an element in larger system
Sprague and Carlson (1982)	Discuss basic DSS architecture, which comprises the following components: data management, models management, and user interface management subsystems	Recognize DSS components and their relationships
Power (2000)	Discuss an example of multi-tier web-based DSS architecture Introduce web-based DSS development tools: HTML, XML, CGI, JavaScript, Java applets, and ActiveX components	Aim to implement the multi-tier web-based DSS architecture as a basis for DSS development
Turban et al. (2007)	Discuss generic DSS architecture, which comprises the following components: data management, models management, user interface management and knowledge-based management subsystems Elaborate on the data management subsystem composed of DBMS, DSS database, data directory, and query facility Elaborate on the models management subsystem composed of MBMS, model base, modeling language, model directory, and model execution, integration, and command processor Elaborate on the user interface management subsystem composed of DGMS and natural language processor or standard objects through GUI Elaborate on the knowledge-based management subsystem composed of KBMS and intelligent system	Intend to rearrange the generic DSS architecture to match the multi-tier web-based DSS architecture Recognize basic elements of each DSS components

2.3. Decision Support System Capability for Vegetated Roofing System Selection

As discussed in the previous section, this study can be seen as research in a series initiated by the Center for High Performance Environment (CHPE), Virginia Tech. This study aims to extend the findings from one of the preceding CHPE's initiated studies emphasized on improving the quality of design processes and decisions involved in the lifecycle of high performance buildings. Among a few completed studies in the series, the development of a prototype DSS aims to adapt a decision-making framework developed by Elizabeth J. Grant (2007) from her dissertation paper, *A Decision-Making Framework for Vegetated Roofing System Selection*. The framework was developed to help designers determine the

most feasible green roof system for a particular project based on a summation of total importance of the advantages represented by each design alternative. This framework has considerable potential to be transformed into a computer-based DSS, especially in the category of Expert Systems (ES). This section discusses Grant's decision-making framework and the development of ES in detail.

The development of a prototype DSS in this dissertation adapts the decision-making framework for vegetated roofing system selection based on certain reasons. First, there is a body of evidence indicates that building project stakeholders have shown a growing interest in adopting green roof technology as part of their projects, which leads to an increasing demand for vegetated roofing system design support tools. According to Grant, green roof systems offer unique benefits which not only serve to reduce the negative impact of buildings on the ecosystem, but also help improve the quality of both interior and exterior living space. In North America, an increasing number of practitioners are being requested to include green roof systems in their project designs. However, effective methods for selecting an appropriate green roof system for a particular project are not well defined. As a result, a decision support system is needed to assist these practitioners in comparing the efficacy of various green roofs in the context of specific projects. Second, the framework applied a sound decision-making system to help decision makers make complex decisions. The framework applied the Choosing By Advantages (CBA) decision-making process which has been widely used by well-known government agencies and business organizations such as the U.S. Forest Service. CBA can be used for assisting decision makers in a wide-range of decisions, ranging from the simplest to the most complex decisions. Lastly, the framework was tested for acceptability against the process employed by experts. As part of her dissertation, Grant used the interview as a research method to capture the green roof selection processes as conceived by several designers and compare them with her developed framework. The framework's functioning was compared with the designers' implicit selection processes in the three green roof design projects. As stated by Grant, the framework proved sufficiently flexible to accommodate the simplicity or complexity of the actual decision situation in each case.

Before examining the mechanism of the decision-making framework for vegetated roofing system selection, it is interesting to briefly look at the typical assembly and generic classification of green roof systems. According to Grant, conventional green roofs are composed of a series of layers which may include vegetation, growing medium, filter, drainage, and protective layers as shown in Figure 2-16. Occasionally, green roof layers may be combined, omitted, or articulated with a series of sub components. Nevertheless, the functions represented by each of these layers must be satisfied by some element in the green roof assembly.

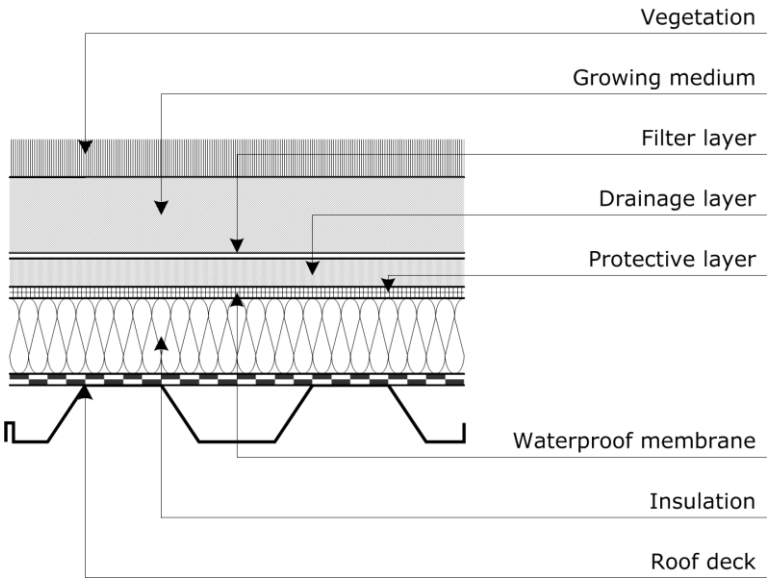


Figure 2-16: Typical green roof assembly, adapted from Grant (2007)

Contemporary vegetated roofing is frequently divided into two generic categories: extensive and intensive, based on virtue of the depth of its growing medium. Extensive green roofs, also called low-profile roofs, eco-roofs, or performance roofs, are referred to those green roofs with medium depths less than 6 inches (150 mm). Intensive green roofs, also called high-profile roofs or roof gardens, are referred to those green roofs with medium depths exceeding 6 inches (150 mm). Based on the German FLL's[11] document, *The Guideline for the Planning, Execution, and Upkeep of Green-Roof Sites*, published in 2002 cited by Grant, green roofs can also be divided into eight generic types based on substrate depth and plant type. These green roof types represent a sufficient range of differentiation among possible green roof system types and practically accommodate all possible green roof systems available in North America.

It is interesting to note that the choice involved in the specification of each green roof layers shown in Figure 2-16 can be considered a variable in green roof selection. However, Grant suggested that varying each subsystem variable across its possible range can generate an unnecessarily cumbersome number of possible solutions. Therefore, it is more practical to use the eight generic types of green roof systems as a basis for system comparison against a reference roof type. The reference roof type was defined as a traditional roof consisting of the layers below the five green roof layers including waterproof membrane, insulation, and roof deck layers. Basically, the framework for vegetated roofing system selection was developed to help designers select the most appropriate roof type for a particular project from a reference roof type and the eight generic green roof types summarized in Table 2-8.

[11] The German Forschungsgesellschaft Landschaftsentwicklung Landschaftsbau (FLL) or Research Society for Landscape Development and Landscape Design

Table 2-8: Generic green roof categories and types, adapted from Grant (2007)

Type of greening	System type	Substrate depth in centimeters (D_{cm})	Substrate depth in inches (D_{in})	Plant type
Extensive greening	1	$2.0 \leq D_{cm} \leq 4.0$	$0.8 \leq D_{in} \leq 1.6$	Moss-sedum
	2	$4.0 < D_{cm} \leq 6.0$	$1.6 < D_{in} \leq 2.4$	Sedum-moss
	3	$6.0 < D_{cm} \leq 10.0$	$2.4 < D_{in} \leq 4.0$	Sedum-moss-herb
	4	$10.0 < D_{cm} \leq 15.0$	$4.0 < D_{in} \leq 6.0$	Sedum-herb-grass
	5	$15.0 < D_{cm} \leq 20.0$	$6.0 < D_{in} \leq 8.0$	Grass-herb
Intensive greening	6	$15.0 < D_{cm} \leq 25.0$	$6.0 < D_{in} \leq 10.0$	Lawn-perennial-small shrub
	7	$25.0 < D_{cm} \leq 50.0$	$10.0 < D_{in} \leq 20.0$	Lawn-perennial-shrub
	8	$50.0 < D_{cm}$	$20.0 < D_{in}$	Lawn-perennial-shrub-tree

The framework for vegetated roofing system selection applied a sound decision-making system known as the Choosing By Advantages (CBA) decision-making system. CBA was originally created by Jim Suhr (1999) for the U.S. Department of Agriculture's Forest Service to help make complex resource allocation decisions in a multiple stakeholder situation. CBA represents a whole process for decision making starting from the moment when a decision is needed and not ending until the implemented decision have been evaluated. CBA processes or methods can be varied depending on the types of decisions. CBA methods are unified by a set of definitions, principles, and models. According to Suhr, the essential principle of the CBA system is that decisions must be based on the importance of advantages rather than the importance of money. In consequence, decisions can be divided into two distinctive types: equal-money or non-money decisions and unequal-money or money decisions. There are CBA methods available for both types of decisions. The methods include the very simple methods for very simple decisions, simple methods for simple decisions, special methods for complex and very complex decisions, and special methods for money decisions.

The recent version of the framework for vegetated roofing system selection in the form of an influence diagram is illustrated in Figure 2-17. This version of the framework applied the CBA tabular method which represents a CBA process for moderately complex decisions together with the CBA cost factor method. The CBA process for moderately complex decisions can normally be divided into five phases: Stage-Setting, Innovation, Decision-Making, Reconsideration, and Implementation. These phases involved in the development of the framework for vegetated roofing system selection were discussed in great detail in Grant's dissertation paper and will be discussed shortly in this document. Referring to Figure 2-17, it is interesting to note that the CBA cost factor method was included as part of the framework to encourage decision makers to examine the results of green roof selection as thoroughly as possible without eliminating the money aspect of green roof systems from their overall consideration. However, cost is not primarily considered in the process of roof system selection because the methods for estimating green roof costs are not well defined. In this version of the framework, cost will be considered if and only if it is necessary after a summation of total importance of the advantages represented by each design alternative is determined.

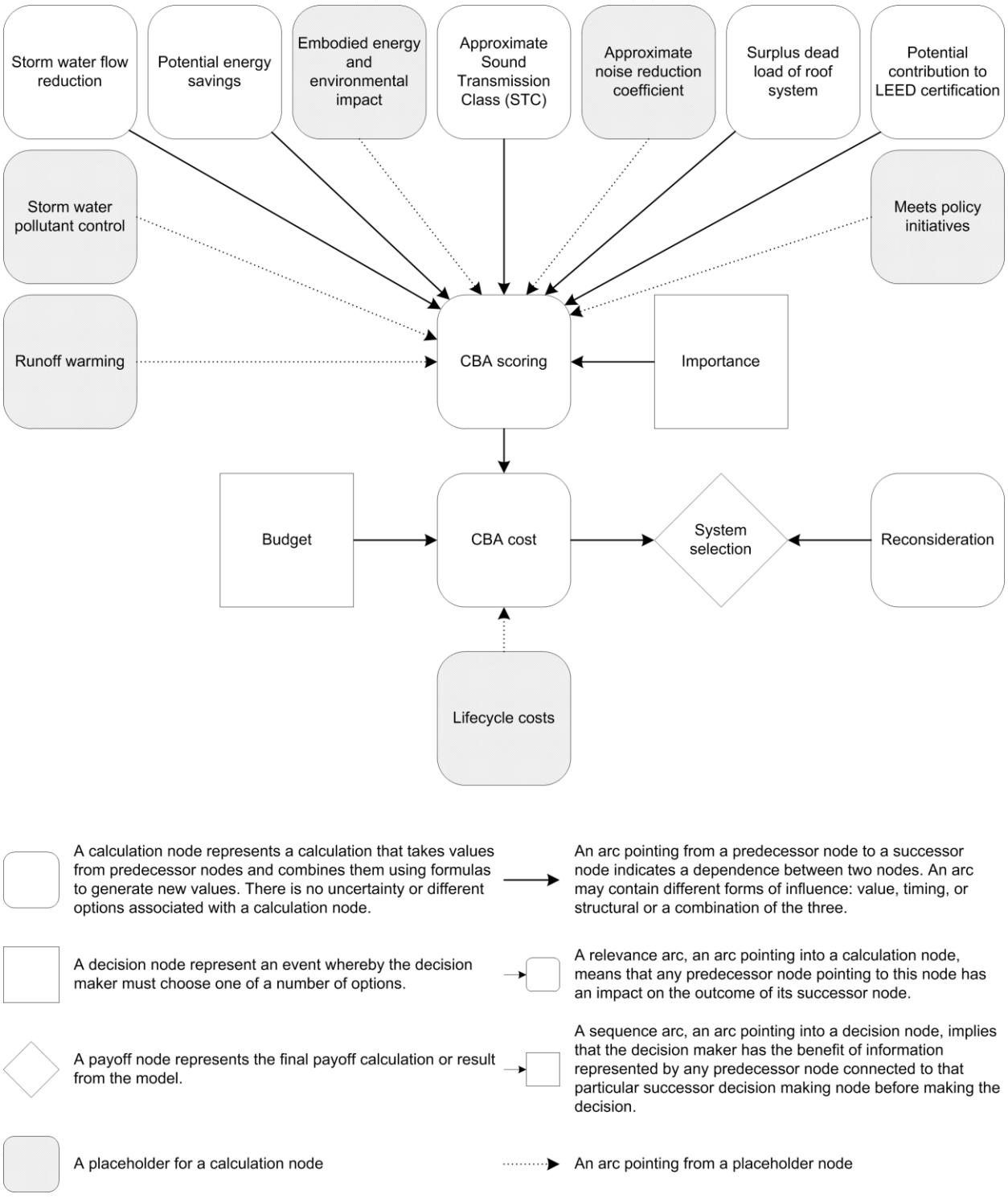


Figure 2-17: Influence diagram of the decision-making for vegetated roofing selection, adapted from Grant (2007)

According to Suhr, the Stage-Setting phase involves defining the purpose and circumstances, problem and its root cause, stakeholders, and criteria of the decision as well as providing CBA training to stakeholders. As mentioned earlier, the framework for vegetated roofing system selection was initially

developed to serve the industrial demands of effective methods for selecting an appropriate green roof system for a particular project. The prospect users of the framework primarily include licensed architects, landscape architects, and roofing consultants or engineers and could be extended to include roofing contractors, horticulturists, specifiers, developers, and policy makers. As part of the framework development, the criteria affecting green roof design was carefully defined and organized into six categories: storm water, energy, acoustics, structure, compliance, and cost. The framework by itself can be considered a navigable path through the decision-making process built upon the CBA system.

The Innovation phase involves formulating a full range of alternatives, determining the relevant attributes of each alternative and the factors containing such attributes, and constructing a detailed comparison display to fully disclose the attributes on the alternative if necessary. Based on published documents, case studies, and expert interviews, Grant was able to formulate a full range of alternatives, which are the eight generic green roof types and a reference traditional roof type. Grant was also successful in defining the relevant attributes and methods for determining their numerical values as well as to identify a set of factors containing such attributes. The applicable factors included in the recent version of the framework consist of the storm water flow reduction, potential energy savings, approximate sound transmission class, surplus dead load of roof system, and potential contribution to LEED certification are presented in Figure 2-17. In her dissertation paper, Grant adapted the Environment Evaluation System (EES), or Battelle method suggested by Dee et al. (1973) to help classify the relevant attributes of alternatives and their numeric values in these factors. The numerical attribute values were determined by multiple decision tree algorithms. The attribute values used in this version of the framework are summarized in Appendix A.

Table 2-9 demonstrates a simplified CBA table displaying the outcomes of the Innovation phase based on Interview No.1 discussed in Grant's dissertation paper. The project discussed in Interview No.1 comprises a 4,400-square-foot (410-square-meter) extensive green roof retrofit on an occupied building, six stories above grade, located in Alexandria, Virginia. In this particular design situation, the Green Roof Type 5, 6, 7, and 8 were excluded because their estimated dead loads were greater than the allowable dead load of 50 pound per square foot. In this case, the structure cannot be upgraded because it was a retrofit project. The Green Roof Type 4 was also excluded from the decision-making process because certain construction and regulation constraints caused by the existing roof system. Furthermore, based on stakeholder opinions, only the storm water flow reduction and surplus dead load of roof system factors were considered relevant factors in this case. In addition to these two factors, the advantages of alternatives in potential energy savings were examined, but they were not impressive enough to be taken into consideration. Nevertheless, the potential energy savings factor is included in Table 2-9 to help demonstrated certain fundamental ideas of the CBA tabular method.

For the CBA tabular method, the Decision-Making phase consists of four steps. The first step is to summarize the attribute of each alternative. For the framework for vegetated roofing system selection, the

values or numeric scores of attributes can be found in Appendix A. The results of this step are summarized in Table 2-10.

Table 2-9: Simplified CBA table displaying the outcomes of The Innovative Phase

Factors	**Alternatives**			
	Reference Roof	**Green Roof Type 1**	**Green Roof Type 2**	**Green Roof Type 3**
Storm water flow reduction Attributes: Advantages:				
Potential Energy Savings Attributes: Advantages:				
Surplus dead load of roof system Attributes: Advantages:				
Total importance:				
Total cost:	Not specified			

Table 2-10: Simplified CBA table displaying the first step of The Decision-making Phase

Factors	**Alternatives**			
	Reference Roof	**Green Roof Type 1**	**Green Roof Type 2**	**Green Roof Type 3**
Storm water flow reduction Attributes: Advantages:	0.20	0.40	0.45	0.50
Potential Energy Savings Attributes: Advantages:	0.44	1	1	1
Surplus dead load of roof system Attributes: Advantages:	1	1	1	1
Total importance:				
Total cost:	Not specified			

The second step of the Decision-Making phase is to decide the advantages of each alternative. This step begins with underlining the least-preferred attribute in each factor. As suggested by Suhr, if there are two or more similar attributes, underline only one. Next, the differences from the least-preferred attributes are summarized. According to Suhr, these differences are the advantages of alternatives. When there is no advantage, the space for showing an advantage is left blank. Table 2-11 shows the results of this step.

The third step of the Decision-Making Phase is to decide the importance of each advantage. This step can be divided into four tasks. The first task is to circle the most important advantage in each factor. If there are two or more alternatives that have the same degree of advantage in each factor, circle only one. The second task is to select the paramount advantage then assign the paramount advantage an importance score of 1, 10, or 100 or any convenient number to establish a scale-of-importance. For the

framework for vegetated roofing system selection, the scale of 0 to 100 was established. The third task is to weigh the importance of each most important advantage, directly or indirectly compared with the paramount advantage. The final task is to decide the importance of each remaining advantage, directly and indirectly compared with the paramount advantage. The outcomes of this step are displayed in Table 2-12.

Table 2-11: Simplified CBA table displaying the second step of The Decision-making Phase

Factors	**Alternatives**			
	Reference Roof	**Green Roof Type 1**	**Green Roof Type 2**	**Green Roof Type 3**
Storm water flow reduction				
Attributes:	0.20	0.40	0.45	0.50
Advantages:		100% more	125% more	150% more
Potential Energy Savings				
Attributes:	0.44	1	1	1
Advantages:		130% more, $40.00/year	130% more, $40.00/year	130% more, $40.00/year
Surplus dead load of roof system				
Attributes:	1	1	1	1
Advantages:	Exist structure sufficient	Exist structure sufficient	Exist structure sufficient	Exist structure sufficient
Total importance:				
Total cost:	Not specified			

Table 2-12: Simplified CBA table displaying the third step of the Decision-Making Phase

Factors	**Alternatives**							
	Reference Roof		**Green Roof Type 1**		**Green Roof Type 2**		**Green Roof Type 3**	
Storm water flow reduction								
Attributes:	0.20		0.40		0.45		0.50	
Advantages:			100% more		125% more		150% more	x
Potential Energy Savings								
Attributes:	0.44		1		1		1	
Advantages:			130% more, $40.00/year	0	130% more, $40.00/year	0	130% more, $40.00/year	0
Surplus dead load of roof system								
Attributes:	1		1		1		1	
Advantages:	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100
Total importance:								
Total cost:	Not specified							

According to Suhr, the paramount advantage can be defined by the Defender-Challenger process. The process starts with selecting one of the advantages as the initial defender and selecting another advantage as the initial challenger. For example, the advantage in storm water flow reduction of Green Roof Type 3 is selected as the initial defender. The advantage in potential energy savings of Green Roof Type 1 is then selected as the initial challenger. Afterward, the decision maker can ask the first question as follows:

Question - In your opinion, which of the two advantages is the most important: the 150% advantage in storm water flow reduction of Green Roof Type 3 or the 130% ($40.00/year net savings) advantage in potential energy savings of Green Roof Type 1?

Answer - The 150% advantage in storm water flow reduction of Green Roof Type 3, because the 130% ($40.00/year net savings) advantage in potential energy savings of Green Roof Type 1 seems to have close to no advantage at all.

The answer to the first question suggests that the advantage in storm water reduction of Green Roof Type 3 remains the defender. Next, the advantage in surplus dead load of the roof system of Reference Roof is selected as the challenger. In consequence, the decision maker can ask the second question as follows:

Question - In your opinion, which of the two advantages is the most important: the 150% advantage in storm water flow reduction of Green Roof Type 3 or the sufficient structural advantage in surplus dead load of the roof system of Reference Roof?

Answer - The structure sufficient advantage in surplus dead load of roof system is the most important one.

The answer to the second question suggests that the sufficient structural advantage in surplus dead load of the roof system of Reference Roof is the paramount advantage. As brought up earlier, the framework for vegetated roofing system selection use the scale-of-importance, ranging from 0 to 100. As a result, the importance score of 100 can be given to the structure sufficient advantage in surplus dead load of the roof system of Reference Roof as shown in Table 2-12. It is interesting to note that the sufficient structural advantages in surplus dead load of the roof system of Green Roof Type 1, 2 and 3 are also given the importance score of 100 because their advantages are similar to Reference Roof in this factor.

After the paramount advantage is defined, the decision maker can begin to weigh the importance of each most important advantage, compared directly or indirectly with the paramount advantage based on the similar scale-of-importance, ranging from 0 to 100. In the design situation discussed in Interview No.1, the 130% ($40.00/year net savings) advantages in potential energy savings of alternatives were given a similar importance score of 0 because there was no significant advantage in this factor. Based on Grant's assumptions given in her dissertation paper, the 150% advantage in storm water flow reduction of Green Roof Type 3 can be considered the most important advantage after the advantages in surplus dead load of the roof system of alternatives. Therefore, the importance score of x can be assigned to the advantage in water flow reduction of Green Roof Type 3 without assigning other importance scores to the advantages in this factor of other alternatives. All assigned importance scores will be used in the final step of the Decision-Making phase.

The final step of the Decision-Making phase is to calculate the total importance of the advantages. According to Suhr, when the costs of the alternatives are equal and even sometimes when they are

unequal, the decision maker is always encouraged to choose the alternative with the greatest total importance of advantages. At the end of this step, the greatest summation of total importance of the advantages is double-underlined and the alternative that has the double-underlined value is chosen. Table 2-13 demonstrates the results of this step in which Green Roof Type 3 is chosen for this particular project with the total importance of 100+x.

Table 2-13: Simplified CBA table displaying the fourth step of The Decision-Making Phase

Factors	**Alternatives**							
	Reference roof		**Green roof type 1**		**Green roof type 2**		**Green roof type 3**	
Storm water flow reduction								
Attributes:	0.20		0.40		0.45		0.50	
Advantages:			100% more		125% more		150% more	x
Potential Energy Savings								
Attributes:	0.44		1		1		1	
Advantages:			130% more, $40.00/year	0	130% more, $40.00/year	0	130% more, $40.00/year	0
Surplus dead load of roof system								
Attributes:	1		1		1		1	
Advantages:	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100
Total importance:		100		100		100		100+x
Total cost:	Not specified							

While Grant's assumptions concerning the importance score of the advantage in storm water flow reduction of Green Roof Type 3 made a lot of sense in the given example, it is also interesting to examine the typical process originally suggested by Suhr for determining the importance scores of advantages in a particular factor of alternatives. The process begins with establishing a numerical scale-of-importance. In this case, the advantage in surplus dead load of the roof system of Reference Roof has an importance score of 100. Next, at the bottom of the scale, indicate that when there is no advantage there is no importance. Afterward, identify the crossover point. Figure 2-18 demonstrated the established numeric scale of importance. As shown in Figure 2-18, the crossover point is assumed to be the point in which the dead load of a possible green roof system is equal to the allowance dead load, which falls between Green Roof Type 4 and 5. After the crossover point is identified, the numeric scores of importance of the advantage of a particular alternative can be estimated as shown in Figure 2-18. Table 2-14 demonstrates the modified CBA table using the established importance scores. It is interesting to note that this method allows the comparison results to be visually displayed. According to Suhr, the results can be effectively displayed in the form of a staked column bar chart as shown in Figure 2-19.

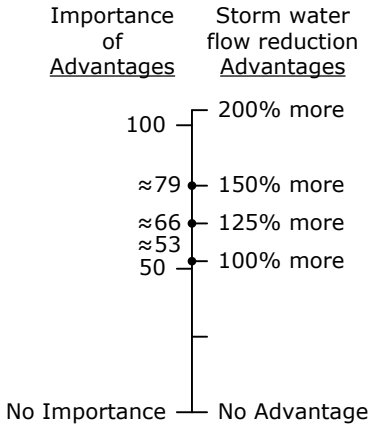


Figure 2-18: Advantage in surplus dead load of roof system vs. advantage in storm water flow reduction

Table 2-14: Simplified CBA table displaying the fourth step of The Decision-Making Phase with the numeric scores of advantages in storm water flow reduction

Factors	**Alternatives**							
	Reference roof		**Green roof type 1**		**Green roof type 2**		**Green roof type 3**	
Storm water flow reduction								
Attributes:	0.20		0.40		0.45		0.50	
Advantages:			100% more	53	125% more	66	150% more	79
Potential Energy Savings								
Attributes:	0.44		1		1		1	
Advantages:			130% more, $40.00/year	0	130% more, $40.00/year	0	130% more, $40.00/year	0
Surplus dead load of roof system								
Attributes:	1		1		1		1	
Advantages:	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100	Exist structure sufficient	100
Total importance:		100		153		166		179
Total cost:	Not specified							

After the decision was made as a result of the Decision-Making phase, the next phase of the CBA process is the Reconsideration phase. As stated by Suhr, this phase includes reconsidering the decision, changing the decision if it should be changed, forming clear, accurate, sensory-rich, motivational perception of the advantages of the selected alternative, and making a strong commitment to successfully implement the final decision as well as removing any barriers to its implementation. Referring to Figure 2-17, The Reconsideration Phase was included into the current version of the framework for vegetated roofing system selection. According to Grant, it was included as a placeholder for any additional information that may either not be included in the parameters under review, or which may serve to refine their values or the importance assigned to the advantages of one system over others.

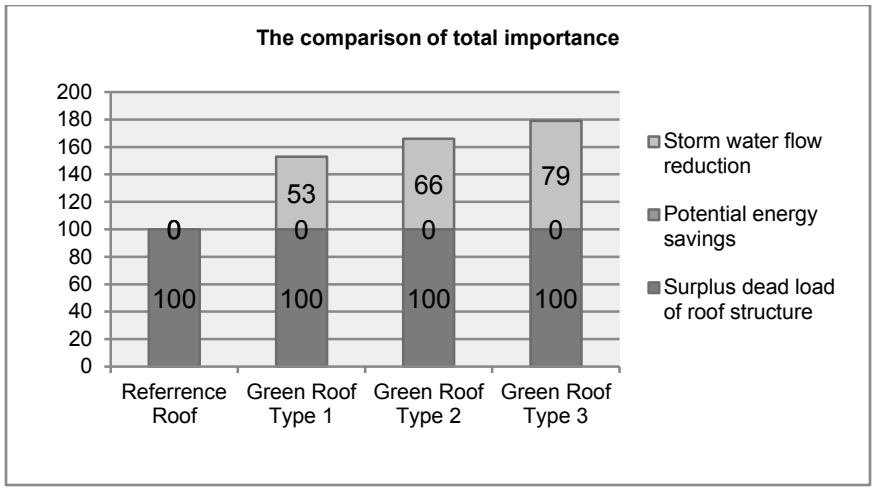


Figure 2-19: Comparison of total importance

The last phase of the CBA process, the Implementation phase, involves implementing the decision, adjusting the decision during implementation if necessary, and evaluating the process and results. As part of Grant's study, the framework was tested for acceptability which represents certain activities of this phase. The framework was evaluated by comparison with the method used by experts in the field of green roof design as captured through an interview instrument as well as by project specific tests of the decision-making model. According to Grant, interviewees were solicited based on her knowledge of their experience with green roofing and the location of their green roof practices within the Mid-Atlantic region. There are three designers who completed two web-based surveys entitled "Introduction, Process and Rationale Questionnaire" and "Framework Input Questionnaire" together with one phone interview. The three sets of surveys and interview responses discussed in Grant's dissertation paper were based on completed projects located in Northern Virginia, chosen by the interviewees. The test results indicate the framework proved acceptably flexible to accommodate the simplicity or complexity of the actual decision situation in each case.

From the perspective of DSS development, the framework for vegetated roofing system selection has considerable potential to be developed further in the form of computer-based DSS, especially in the category of Expert Systems (ES). According to Turban et al. (2007), ES are one of Artificial Intelligence (AI) applications in which the term was derived from the term "knowledge-based expert system". MYCIN is one of the most well-known ES developed by a group of researchers at Stanford University in 1970s for helping doctors diagnose bacterial infection of the blood. Turban et al. defined an ES as a system that used human knowledge captured in a computer to solve problems that normally require human expertise. Expertise can be defined as the extensive, task-specific knowledge that experts possess. In the field of ES, an expert is a person who has special knowledge, judgment, experience, and methods to give advice and solve problems and is able to apply these aptitudes. Altogether, experts have expertise that can

solve a problem with a performance level that is significantly better than the average person and can explain certain phenomena in the problem domain (Turban et al. 2007).

As stated by Turban et al., ES can be considered an excellent tool for preserving professional knowledge vital to a company's competitiveness as well as for documenting professional knowledge for examination or improvement. ES are also a good tool for training new employees and for distributing knowledge in an organization. Furthermore, ES allow knowledge to be transferred more easily at a lower cost and can be useful tools for knowledge storage and transfer in organizations.

Turban et al. divided ES into two generations. The first generation ES use if-then rules to represent and store their knowledge such as rule-based ES. The second generation ES adopt multiple knowledge representation and reasoning methods including, but not limited to, decision trees, decision tables, and neural networks. Turban et al. also categorized ES into ten generic categories based on the problem areas that they address. Design is one of the problem areas addressed by ES. Circuit layout, building design, and plant layout are several design problems described by Turban et al. In general, ES for design problems deal with configuring objects under constraints. Design expert systems are capable of constructing descriptions of objects in various relationships with one another and ensuring that these configurations conform to stated constraints.

The framework for vegetated roofing selection has considerable potential to be developed further in the form of an ES for solving green roof design problems, because it fulfills the four features of ES suggested by Turban et al. The four features of ES include expertise, symbolic reasoning, deep knowledge, and self-knowledge. For the expertise aspect, the framework possesses expertise acquired from literature reviews, case studies, and expert interviews, which enables it to help make expert-level decisions on green roof system selection. For the symbolic reasoning aspect, the primary mechanism of, and knowledge in the framework are symbolically represented; for examples, the knowledge used to determine the numeric values of attributes are symbolically represented in the decision tree format. For the deep knowledge aspect, it can be considered that the knowledge necessary for green roof system selection contained in the framework is hardly found in non-experts. For the self-knowledge aspect, although the framework was not initially created as a computer-based system that can learn from its successes and failures, the framework applied the CBA tabular method, which allows a decision maker to examine the framework reasoning and provides proper explanations as to why a particular conclusion was reached.

Figure 2-20 demonstrates a schematic view of an expert system. According to Turban et al., an ES such as the one shown in Figure 2-20 normally consists of two environments: the consultation environment and the development environment. The consultation environment is typically used by a user to obtain expert knowledge and advice. The development environment is normally used by an ES developer team to build the components and put knowledge into the knowledge base. While Figure 2-20 displays a number of ES components, there are only three major components that can be found in most ES: knowledge base, inference engine, and user interface. The knowledge base is the foundation of an ES which contains the

relevant knowledge for understanding, formulating, and solving problems. The inference engine, also known as the control structure of the rule interpreter in rule-based ES, is a computer program that provides a methodology for reasoning about information in the knowledge base and on the blackboard as well as for formulating conclusions. The user interface contained in an ES is a language processor for friendly, problem-oriented communication between the user and computer. According to Turban et al., most existing systems use the question-and-answer approach to interact with the user due to technological constraints and may be supplemented by menus, electronic forms, and graphic.

In addition to the three major components, the knowledge acquisition subsystem, blackboard, and explanation subsystem are also likely to be found in ES that interacts with the user. As stated by Turban et al., knowledge acquisition is the accumulation, transfer, and transformation of problem-solving expertise from experts or documented knowledge sources to a computer program. The potential knowledge sources may include human experts, textbooks, multimedia documents, databases, special research reports, and information available on the Web. Knowledge acquisition can also be seen as a process for constructing and expanding the knowledge base. The blackboard is an area of working memory set aside as a database for the description of a current problem specified by the input data. The blackboard is also used for recording intermediate hypotheses and decisions. The explanation subsystem provides the ability to trace responsibility for conclusions to their sources and explain the ES behavior by interactively answering questions used to derive the specific recommendations. Other than these three components, the knowledge refinement component is also included in Figure 2-20 although it is rarely found in most ES. The knowledge-refining system provides the ability to analyze the reasons for its success or failure which could lead to improvements that result in a more accurate knowledge base reasoning. The critical component for knowledge refinement is the machine-learning component which is currently being developed in experimental ES at several universities and research institutions.

In addition to the generic structure of an ES, Turban et al. also suggested a typical process for ES development. The process includes knowledge acquisition, knowledge representation, selection of development tools, system prototyping, evaluation, and improvement. The knowledge acquisition phase involves defining the nature and scope of the problem, identifying proper experts, and acquiring knowledge. The result of knowledge acquisition is a knowledge base that can be represented in different formats during the knowledge representation phase. It is interesting to note that these two phases of the process were accomplished in Grant's study. In her work, Grant defined the nature and scope of the problem in relation to the design of vegetated roofing systems. Grant was also able to identify experts in the fields related to vegetated roofing system design and acquire their knowledge through electronic surveys and phone interviews. In her study, knowledge was also acquired from other knowledge sources including relevant published works and case studies. The knowledge was evaluated for its consistency and applicability through the test for acceptability. According to Grant's dissertation paper, the acquired knowledge was mainly represented in two formats: decision tree and decision table. The decision tree format was used to represent a set of knowledge necessary for determining the attribute values of

alternatives. The decision table format was used to represent the whole process of CBA decision-making system. Although it was not discussed by Grant, the question-and-answer processes such as the Defender-Challenger process used for identifying the paramount advantage can also be converted into if-then rules then chained using either forward or backward chaining approaches.

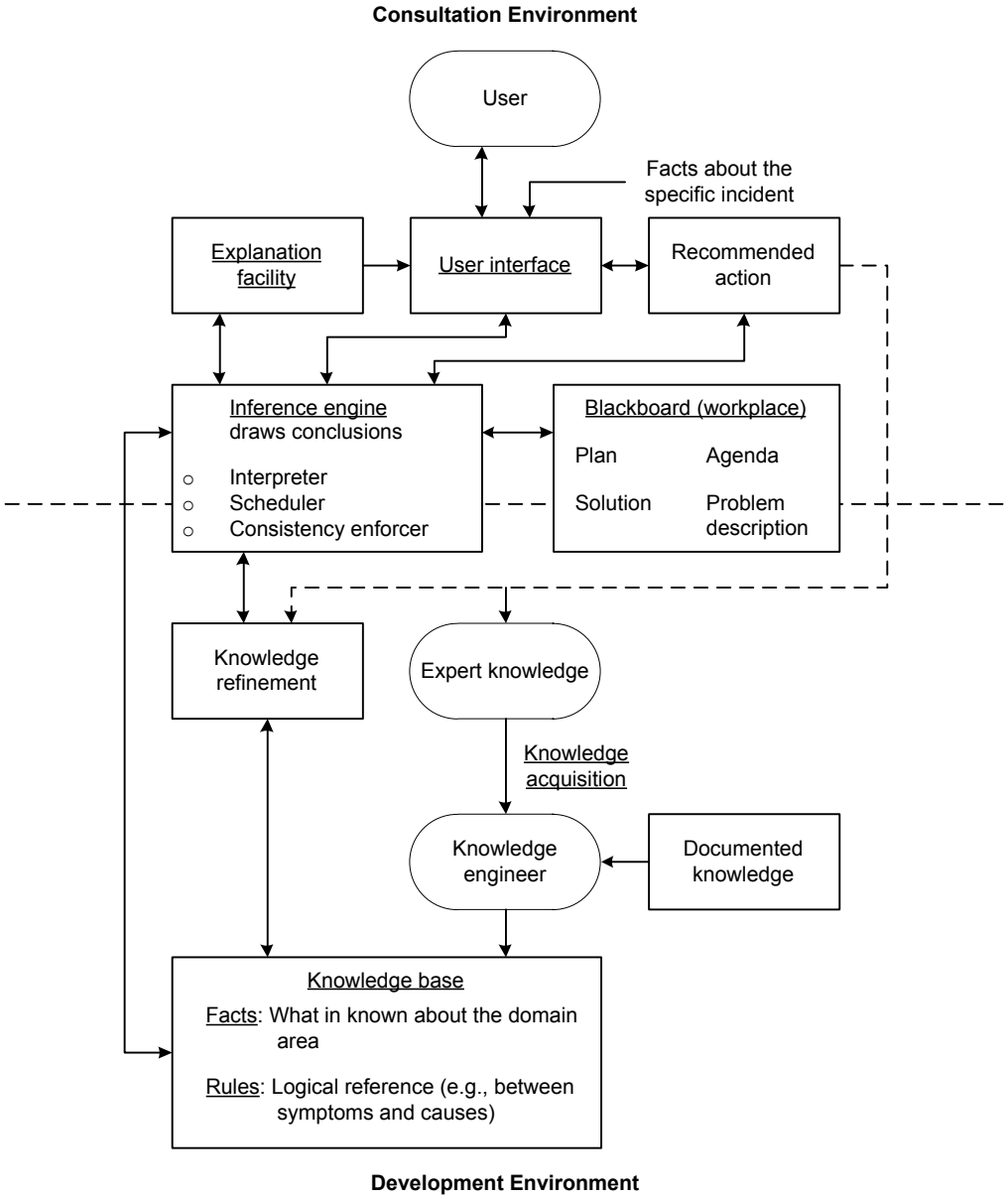


Figure 2-20: Schematic structure of an Expert System (ES), adapted from Turban et al (Turban et al. 2007).

Continuing from Grant's study, it can be considered that this research emphasizes on the selection of development tools, system prototyping, and evaluation phases of the ES development process. According to Turban et al., the selection of development tools phase involves selecting an ES development tool from three different kinds of tools including the general-purpose development

environment, ES shells, and tailored turn-key solutions. After a development tool is selected, a prototype ES then can be developed based on the synthetic requirement of the selected tool during the system prototyping phase. Next, the evaluation phase involves verifying and validating the developed prototype. As stated by Turban et al., verification ensures that no error occurred at the coding stage and validation ensures that the system can solve the problem correctly and effectively. After the system is evaluated, it can be improved further in the improvement phase. The improvement phase can be done in a future study when the prototype is mature enough to be developed further in a formal development process such as SDLC or RAD.

Figure 2-21 represents the discussed ES development process. It is important to note that the ES development process demonstrated in Figure 2-21 can be seen as a guideline for the development of ES components that support the knowledge-based subsystem of the prototype DSS developed as part of this dissertation. In a broader view, the issues concerning the selection of development tools should be addressed as part of the feasibility study report and the issues related to system prototyping and evaluation should be included as part of the prototyping process.

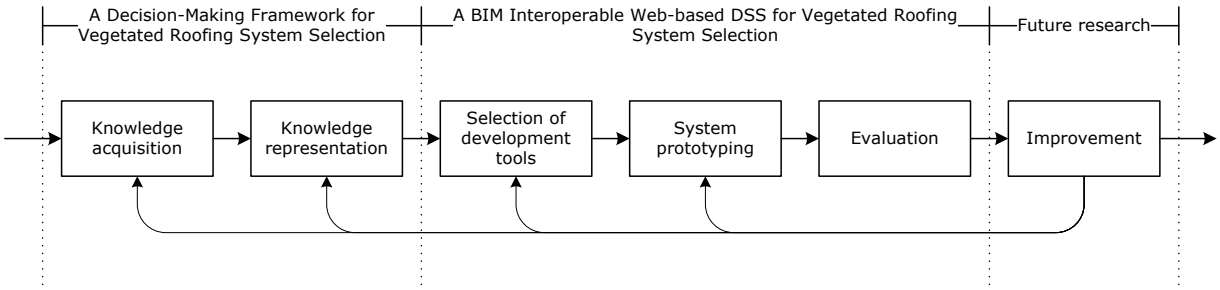


Figure 2-21: Process for ES development

Besides the ES generic structure and development process, Turban et al. also provided certain suggestions about the potential problems and limitations of ES. Essentially, the problems and limitations fall within three areas: knowledge acquisition, system performance, and system operation. The knowledge acquisition problems and limitations may include the availability of knowledge, human expertise extraction, and knowledge transfer. The system performance problems and limitations may comprise the accuracy and consistency of problem solving and system costs. The system operation problems and limitations may include the user's cognitive limits and trust. As suggested by Turban et al., the technical problems and limitations of ES can be reduced by the rapid progress of information technology. Correspondingly, ES tend to be more cost-effective when they are distributed through the Web. Turban et al. also suggested that the level of managerial and user involvement directly influences the success level of ES. The cooperation of experts and end user training programs show considerable potential in improving the success rate of ES, especially in the areas of knowledge acquisition and system operation.

In conclusion, the development of a prototype DSS discussed in this dissertation adapts the framework for vegetated roofing system selection created by Grant (2007). The framework applied the Choosing By Advantages (CBA) system developed by Suhr (1999) as a basis of decision making process. Based on its features, the framework has significant potential to be developed in the form of an Expert System (ES). From the development stand point, there are two major components of conventional ES that will be integrated to the knowledge-based management subsystem of the prototype DSS: knowledge base and inference engine. Within the scope of this dissertation, the knowledge acquired from published documents, case studies, and expert interviews described in Grant's study will be converted to computer-understandable problem-solving and decision-making rules, which then will be stored in the knowledge base. Knowledge in the knowledge base then will be utilized by the influence engine to infer conclusions. In a broader view, the knowledge-based management subsystem works in conjunction with other parts of the DSS including data, model, and interface management subsystems to provide relevant suggestions for practitioners or decision makers about green roof selection. Finally, the usefulness of this approach to extend practitioner knowledge bases and the design process will be assessed.

Chapter 3

3. Methodology

3.1. Research Organization

This study adopts a conceptual model of concentric frames suggested by Groat and Wang (2002) as a basis for constructing the research framework. According to Groat and Wang, the model was developed to help a researcher maintain the coherence and continuity among the systems of inquiry, strategies, and tactics. In this model, the systems of inquiry can be seen as the outermost border that frames the choice among a range of research strategies or designs. Consequently, the choice among a range of strategies, the frame between the outermost and innermost frames, then narrows down the choice among a range of tactics, the innermost frame. Based on Groat and Wang's conceptual model of concentric frames, the methodology of this research will be systematically discussed throughout this chapter.

According to Groat and Wang, the system or systems of inquiry can be chosen from the three paradigmatic clusters: postpositivist, naturalistic, and emancipatory. Each cluster represents a set of research paradigms that share similar ontological assumptions, also referred to as the nature of reality, and epistemological assumptions, also referred to as the nature of knowledge and the relation between knower and would-be-known. For the paradigms in the postpositivist cluster, it is assumed that there is only one reality which can be known within a certain level of probability. Postpositivism assumes that objectivity is important and may be imperfectly realized. In postpositivist research, researchers normally manipulate and observe research subjects in a dispassionate, objective manner. For the paradigms in the naturalistic cluster, it is assumed that there are multiple, socially constructed realities. Naturalistic researchers acknowledge the interactive connection between researchers and participants. In naturalistic research, researchers play an important role in defining the theoretical position and values essential to their research. Naturalistic researchers also consider the role of interpretation and creation in reporting their research findings. For the paradigms in the emancipatory cluster, it is assumed that there are multiple realities which could be shaped by social, political, cultural, economic, ethnic, gender, and disability values. Emancipatory researchers take into consideration the interactive link between researchers and participants as well as consider that knowledge is socially and historically situated. It is interesting to note that Groat and Wang's paradigmatic clusters allow a researcher to flexibly adopt the combined systems of inquiry selected from the three clusters in one research model.

Based on Groat and Wang's three paradigmatic clusters, this study can be considered naturalistic research by its very nature. In general, this study involves developing a prototype web-based DSS and examining the usefulness of the prototype. From the ontological standpoint, this research assumes that there are multiple, socially constructed realities which can be reflected in the studies on DSS prototyping as well as on the usefulness of the prototype. According to Sprague and Carlson (1982), DSS are

adaptive systems, which evolve to accommodate different behavior styles and capabilities over their lifetime. Sprague and Carlson suggested that there is no comprehensive theory of decision making and the conditions that decision makers face tend to change rapidly. As a result, it is impossible to find an absolute DSS solution for a particular problem that decision makers encounter. The lack of a comprehensive theory of decision-making and the rapid change of decision making conditions affect the usefulness of DSS. Frequently, DSS tend to be more useful over time when they are more mature.

From the epistemological standpoint, it can be considered that the interactions between Sprague and Carlson's five evolving roles in DSS, the manager or user, the intermediary, the DSS builder, the technical support, and the toolsmith, are very critical to the success of DSS development. Furthermore the interactive link between a researcher/DSS developer and decision makers/DSS users is also very important for the study on the usefulness of the prototype DSS. When considering this research as a whole, the researcher can be seen as an influential part of the studies on DSS development and the usefulness of the prototype. Figure 3.1 demonstrates the system of inquiry selected for this research, relative to Groat and Wang's paradigmatic clusters.

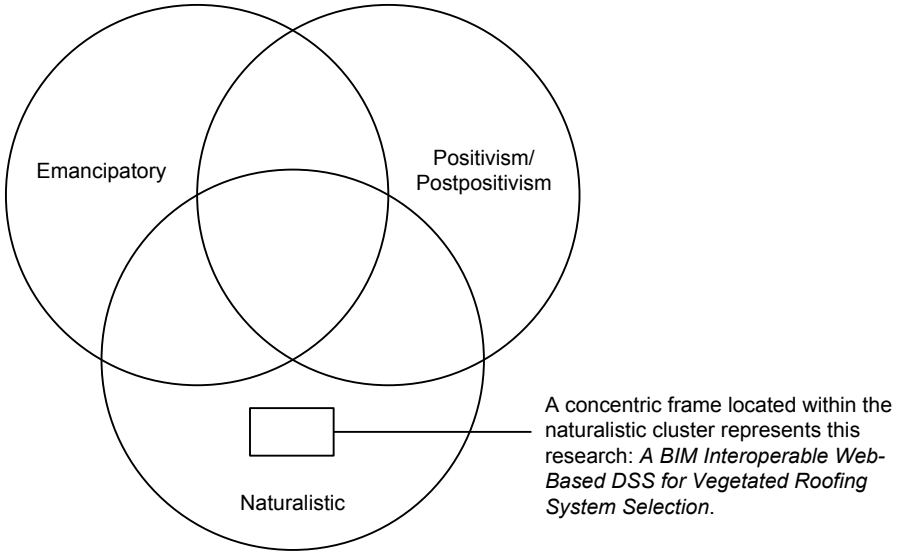


Figure 3-1: Research system of inquiry

In addition to the three paradigmatic clusters, Groat and Wang also suggested certain criteria used for evaluating research quality at the paradigmatic level based on Guba's (1981, cited by Groat and Wang 2002) quality standards for naturalistic inquiry. For naturalistic research, the evaluation criteria include credibility, transferability, dependability, and confirmability. Credibility involves establishing truth value, which entails a holistic approach to the research problem. Truth value can be established by contemplating the natural complexities associated with the situation or circumstance being studied and can be demonstrated in two ways: triangulation and member checks. Transferability takes into consideration the extent in which the conclusions of one study can be applied to another setting or circumstance. A researcher can achieve transferability by providing a sufficiently "thick" description that

the relative similarity of the two contexts can be extensively assessed. Dependability considers the tendency that data can be unstable, which may be caused by different realities being tapped or by instrumental shifts stemming from developing insights on the part of the investigator-as-instrument of research. In order to ensure dependability, the audit trial, which documents all the process whereby data were collected, analyzed, and interpreted, must be established. Confirmability assures that the investigator's data and interpretations are verifiable. Confirmability can be accomplished via a combination of triangulation and reflexivity on the part of researcher.

This research adopts the measures of naturalistic research quality discussed above. In order to demonstrate credibility, this research primarily implements one of combined research strategies suggested by Groat and Wang, the two-phase research design. According to Groat and Wang, the major advantage of combined research strategies is triangulation which is the principle of combining strength and neutralizing weakness of two or more research strategies. Based on the principle of triangulation, this study employs two research strategies: logical argumentation and qualitative research. The combined strategy used in this study will be discussed further shortly in this section.

For transferability, it can be considered that this study focuses on two specific problems: 1) the development of a prototype BIM interoperable web-based DSS for vegetated roofing system selection and 2) the usefulness of DSS in improving the quality of vegetated roofing system design decisions. In order to confront these problems, a specific process for DSS prototyping and a particular process for defining the usefulness of the developed DSS are applied. Although these processes are specifically developed corresponding to the encountered problems, the processes are fundamentally derived from the processes that are widely used in practice. Therefore, it can be assumed that the processes applied in this study are not completely unique when compared to other processes for DSS prototyping and evaluation and can be adapted to other applications in general.

In order to achieve dependability, the audit trial will be conducted as one of research activities using documentation methods that are appropriate to the selected research strategies and their associating tactics. For the DSS development, the standardized documentation methods such as the Unified Modeling Language (UML) created by the Object Management Group (OMG) can be applied as a basis for documentation. For the study of the usefulness of the developed DSS, one or more methods suggested by Groat and Wang, such as interview notes, can be applied.

Furthermore, in order to ensure confirmability, this study aims to combine the triangulation of research methods discussed earlier with the practice of reflexivity. In this study, it is assumed that the investigator’s reflections on the issues of DSS prototyping and defining the usefulness of developed prototype are considerably based on personal background in architectural design and research and DSS design and development. Essentially, the reflections will be provided as part of research findings. If any change in personal perspective emerges during the course of study, it will be deliberated as appropriate.

After the system of inquiry is defined, the research strategies and their associating tactics can be framed. As mentioned earlier, this study is one of combined research strategies suggested by Groat and Wang defined as two-phase research design in which two or more strategies are combined in a sequence of distinct phases. As a result, the particular procedures and standards associated with each strategy can be presented fully and distinctly. According to Groat and Wang, it is important to consider the potential lack of coherence and connection between research strategies when adopting this strategy. For this research, it can be considered that the study on DSS prototyping is distinct from the study on the usefulness of the developed system. In other words, the prototype DSS has to be developed before it can be tested and evaluated by its prospective users. This also explains the coherency and connection between the two processes whereby the outcomes of one process are expected to be used in the process that comes after it.

According to Groat and Wang, research that involves computer software development usually implements the logical argumentation research strategy. In general, Groat and Wang divided logical argumentation research into three categories based on three underlining logical systems: formal-mathematic, mathematical-cultural, and cultural-discursive. Between these three logical systems, research that comprises the development of computer programs tends to implement the formal-mathematic systems, which depend heavily upon rule-based propositions that are readily adaptable to the computer.

For formal-mathematic logical systems, Groat and Wang suggested a tactic whereby a logical system can be framed by mathematical representation and/or computer modeling. As stated by Groat and Wang, a logical system can be framed by capturing the behavior of an empirical reality in mathematical terms, which can then be represented by a system of mathematical rules and relationships. Subsequently, computer programs can be developed based on such mathematical rules and relationships. It can be considered that this tactic is similar to the process for expert system development discussed in the previous chapter. This research involves developing a web-based DSS for vegetated roofing system selection derived from knowledge that has been captured from different knowledge sources and represented in computer understandable formats. In order to develop the DSS effectively, this study adopts an evolutionary software engineering process, the prototyping process, which will be discussed further in this chapter.

While formal-mathematic systems seem to be appropriate for this study, it has certain limitations that should be taken into consideration. According to Groat and Wang, most of logical systems tend to have a limited lifetime, because they become outdated with progress or with changes in the empirical environment that produced them. Consequently, formal-mathematical systems change by means of new evidence that emerges to alter constructed paradigms. Based on this shortcoming, it is important to ensure that the DSS development process and the developed DSS itself can accommodate unexpected changes flexibly and effectively. In addition to the issue of system lifetime, Groat and Wang also suggested that a logical system may not be an accurate representation of the reality it intends to explain

although such a system is internally consistent from the logical point of view. This is the reason why the study on the usefulness of the developed DSS has become very important to the overall research. This process helps confirm that if a BIM interoperable web-based DSS for vegetated roofing system selection can be developed, it should also be useful to its prospective users.

Because the logical argumentation strategy used to frame the study on DSS development could possibly make the DSS solution inapplicable to real world situations even though it is fully functional, the study on the usefulness of the prototype DSS is necessary to be included as part of this research. Therefore, it is appropriate to combine the logical argumentation strategy with another strategy to neutralize this weakness. Based on research strategies suggested by Groat and Wang, the qualitative research strategy tends to be the most suitable for this combined research. This is because qualitative research has the capacity to take in rich and holistic qualities of real life circumstances and sensitivity to meanings and processes of people's activities and artifacts. Furthermore, it is flexible in design and procedures, which allows for adjustment to be made as the research proceeds. This flexibility allows the qualitative research approach to accommodate the changes of logical systems effectively.

As a qualitative study, the study on the usefulness of the prototype DSS adopts the interpretivism approach suggested by Groat and Wang. Interpretivism is a qualitative approach that considers the value of the permanence and priority of the real world of subjective, first-person experience. Therefore, within the scope of this research, it is important to learn of the usefulness of the prototype DSS from the user's hands-on experience.

According to Groat and Wang, qualitative research can be organized by a process that represents the interactive relationship among data collection, data reduction, data display, and conclusion drawing and/or verifying. The process begins with the data collection phase. For data collection, this study implements the interview tactic which comprises interactive activities that may include in-depth interviews, key informants interviews, and career histories. Subsequently, the process continues on to the data reduction phase whereby the volume of collected data is reduced into manageable "chunks". Data reduction can be done by coding the "chunks" into various themes. As suggested by Groat and Wang, coding themes or schemes should be clearly documented for ongoing use as well as for research records. Typically, the codes can be noted in the margins of transcripts or documents. Next, for the data display phase, Groat and Wang suggested that data can be organized and displayed in the form of charts, graphs, and tables, which can be useful for data analysis and communication. Finally, after the data are coded, reduced, and displayed, the conclusion drawing and/or verifying phase involves identifying patterns, providing explanations, and evaluating the findings.

As suggested by Groat and Wang, when adopting qualitative research as a research strategy, it is important to take into consideration the weaknesses of this study. First there is the potential challenge of dealing with vast quantities of data. For this aspect, within the scope of this combined research, the logical argumentation research conducted prior to the qualitative research tends to be helpful in narrowing

down the quantities of data. Second, because the research design and procedures of qualitative research are highly flexible, there is a need to establish few guidelines or step-by-step procedures. In order to respond to this issue, a procedure for examining the usefulness of the prototype DSS will be discussed further in this chapter. Third, the credibility of qualitative data can be questionable in the postpositivist paradigm. Within the scope of this research, the weakness concerning the credibility can be lessened by the triangulation of research strategies whereby the logical argumentation strategy is combined with qualitative research.

It is essential to note that the logical argumentation and qualitative research strategies in combination are also often used in real world DSS projects to improve the effectiveness of DSS, which in turn helps increase the success rate of the projects. According to Turban et al. (2007), the term “effectiveness” can be defined as "the degree of goal attainment" or "doing the right things". As suggested by experts in the field of DSS such as Keen (1980), the goal of DSS is to help improve the decision maker's decisions and productivity. In other words, what DSS are supposed to do is to provide decision makers with the right information, in the right format, at the right time, and at the right cost (Power 2000).

Frequently, the effectiveness of DSS tends to be qualitative and intangible at the beginning stages and becomes more quantitative in terms of costs and benefits in the later stages of the System Development Lifecycle (SDLC). While there are a number of methods that can be used to improve the effectiveness of DSS at different stages of the SDLC, experts in the field of DSS including, but not limited to, Sprague and Carlson (1982) and Power (2000) recommended that the value analysis method proposed by Keen (1980) in his article, *Value Analysis: Justifying Decision Support Systems*, can be considered the most appropriate method for the DSS that is developed for the first time. This is because this method focuses on value and recognition that qualitative benefits are of central relevance, can reduce uncertainty and risk, and encourages innovation. In addition, this method emphasizes the use of prototyping and qualitative study to help improve the effectiveness of DSS. It can be considered that this method has considerable potential to help strengthen the research methodology proposed in this study, which should be briefly discussed in this section.

According to Keen (1980), value analysis is a systematic method that emphasize value first, cost second, simplicity, and robustness by assuming that there is no need to pin down precise estimates of uncertain, qualitative future variables. This method also focuses on reducing uncertainty and risk. In addition, this method revolves around innovation rather than routinization. As stated by Keen, value analysis uses prototyping as a mechanism for factoring risk by reducing the initial investment, the delay caused by the project approval process, and the delivery of a tangible product. Prototyping is also used as a mechanism for separating cost and benefit by keeping the initial investment within a relatively small, predictable range.

Value analysis comprises three stages as shown in Figure 3-2. The first stage begins with establishing value, which focuses on defining an operational list of benefits and cost threshold, and then building

"Version 0", a prototype, which encompasses defining the system architecture and routines for the prototype. As stated by Keen, this stage is similar to the way in which effective decisions to adopt innovations are made. Continuing from the first stage, the second stage involves assessing the prototype and establishing the cost of "Version 1", the full system. Afterward, the last stage involves building "Version 1", monitoring benefits and costs, and then evolving to "Version N".

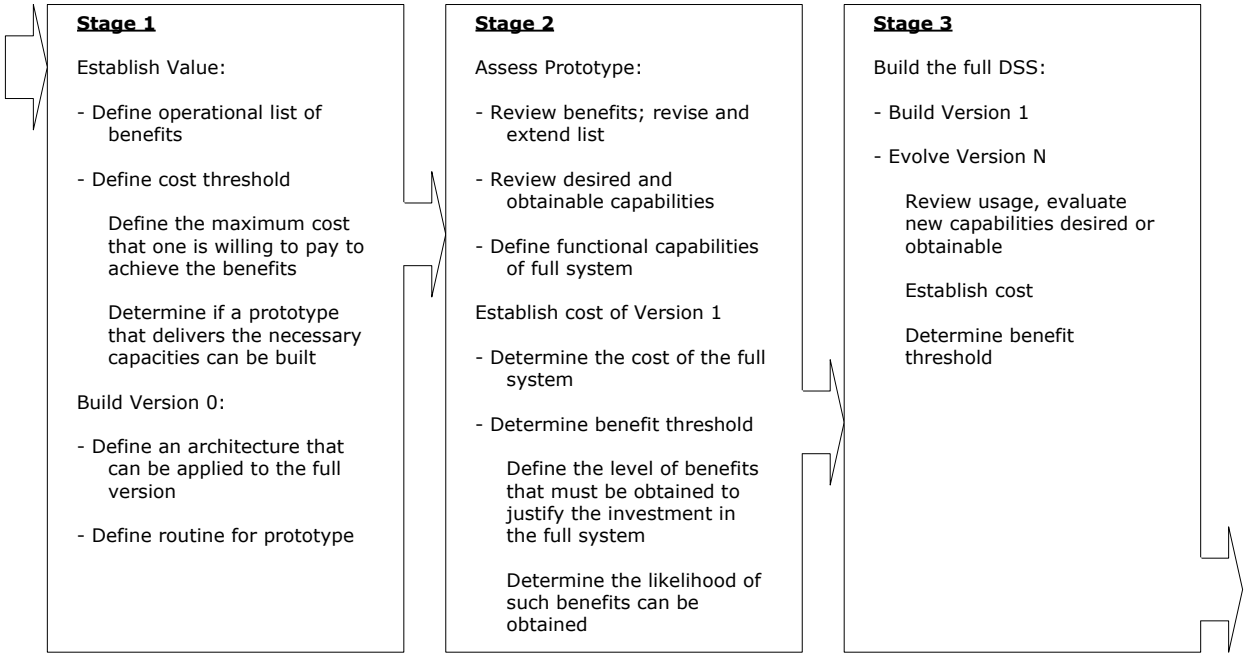


Figure 3-2: Value analysis, adapted from Keen (1980)

As shown in Figure 3-2, the activities involved in value analysis seem to be compatible with and can be easily adapted to the prototyping and qualitative research processes proposed in this study. The value analysis method helps define the measures of the effectiveness of the prototype in the form of operational benefits and the cost threshold of prototype development. With the operational benefits and cost threshold in mind, the prototype can be developed in order to deliver the defined operational benefits at the right cost through the prototyping process. After the prototype is developed, the effectiveness of the prototype can be further studied using the qualitative research process. As part of the qualitative research process, the measures of the effectiveness can be transformed into open-ended questions concerning the operational benefits and acceptable cost of the full system as well as used as predefined categories for data analysis. Afterward, the findings of the qualitative study can be concluded in the form a guideline that describes the capabilities of the full system. In the future, if the Center for High Performance Environments (CHPE) decides to invest in the full system, the team of professional developers can use the information provided in the guidelines to estimate the cost, define the benefit threshold, and build the full system.

In conclusion, based on Groat and Wang's conceptual model of concentric frames, the overall organization of this research can be illustrated as shown in Figure 3.3. In Figure 3.3, the broadest frame of the conceptual model represents the naturalistic system of inquiry that frames the implemented combined research strategies. In this research, there are two research strategies combined using the two-phase research design. As shown in Figure 3.3, the first phase of this research adopts the logical argumentation strategy using the formal-mathematic approach. This strategy then frames the research tactic built upon the prototyping process consisting of the communication, quick planning, modeling in the form of quick design, prototype construction, and deployment, delivery, and feedback activities. The development of a prototype DSS using the prototyping process for software engineering will be discussed further in Section 3.2. Figure 3.3 also demonstrates the second phase of this research which implements the qualitative research using the interpretivism approach. This strategy then frames the research tactic, which is the qualitative research process consisting of the interview, data reduction or coding, data display, and conclusion drawing and/or verifying activities. The study on the usefulness of the prototype DSS using the qualitative research process will be discussed further in Section 3.3.

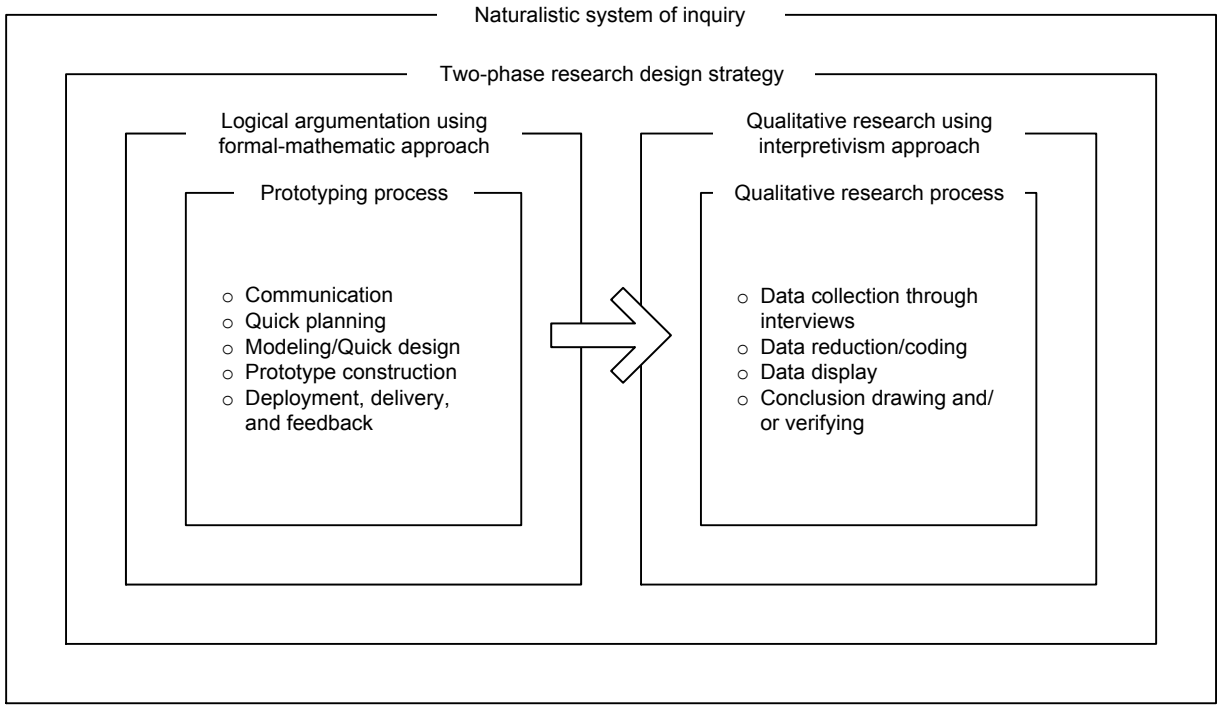


Figure 3-3: Research organization

3.2. Prototyping Process

3.2.1. Principles that Guide Software Engineering Process and Practice

As mentioned earlier, the first phase of this study adopts the prototyping software engineering process for the development of a prototype DSS for vegetated roofing system selection. This evolutionary process consists of five steps: 1) communication, 2) quick design, 3) modeling in the form of quick design, 4)

prototype construction, and 5) deployment, delivery, and feedback. Before discussing each step in detail, it is interesting to briefly discuss some fundamental principles that guide software engineering process and practice. These principles are frequently applied to almost every process and practice including the prototyping process implemented in this study.

According to Pressman (2010), it can be considered that software engineering is guided by a collection of core principles. Within the collection, there are principles that help in the application of a meaningful software process at the process level. For every software engineering process, the basic tenets of agile development should be applied, meaning that every aspect of the software engineering work should emphasize economy of action. Besides agility, the software engineering process should focus on quality at every step and be adaptable to constraints imposed by the problem, the people, and the project itself. The process should also provide effective mechanisms for facilitating communication and coordination, managing change, and assessing risk. Furthermore, the process should result in work products that provide value for other process activities, actions, and tasks. It is also interesting to note that the people who perform the software engineering work also have a significant impact on the process. Therefore, it is recommended to build an effective, self-organized, team that has mutual trust and respect.

Besides the principles that guide process, principles that help in the execution of effective software engineering methods at the level of practice are also included as part of the collection. Fundamentally, the goal of software engineering practice is to deliver on-time, high-quality, operational software that contains functions and features that serve the needs of all stakeholders (Pressman 2010). To achieve this goal, Pressman suggested that a set of principles can be applied regardless of the analysis and design method adopted in the software project. According to Pressman, divide and conquer can be considered one of most important principles for software engineering practice. This principle includes a technique known as Separation of Concern (SoC) whereby a large problem can be easier to encounter when it is subdivided into a collection of concerns or elements. While this technique establishes a philosophy for software, another technique, modularity, can be seen as a mechanism for realizing the philosophy. Modularity considers that any complex can be divided into modules in which a particular module focuses on one well-constrained aspect of the system. In addition to divide and conquer, the software engineering practice should also maintain the consistency of software elements, focus on the transfer of information, use abstractions for communication, and try to find patterns that help resolve recurring problems. In practice, it is also important to examine and represent a problem and its solutions from a number of different perspectives. Looking at the problem from different perspectives can help the software team obtain greater insight about the problem and uncover errors and omissions.

The principles described above, if they are applied appropriately throughout the software process, can help the software team produce high-quality software products that can be easy to maintain over their lifetime. These principles are often used as a basis for other principles applied for each phase of the prototyping process, which will be discussed further in detail in the following subsections.

3.2.2. Communication

The communication step of the prototyping process fundamentally concentrates on understanding stakeholders' objectives for the project and on gathering requirements that help define software features and functions (Pressman 2010). According to Pressman, effective communication can be considered one of the most challenging activities that software engineers normally confront in a software engineering project. In order to ensure that effective communication will occur, it is important to recognize the two primary stakeholders in a software project team: customer and end user. Furthermore, it is also essential to understand certain communication principles that can be used to improve the effectiveness of the communication between software engineers and other stakeholders, especially the customer and end users.

Referring to Pressman, the effective communication between software engineers and the customer and end users is extremely important, because the customer and end users have the most significant impact on the technical work in the later phases of the software engineering process. As a result, it is important to understand the nature of these two stakeholders. In small projects, the customer and end users can be the same group of people. In larger and more complex projects, the customer and end users typically represent two distinct groups of people. In order to differentiate these two project stakeholders, Pressman defined a customer as the person or group who requests objectives for the software, provides basic product requirements, and coordinates funding for the project. Pressman also defined an end user as the person or group who will actually use the software that is built to achieve some business purpose and will define operational details of the software so the business purpose can be achieved.

In addition to the discussion on the nature of the customer and end users, Pressman also discussed the basic principles for effective communication. According to Pressman, the essential principle of effective communication is to listen. For all forms of communication, software engineers should listen to the speakers' words, rather than formulating their response to those words. In addition to listening, it is also important to take notes and document any decision made during each communication. Furthermore, it is also essential to understand that there are many instances whereby software engineers and other project stakeholders may negotiate on issues such as functions and features, priorities, and delivery dates. The project team should remember that negotiation works best when all parties win; thus, it demands compromise from all parties.

Among different forms of communication, face-to-face communication can be considered the best way to communicate, but it actually works well when other representation of the relevant information such as a drawing or a straw man document is present. It is interesting to note that a sketch or a drawing can often provide clarity when verbal communication fails to communicate.

For communication in the form of project stakeholder meeting, it is important for software engineers to understand the problem, get familiar with business domain jargon, and prepare an agenda before the meeting. In a meeting, it is recommended to have a facilitator who is responsible for keeping the

conversation moving in a productive direction, mediating any conflict that does occur, and ensuring that other principles are followed. The facilitator should also keep the conversation modular by leaving one topic only after it has been resolved. As stated by Pressman, it is important to remember that a particular conversation should be moved on once the participants agree to something, if the participants cannot agree on something, and if a feature or function is unclear and cannot be clarified at the moment.

In order to promote effective communication in the DSS prototyping process that takes place in this study, it is essential to identify the key customer and end users who would like to voluntarily participate in the process and invite them to participate in the communication phase of each prototyping cycle. It can be assumed that the Center for High Performance Environments (CHPE) is the primary customer of the DSS prototyping project. For the end users, according to Grant's (2007) study discussed in the previous chapter, the potential end users of the vegetated roofing selection framework may include licensed architects, landscape architects, and roofing consultants or engineers. The end users may also be extended to include roofing contractors, horticulturists, specifiers, developers, and policy makers. In this study, a few practitioners may be invited from the CHPE affiliates, which include public sector organizations, corporations, professional societies and associations, architectural and engineering firms and school districts, to play the end user role in the prototyping process. It is important to note that, when any communication establishes in the prototyping process, the effective communication principles should be applied.

3.2.3. Quick Planning

The planning phase of the software engineering process comprises a set of management and technical practices that allow the software team to make a projection as to its strategic goal and tactical objectives (Pressman 2010). According to Pressman, the planning activity aims to help the software team manage uncertainties and reduce risks, which can be caused by a variety of issues such as unforeseen technical problems, misunderstandings, and business changes. While planning seems to have significant impacts on the quality of a software project, most of the time the software project tends to be either under planned or over planned. Therefore, it is important to understand some common planning principles that can be applied to a wide range of software projects including the DSS prototyping project discussed in this dissertation.

As stated by Pressman, a software project plan can be seen as a map that defines the software engineering work. The plan typically describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule. According to Pressman, the first thing that has to be addressed in the plan is the scope of the project. Software scope comprehensively describes software functions and features, input and output data, and content that are to be provided for end users as well as the performance, constraints, interfaces, and reliability that bound the system. The scope can be defined using a narrative description of software scope developed after communication with all stakeholders or a set of use cases developed by end

users. In addition to the software scope, Putnam and Myers (1997, cited by Pressman 2010) suggested that it is also critical to conduct a feasibility study on four areas: technology, finance, time, and resources. While the scope provides the software team with a destination, the feasibility study helps determine whether or not the team should travel toward such a destination. Pressman therefore suggested that the software project plan should clearly discuss both software scope and feasibility.

In addition to the understanding of scope and feasibility, there are other useful planning principles that should be examined. As stated by Pressman, it is important for the software project team to recognize the iterative nature of planning. The software plan should be able to reflect how changes can be accommodated flexibly and effectively. The plan should also define how the software team intends to ensure quality throughout the software project. If risks that have high impact on the project outcomes and high probability to occur are identified, contingency planning is required. When planning a project, it is important for the project team to estimate effort, cost, and task duration based on the team's current understanding of the work to be done as well as to be realistic as much as possible. In the plan, the level of detail, also referred to as granularity, should also be adjusted over time, especially when a particular set of tasks is close to its deadline. The plan should be tracked frequently in a daily basis and should be adjusted as much as it is required. It is also important to note that stakeholders should be encouraged to participate in the planning phase of the software engineering process. This is because the stakeholders normally play an important role in defining priorities and establishing project constraints. The participation of the stakeholders would allow software engineers to negotiate order of delivery, time line, and other project-related issues.

Based on the planning principles discussed above, a software project plan can be seen as an outcome of the planning activity. The plan essentially addresses the scope and feasibility of the project. For the prototype DSS developed as part of this study, it can be considered that the useful information concerning the scope and feasibility is also provided in Chapter 2.2, Chapter 2.3, and Appendix B to a certain degree. While the scope of the prototype DSS development was largely defined in Grant's dissertation document discussed in Chapter 2.3, the scope can be considerably defined by a narrative description of software scope developed after communication with all stakeholders. Moreover, the scope can also be defined by the lessons learned from previous throwaway prototypes comprehensively discussed in Appendix B.

As mentioned in Chapter 2.2, the simple but comprehensive enough feasibility report should be provided as part of the prototype DSS development. As suggested by Putnam and Mayer, such a study should concentrated on four solid dimensions of a software project and should be included as part of the project plan. Therefore, in this study, the feasibility study will be conducted within the planning phase of the prototyping process and will concentrate on the aspects of technology, finance, time, and resources involved in the development of web-based DSS. Among the four aspects of feasibility study, technology tends to be the most critical aspect of the system that will be developed. This is because the chosen

technology plays an important role in defining how much money, time, and resources required for WebApp development. The study should carefully emphasize the data exchange formats, Web 2.0 programming languages, Web application development frameworks, and Web application operational environments that effectively serve the purpose of the prototype DSS development. In addition to the technological aspect, the feasibility study should define how the prototype DSS can be developed with limited funding, time, and resources.

Although risks can be considerably low when adopting the prototyping process and developing prototype DSS within an academic environment, the development of prototype DSS still has to deal with unknown software functions and features and fuzzy requirements. Throughout the prototyping process, it can be assumed that changes can be accommodated flexibly and effectively. This is because the process allows changes to be addressed in the planning phase every time the prototype development evolves into the next cycle. Furthermore, for the overall project management, the Gantt method may be prudently used for project scheduling, progress tracking, and granularity adjustment.

3.2.4. Modeling in the Form of Quick Design

In general, models are created to gain a better understanding of the actual entity to be built. The objectives of models in software engineering work involve depicting the software from the customer's viewpoint and representing the software at a more technical level (Pressman 2010). In order to accomplish these objectives at different levels of abstraction, the basic modeling principles applied in software engineering work should be examined. According to Pressman, there are two types of models that can be created in software engineering work: requirements models and design models. Each type of model has its own modeling principles, which should also be carefully studied. Furthermore, in software engineering work, there is a broad range of software application domains whereby each of them requires specific principles for requirements modeling and design modeling. Because the whole category of web-based DSS can be seen as a set of software applications in the Web Application (WebApp) domain, it is also important to discuss some useful requirements and design modeling principles applied specifically for WebApps in this subsection.

Before discussing the principles for requirements modeling and design modeling in detail, it is necessary to examine some basic modeling principles that can be applied regardless of the types of models and software applications to be created. Above all, when creating any model, Pressman suggested that it is important to consider two goals of software engineering work: 1) to build quality software and 2) to deliver the software to the customer in a timely manner. In order to achieve these goals, models that could slow down the process or provide little new insight should be avoided. It is also recommended to create only the models that make software construction easier and faster and keep updating such models as changes occur. Because simple models tend to make the software solution easier to integrate, easier to test, and easier to maintain, it is also recommended to produce the simplest models that will describe the problems or the software solution. Besides, the models should be purposeful, be flexible to change, and

be useful rather than perfect. Occasionally, it may be necessary to adapt model notation and rules to the application or system at hand. It is important to keep in mind that the most important characteristic of the model is to communicate information that enables the next software engineering task. Moreover, every created model should be reviewed by the members of the software team to receive feedback that can be used for correcting modeling mistakes, changing misinterpretations, and adding features or functions that were inadvertently omitted. It is also interesting to note that, because sometimes software work can teach lessons on a subconscious level, it is possible for experienced software engineers to follow one's own instincts when a properly created model does not seem to be right.

As mentioned earlier, there are two classes of models in software engineering work. The first class comprises requirements models, also referred to as analysis models, which represent customer requirements. The second class encompasses design models which represent characteristics of the software that help practitioners to construct it effectively. In general, based on these different models of representation, each class of model requires its own principles for modeling. In more detail, the modeling principles for each class are quite varied depending on the software application domains. Therefore, it is essential to discuss the modeling principles for each class of models both in general and in detail with an emphasis on the WebApp domain.

Requirements Modeling Principles

Requirement models represent customer requirements by describing the software in three different domains including the information, functional, behavioral domains (Pressman 2010). The information domain comprises the data that flow into and flow out of the system and the data stores that collect and organize persistent data objects. The functional domain encompasses software functions that provide direct benefit to end users and internal support for the software features that are visible to end users. The behavioral domain involves the interactions between the software and its external environment that cause the software to behave in a specific way.

For requirements modeling, the analysis task should begin with essential information of the software viewed by the end users and then move toward implementation detail. The information domain of a problem should be well understood and represented. The functions that the software performs should also be carefully defined and described at many different levels of abstraction, ranging from a general statement of purpose to a detailed description of the processing elements that must be involved. The behavior of the software driven by its interaction with the external environment should be well represented.

In addition to the principles described above, the divide-and-conquer strategy should also be applied. In software work it is critical to divide a large, complex problem into sub problems until each sub problem is relatively easy to understand. Based on this principle, the concept of partitioning, also called the separation of concerns, derived from the divide-and-conquer strategy can be considered a key strategy in requirements modeling.

Because web-based DSS can be seen as software applications in the Web application domain, it is important to understand the methods for requirements modeling that are specific for WebApps. According to Pressman, the demand for requirements modeling for WebApps depends on five factors. The first factor is the size and complexity of WebApp increment. The second factor is the number of stakeholders. The third factor is the size of the WebApp team. The fourth factor is the degree to which members of the WebApp team have worked together before. The last factor is the degree to which the organization's success is directly dependent on the success of the WebApp. It is interesting to note that, except for the fourth factor, the larger quantity in size, number, or degree indicates the higher demand for requirements modeling. For the fourth factor, the larger degree to which the WebApp team members have worked together in previous projects indicates the smaller demand for requirements modeling.

Based on the factors described above, it can be considered that the degree to which requirements modeling should be emphasized in this study tend to be small in almost all factors. Therefore, a lightweight requirements modeling or analysis approach can be applied. As stated by Pressman, the lightweight analysis approach assumes that the design of a specific part of the WebApp only demands an analysis of those requirements that affect only that part of the WebApp.

In order to effectively create requirement models for a WebApp, it is important to understand the input and output for requirements modeling. According to Pressman, when a WebApp is engineered, an agile version of the generic software engineering processes, including the prototyping process, can be applied. Before the modeling phase begins, the communication phase typically allows the software team to identify stakeholders and end user categories, the business context, defined informational and applicative goals, general WebApp requirements, and usage scenario. Such information can be represented in the form of natural language descriptions, rough outlines, sketches, and other informal representations and used as inputs for requirements modeling.

Based on the input information, requirements modeling serves as a disciplined mechanism for representing and evaluating WebApp content and function, the modes of interaction that users will encounter, and the environment and infrastructure in which the WebApp resides (Pressman 2010). As suggested by Pressman, each of these characteristics can be represented using five major classes of models: 1) the content model, 2) the interaction model, 3) the functional model, 4) the navigation model, and 5) the configuration model. These models enable the software team to analyze WebApp requirements in a more structured manner and can be seen as the requirements modeling outputs.

For WebApp requirements modeling, content models are used to determine a full range of content to be provided by the WebApp including text, graphics and images, videos, and audio data. According to Pressman, the content model contains structural elements including content objects and all analysis classes. A content object can be anything from, for example, a textual description, a news article, a photograph or an image, a users' discussion forum response, an animated corporate logo, a short video, or an audio clip. An analysis class is a user-visible entity that can be created and manipulated when a

user interacts with the WebApp. Content models can be represented using diagrams such as the data tree.

Typically, it can be considered that the majority of WebApps comprises a conversation between an end user and application functionality, content, and behavior (Pressman 2010). Such a conversion can be represented using an interaction model. The interaction model can be composed of use cases, sequence diagrams, state diagrams and/or user interface prototypes. Normally at the analysis level, a set of use cases is sufficient for describing the interaction between an end user and WebApp functionality. For more sophisticated analyses which involve multiple analysis classes or many tasks, a set of sequence diagrams and/or a set of state diagrams can be used to depict the complex interaction. For the consistency and effectiveness of models, the Object Management Group's Unified Modeling Language (OMG's UML) specifications can be used as a basis for creating the use cases, sequence diagrams, and state diagrams. The UML specifications can be found from the OMG's website at www.uml.org. Besides these three models, it is also recommended for software engineers to produce a set of interface prototypes. Because the tools for constructing WebApps are relatively inexpensive and functionally powerful, it is feasible to create the interface prototypes that implement the major navigational links and represent the overall screen layout in very much the same way that it will be constructed.

More and more WebApps tend to provide not only content in a variety of formats, but also a broad array of computational and manipulative functions that can be associated with the content. As a result, it is important to analyze the functional requirements of any WebApp to be developed. According to Pressman, functional models address two processing elements of the WebApp whereby each element represents a different level of procedural abstraction. The first element is the user-observable functionality that is delivered by the WebApp to end users. The user-observable functionality encompasses any processing functions that are originally initiated by the end users. The second element comprises the operations contained within analysis classes that implement behaviors associated with the class. These operations handle class attributes and are involved as classes collaborate with one another to achieve some required behavior. Both processing elements can be modeled using the UML activity diagrams.

As part of today's WebApps, the navigation system plays an important role in assisting WebApp users to find what they want from a WebApp. According to Pressman, the navigation strategy can be defined using navigation models. Pressman suggested that the navigation modeling in the requirements modeling stage should concentrate on overall navigation requirements, because the mechanics of navigation are defined as part of design. The process for navigation requirements analysis or modeling mainly involves asking and answering questions considering how each user category will navigate from one WebApp element to another. For example, the following question may be asked as part of the process: "Should certain elements require fewer navigation steps to reach than others?". It is important to note that the project stakeholders should be involved in the process to help determine overall requirements for navigation.

Besides the four classes of requirement models discussed above, the configuration models should also be developed to describe the environment and infrastructure in which the WebApp resides (Pressman 2010). In small WebApps, the configuration model may provide only a list of server-side and client-side attributes which may not have a significant impact on analysis and design. In larger and more complex WebApps, a variety of configuration complexities such as distributing load among multiple servers, caching architectures, and remote databases may have a significant impact on analysis and design. According to Pressman, when complex configuration architectures must be considered, the UML deployment diagrams can be used for configuration modeling.

Design Modeling Principles

In addition to the principles for requirements modeling, it is also important to examine the principles for design modeling. As stated by Pressman, the software design modeling begins by representing the software as a whole then gradually refining the software to provide guidance for constructing each detail. Based on this principle, it is important to ensure that the elements of the design model can be traced back to the requirements models. Fundamentally, the design model translates customer's requirements depicted by requirements models into software architecture. Software architecture is a set of subsystems that implement major functions and a set of components that represent the realization of requirements classes. Because software architecture can be considered the skeleton of the system to be built (Pressman 2010), design should essentially start with the architectural aspects of the software. Afterward, the design should move further toward the component-level issues only after the architecture has been established.

For architectural design, data design can be considered the most essential design element. According to Pressman, a well-structured data design can enormously help simplify program flow, overcome the difficulty of software component design, and increase the overall processing efficiency of the system. In addition to data design, the design of interfaces, both internal and external, should also be taken into consideration. Interface design fundamentally emphasizes the needs of end users and ease of use. A well-designed interface can make integration easier and assist the tester in validating component functions. In contrast, a poor-designed interface can give end users a bad impression about the software.

Besides data design and interface design, component-level design considers the functionality that is delivered by a software component. According to Pressman, the component-level design should be functionally independent. The functionality provided by a component should be cohesive, meaning that it should concentrates on one and only one function or sub function. As part of component-level design, it is important to ensure that components are loosely coupled to one another as well as to the external environment. Above all, design should be developed iteratively whereby each time the design should be simpler than the one before it. Furthermore, design models resulting from the architectural design, data design, interface design, and component-level design should be easy to understand.

Similar to many other software domains, the quality of WebApps is introduced during design. According to Pressman, the WebApp quality can be defined in terms of usability, functionality, reliability, efficiency, maintainability, security, scalability, and time-to-market. Based on such terms, Figure 3-4 summarizes the quality measures suggested by Olsina et al. (1999, cited by Pressman 2010) and Offutt (2002, cited by Pressman2010). In order to attain the quality presented in Figure 3-4, Kaiser (2002, cited by Pressman 2010) suggested a set of design goals that are applicable to virtually every WebApp. The design goals include simplicity, consistency, identity, robustness, navigability, visual appeal, and compatibility. In order to achieve these design goals, Pressman suggested that the WebApp design activity should fundamentally concentrates on six different elements of the design which can be depicted by a design pyramid for WebApps shown in Figure 3-5. In Figure 3-5, each element in the pyramid responds to a particular aspect of a WebApp and has its own methods for modeling. Accordingly, it is necessary to examine the modeling principles for the six WebApp design elements presented in Figure 3-5.

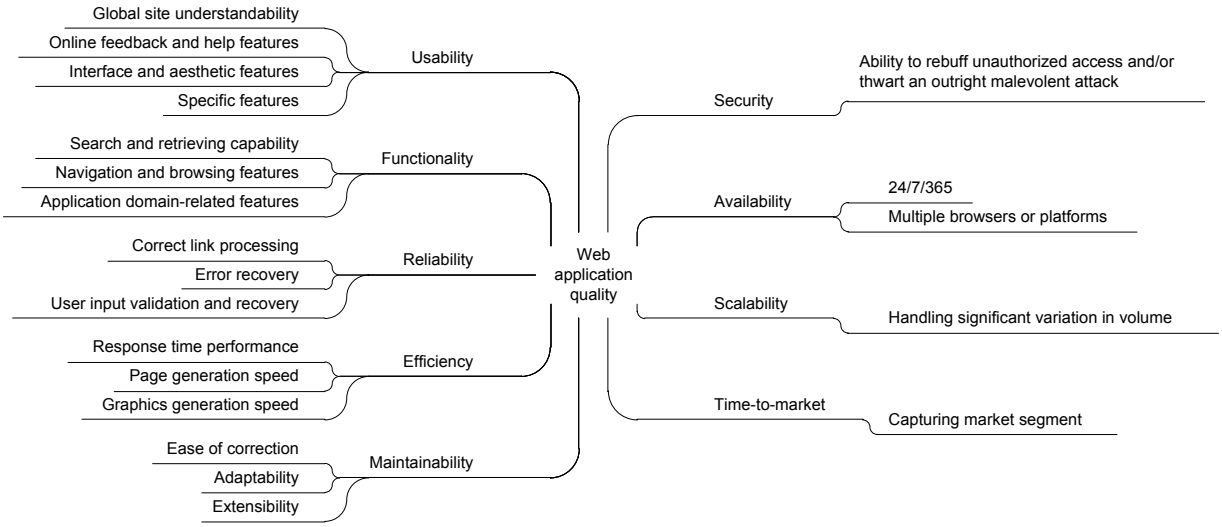


Figure 3-4: Quality requirements, adapted from Pressman (2010)

Interface design for WebApps predominantly concentrates on three objectives of a WebApp interface (Pressman 2010). The first objective is to establish a consistent window into the content and functionality provided by the interface. The consistent interface can be achieved using aesthetic design that emphasizes the layout and form of navigation mechanism. The second object is to guide the user through a series of interactions with the WebApps. For this objective, an appropriate metaphor can be drawn to guide user interaction. The interaction metaphor helps enable the user to gain an intuitive understanding of the interface. The metaphor may include appropriate images and concepts from the users' experiences, but it does not have to be an exact reproduction of a real world experience. The last object is to organize the navigation options and content available to the user. According to Pressman, the navigation options can be selected from one of the following mechanisms: 1) navigation menus, either vertically or horizontally organized; 2) graphic icons such as button and switches; and 3) graphic images

in which each image can be selected by the user and can implement a link to a content object or WebApp functionality.

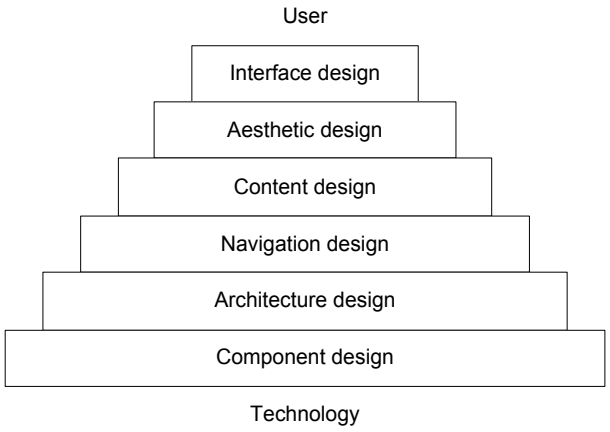


Figure 3-5: Design pyramid for WebApps, adapted from Pressman (2010)

As mentioned earlier, aesthetic design helps achieve the consistency of the interface. As stated by Pressman, aesthetic design, also referred to as graphic design, is an artistic effort that enhances the technical aspects of WebApp design. In order to be successful with aesthetic design, it is important to recognize the WebApp users and what would be the appropriate aesthetic impression for such users. In general, aesthetic design emphasizes two categories of issues: layout and graphic design. For the layout, a web page can take advantage of the white space. The webpage should emphasize its content which typically account for 80 percent of the area on the page. On the Web page, layout elements should be organized from top-left to bottom-right based on how the vast majority of users will scan a Web page. In addition, navigation, content, and function should be grouped within the Web page, and the use of the scrolling bar should be limited. When designing layout, it is also important to take into consideration the solution and browser window size. From layout, the aesthetic design process moves toward graphic design. Graphic design focuses on global color schemes; text types, sizes, and styles; the use of supplementary media; and all other aesthetic elements of an application. Fundamentally, aesthetic design can be presented using interface prototypes used in requirements modeling with the higher level of refinement.

Content design concentrates on the different design tasks mainly conducted by two groups of individuals who have different skill sets (Pressman 2010). The first task, which belongs to content designers, involves creating a design representation for content objects and the mechanisms required to establish their relationship to one another. The second task, which belongs to content authors, involves generating information within a particular content object. As stated by Pressman, a single Web page is typically formulated by a number of content objects and their relationships to each other. The number of contents is a function of user needs, constraints caused by download speed or the internet connection, and restrictions imposed by the tolerable amount of scrolling. Content objects and their relationships can be

represented by the UML association and an aggregation. After the models of content objects are created, the information that each object is to deliver must be authored and formatted to meet the customer's needs.

As part of WebApp design, architecture design aims to define content architecture and WebApp architecture. As stated by Pressman, content architecture emphasizes the system in which content objects are structured for presentation and navigation. In most cases, content architecture can be designed based on four different content structures or models: 1) linear structure, 2) grid architecture, 3) hierarchical structure, and 4) network architecture. WebApp architecture focuses on the system whereby the application is structured to manage user interaction, handle internal processing tasks, effect optimal navigation for users, and present content. Pressman discussed in his book a WebApp architecture known as the Model-View-Controller (MVC) originally suggested by Krasner and Pope (1988). The MVC can be seen as one variation of the multi-tier architecture which is widely used as a basis for WebApp architecture design. For WebApps in the form of web-based DSS, the web-based DSS architectures also built up on the multi-tier architecture, such as those suggested by Power (2000) and Turban et al. (2007), can be used as a basis for architecture design.

Navigation design considers the semantics of navigation for different users of the site and the mechanics or syntax of achieving the navigation (Pressman 2010). When interacting with a WebApp, a particular user typically encounters a series of Navigation Semantic Units (NSU). Cachero et al. (2002, cited by Pressman 2010) defined the NSU as a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements. An NSU composes of a set of navigation elements referred to as Ways of Navigation (WoN) suggested by Gnaho and Larcher (1999, cited by Pressman 2010). According to Pressman, a WoN is a navigation design model that represents the best navigation pathway to achieve a navigational goal for a specific group of user. Besides the semantics of navigation, the navigation syntax can be defined by a number of opportunities including a horizontal navigation bar, a vertical navigation column, individual navigation links, tabs, and sitemaps. In addition to the opportunities as such, it is also useful to establish appropriate navigation conventions and aids to make navigation more user friendly.

According to Pressman, it can be considered that modern WebApps increasingly provide sophisticated processing functions like those delivered by traditional software. In order to effectively provide such sophisticated functions, Pressman suggested that it is important to design and construct program components that are identical in form to those software components found in traditional software. Because the components of WebApps and traditional software applications are quite similar in form, the traditional component design methods can be applied. However, the component design methods for WebApps can be varied depending on the implementation environments, programming languages, design patterns, and frameworks.

In summary, the modeling phase can be considered one of the most important phases in the prototyping process. This is because the quality of software is introduced during this phase of the process. The modeling activity begins with requirements modeling then moves toward design modeling. For the prototyping process discussed as part of this study, the principles for requirements modeling and design modeling discussed above and the OMG's UML specifications and guidelines will be used as a basis for modeling. For this study, the modeling activity can be seen as an audit trial activity of naturalistic research. The requirements and design models will be created when necessary, largely in the form of simple and informative models. The models are also considered the most important documents for demonstrating dependability involved within this phase of the research. At the end of this study, if the prototype is proved to be useful for the end users in supporting their decisions concerning the vegetated roofing system selection, the finalized models can be further consolidated in the form of a guideline for the development of the fully functional DSS that will be developed in the near future.

3.2.5. Prototype Construction

As part of the software engineering process, the construction phase composes of manual or automated code generation and the testing that is required to uncover errors in the code (Pressman 2010). Code generation or coding can be one of the direct creations of programing language source code such as Java, the automatic generation of source code using an intermediate design-like representation of the component to be built, or the automatic generation of executable code utilizing a fourth-generation programming language such as Visual C++. Testing composes of different levels of activities including unit testing, integration testing, and acceptance testing. As stated by Pressman, coding and testing tasks in combination result in operational software that is ready for delivery to the customer or end users. Therefore, it is important to understand certain principles and concepts that are applicable to coding and testing.

Coding Principles and Concepts

According to Pressman, the principles that guide the coding task can be diverse because of the implemented programming styles, languages, and methods. However, there are a number of fundamental principles which can be categorized into three categories: preparation, programming, and validation. The preparation principles are applied before even one line of code is written. The coding preparation generally involves understanding the encountered problems and understanding basic design principles and concepts. The coding preparation also involves selecting appropriate programming language and environment. The appropriate programming language must be able to serve both the needs of the software to be built and its operational environment. The appropriate programming environment should provide tools that make the programming work easier and faster. During the coding preparation, it is possible and encouraged to create a set of unit tests for component testing that will be applied after each component is completely constructed.

Once the coding task is started, certain principles for programming can be applied. From the programming standpoint, the structured programming practice should be adopted to constrain program algorithms, and the pair programing practice should be used if applicable. From the data standpoint, the programming principle may include choosing data structures that will fulfill the requirements of the design. From the architecture standpoint, the principles may involve understanding the overall architecture of the software and creating the interfaces that are consistent with the architecture. From the technical standpoint, the principles may include keeping conditional logic simple and creating easily testable nested loops. Furthermore, for good practice, it is recommended to choose meaningful variable names, to follow other adopted local coding standards, and to write the code that is self-documenting. It is also useful to create a visual layout, such as indentation and blank lines. The visual layout often helps the written code to be easier to read and understand.

After the coding work is completed, it must be validated. The validation principles typically involve conducting a code walkthrough when appropriate and refactoring the code. Furthermore, it is also critical to perform unit tests and try to correct the uncovered errors.

Testing Principles and Concepts

Besides the coding principles, Pressman also discussed a set of testing rules suggested by Myers (1979, cited by Pressman 2010), certain principles that guide the testing task suggested by Davis (1995, cited by Pressman 2010), and some principles and concepts that can be specifically applied to WebApp testing. According to Myers, testing can be seen as a process of executing a program, aiming to find an error. Myers suggested that a "good test" case is one that has a higher probability of finding an as-yet-undiscovered error. Myers also stated that a "successful test" is one that uncovers an as-yet-undiscovered error. Derived from these rules, Pressman suggested that the main objective of testing is to design tests that can systematically uncover diverse classes of errors with a minimum amount of time and effort. Pressman also suggested that testing will uncover errors in the software if it is conducted successfully. Nevertheless, it should be kept in mind that testing cannot show the absence of errors and defects. Testing can show only the software errors and defects that are present.

In addition to the rules and principles discussed above, Pressman also suggested certain useful principles for testing derived from a set of testing principles suggested by Davis. As stated by Pressman, it is important to consider that the most serious defects are those that make the program unable to meet its requirements. Therefore, all tests, especially the functional tests that directly focus on requirement, should be traceable to customer requirements. As mentioned earlier, all tests can be planned and designed as part of the coding preparation before any code is generated. Testing should initially focus on finding errors in individual components then extend to find errors in integrated clusters of components and ultimately in the entire system. However, it is important to note that it is impossible to conduct exhaustive testing. Testing should try to cover program logic and ensure that all conditions in the component-level design are exercised. Besides these principles, Pressman suggested that the Pareto principle should also

be applied. This principle considers that eighty percent of all errors found during testing will likely be traceable to twenty percent of all program components. According to Pressman, it is important to isolate these error-prone components and extensively test them.

Because of their unique nature, WebApps require a specific set of testing activities. According to Pressman, a single goal of WebApp testing is to uncover errors in the following areas: 1) content, 2) function, 3) usability, 4) navigation, 5) performance, 6) capacity, and 7) security. In order to achieve this goal, it is essential to understand certain characteristics of errors that can be found within a WebApp environment, a testing strategy and its associated tactics, a test planning approach, and a WebApp testing process.

As suggested by Nguyen (2000, cited by Pressman 2010), errors that can be found within a WebApp environment have certain unique characteristics, which makes them difficult to be uncovered, traced, and reproduced. An error within a WebApp environment is difficult to be uncovered, because a problem that is likely to be uncovered by one of WebApp tests, especially the one initially evidenced on the client side, could only be one symptom of the error rather than the error itself. While the symptoms of an error can easily be uncovered via different WebApp tests, the error that causes such symptoms is quite difficult to be traced. This is because WebApps typically reside within a multi-tier architecture, which makes the error difficult to be traced across different architecture layers such as the client, the server, or the network. Besides the different architecture layers, most of the WebApps are commonly operated within two operating environments: static and dynamic; therefore, it is difficult to determine which operating environment would exactly be the one in which the error resides. In addition to the diverse environments, the error may be attributable to the WebApp configuration which has considerable potential to be overlooked. Furthermore, for the reason that a WebApp is usually implemented within diverse environments and in a number of different configurations, it is also quite difficult to reproduce an error outside the environment wherein the error was originally found.

Frequently, the characteristics of errors encountered during WebApp testing are different from those often found during conventional software testing. Therefore, it is important to examine a strategy and tactics that have been specifically recommended for WebApp testing. According to Pressman, the strategy and tactics for WebApp testing can be derived from the basic principles for conventional software testing and for object-oriented system testing. The strategy for WebApp testing aims to uncover and correct errors before the WebApp becomes available to its end users using both technical reviews and executable tests.

In order to achieve the strategic goal discussed above, certain testing tactics should be applied. For WebApp testing, technical reviews can be conducted to uncover errors as soon as requirements and design models are created. Technical reviews may be applied to uncover errors in the content model, ensure that all use cases are accommodated in the interface model, and uncover navigation errors in the design model for the WebApp. After the WebApp is constructed, executable tests can be applied. Executable tests may be conducted to uncover errors in the presentation and/or navigation mechanics of

the user interface as well as to test the navigation throughout the architecture, environmental configuration compatibility, security, and performance of the WebApp. Furthermore, at the component level, unit tests may also be conducted to uncover errors within the functional components of the WebApp.

As a whole, the WebApp can be tested by a controlled and monitored population of end users whereby the results of the interactions between the WebApp and its end users are evaluated for content and navigation errors, usability concerns, compatibility concerns, and Web security, reliability, and performance. It is interesting to note that that the WebApp testing process can be considered an ongoing activity, because many WebApps tend to evolve repeatedly. Throughout WebApp lifecycle, the testing process is frequently conducted by Web support staff who employ regression tests derived from the tests developed when the WebApp was initially engineered (Pressman 2010).

As mentioned earlier as part of the general principles for software testing, a test plan can be initially developed before any code is written. As recommended by Pressman, whether the test plan is included as part of the project plan or prepared separately, it should address three issues relating to WebApp testing. First, it should identify the task set to be applied when testing begins. Second, it should define the work products to be produced when each testing task is executed. Lastly, it should also identify how the results of testing will be evaluated, recorded, and reused when regression testing is conducted.

Besides the characteristics of errors, the testing strategy and tactics, and the test planning approach discussed above, it is also essential to examine a process for WebApp testing. According to Pressman, the WebApp testing process consists of seven steps: 1) content testing, 2) interface testing, 3) navigation testing, 4) component testing, 5) configuration testing, 6) performance testing, and 7) security testing. This seven-step testing process can be mapped onto the design pyramid for WebApps as shown in Figure 3-6. In Figure 3-6, each step of the process emphasizes a particular issue relating to WebApp testing which will be discussed further in detail.

The WebApp testing process shown in Figure 3-6 begins with content testing then flows from left to right and from top to bottom. According to Pressman, content testing applies technical reviews to uncover semantic errors and executable test cases to uncover content errors. Such content errors are traceable to dynamically derived content that is driven by data acquired from one or more databases. This testing step aims to achieve three important objectives. The first objective is to uncover syntactic errors in text-based documents, graphical representations, and other media. This objective may be fulfilled by employing automated spelling and grammar checkers to uncover detectable syntactic errors. For better results, the content should be reviewed by a human reviewer, such as a professional copy editor, to uncover typographical errors, grammatical mistakes, errors in content consistency, errors in graphical representations, and cross-referencing errors. The second objective of content testing is to uncover semantic errors in any content object presented as navigation occurs. This objective can be achieved by asking a reviewer to review the information presented within each content object. The last objective of

content testing is to find errors in the organization or structure of content that is presented to the end user. This objective can be accomplished by testing the structure and organization of the content architecture to ensure that required content is delivered to the end user in the appropriate order and relationships.

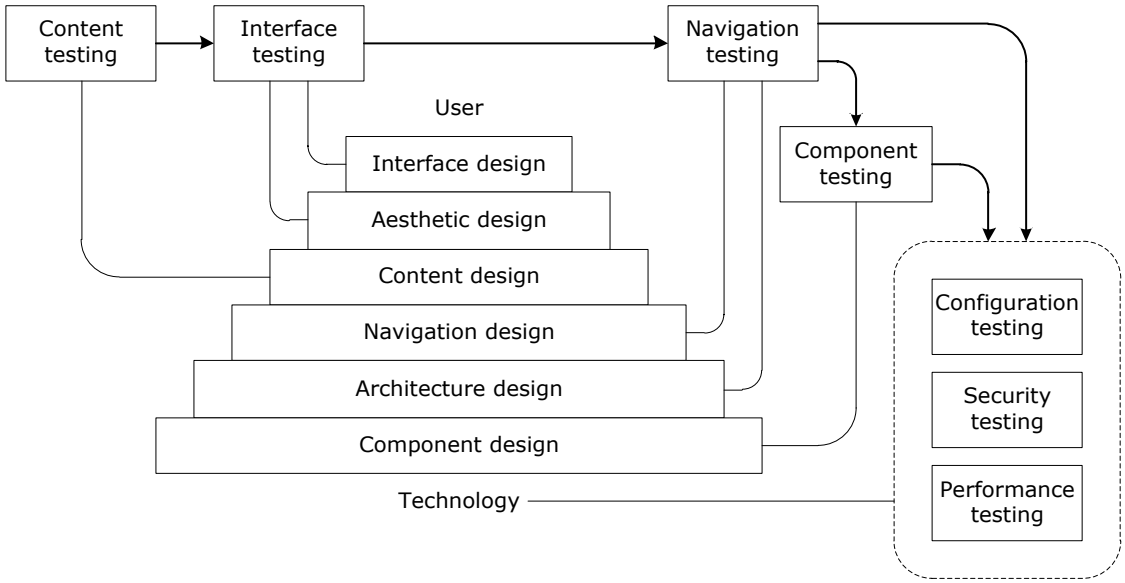


Figure 3-6: Testing process for WebApps, adapted from Pressman (2010)

Because modern WebApps often interface with sophisticated database management systems and generate dynamic content objects that are created in real time utilizing the data acquired from a database, database testing should also be conducted as part of content testing. Basically, Nguyen (2001, cited by Pressman 2010) suggested that the data used to create dynamic content objects can be acquired through five different layers of interaction. The five layers of interaction can be sequentially listed as follows: 1) user interface layer, 2) WebApp layer, 3) data transformation layer, 4) data management layer, and 5) data access layer. Referring to Pressman, test-case design methods can be applied for database testing, focusing on each interaction between these layers. First, testing should ensure that valid information is passed between the user interface layer on the client side and the WebApp layer on the server side. Second, testing should make sure that the WebApp layer processes scripts correctly and properly extracts or formats user data. Third, testing should ensure that user data are passed correctly to a server-side data transformation function that format appropriate queries. Finally, testing should make sure that queries are passed to a data management layer that interacts with database access routines. According to Pressman, the data transformation, data management, and database access layers are frequently constructed using reusable components that have been validated separately and as a package. Therefore, database testing for WebApps often focuses on the design of test cases to exercise the interactions between the user interface layer on the client side and the WebApp and data transformation layers on the server side.

Continuing from content testing, user interface testing emphasizes the execution of application-specific aspects of user interaction as they are revealed by interface syntax and semantics (Pressman 2010). As stated by Pressman, user interface testing has two strategic goals. First, it aims to uncover errors related to specific interface mechanisms, for example, errors in the execution of a menu link. Second, it intends to uncover errors in the way that the interface implements the semantics of navigation, WebApp functionality, or content display. Pressman suggested that these goals can be achieved by certain tactics. First, tests on interface features should be conducted to ensure that design rules, aesthetics, and related visual content are available for the end user without error. Second, tests on individual interface mechanisms should be performed in a way that is analogous to unit testing. Third, tests on each interface mechanism should be conducted within the context of a use case or navigation semantic unit for a specific user category. Fourth, tests on the complete interface should be performed against selected use cases and navigation semantic units to uncover errors in the semantics of interface. Lastly, tests on the interface should also be tested within a variety of environments such as different Web browsers and devices to ensure that it will be compatible with such environments.

Based on the strategy and tactics discussed above, user interface testing can be divided into four activities: 1) interface mechanisms testing, 2) interface semantics testing, 3) usability testing, and 4) compatibility testing. Interface mechanisms testing is applied to uncover errors associated with individual interface mechanisms using tests similar to unit tests and integration tests. The mechanisms needed to be tested may include links, forms, client-side scripts, dynamic HTML, pop-up windows, CGI scripts, streaming content, cookies, and application-specific interface mechanisms. Interface semantics testing is applied to evaluate how well the design responds to end users and how well it provides clear direction, feedback, and consistent language and approach. Interface semantics can be tested against a series of use cases. Each case essentially becomes an input for the design of a testing sequence which aims to uncover errors that will preclude a user from achieving the objective associated with the use case. Usability testing is applied to determine the degree to which the WebApp interface is easy to use as perceived by end users. Usability testing can happen at a variety of different levels of abstraction from a specific interface mechanism to a complete Web page to the complete WebApp. Fundamentally, a series of usability tests are often designed within the following usability categories: interactivity, layout, readability, aesthetics, display characteristics, time sensitivity, personalization, and accessibility. Compatibility testing is applied to uncover errors or execution problems that can be traced to computing configuration differences between computers, display devices, operating systems, browsers, and network connection speeds, and so on. Compatibility testing begins with defining a set of commonly encountered client-side computing configurations and their variants. Afterward, the testing activity continues on deriving a series of compatibility validation tests, frequently adapted from existing interface, navigation, performance, and security tests.

The next step of WebApp testing is to conduct navigation testing. According to Pressman (2010), the objectives of navigation testing are to ensure that the mechanisms that allow the WebApp user to travel

through the WebApp are functional and to validate that a particular semantic unit can be achieved by the appropriate user category. These objectives can be achieved by two testing activities: navigation syntax testing and navigation semantics testing. Navigation syntax testing is applied to test navigation mechanisms to make sure that each mechanism performs its intended function. The mechanisms needed to be tested may include navigation links, redirects, bookmarks, frames and framesets, site maps, and internal search engines. It is interesting to note that navigation testing often begins during interface testing when interface mechanisms are tested. Navigation semantics testing is applied to test a particular navigation semantic unit to ensure that navigation requirements defined by one or more use cases for a user category are fulfilled. Navigation semantics testing is usually conducted by software engineers during early stages of navigation testing then the responsibility is passed on to other project stakeholders, and ultimately, to nontechnical users in the later stages of navigation testing.

Continuing from navigation testing, component-level testing, also referred to as function testing, can be applied to uncover errors in WebApp functions using black-box and white-box techniques (Pressman 2010). In WebApp component-level testing, test cases are frequently derived by forms-level input. According to Pressman, three test-case design methods can be applied: equivalence partitioning, boundary value analysis, and path testing. Equivalence partitioning method divides the input domain of the function into input categories or classes from which test cases are derived. In this manner, the input form can be assessed to determine which classes of data are relevant for the function. This method allows test cases for each class of input to be derived and executed, while other classes of input remain constant. Boundary value analysis method tests forms data against their boundaries to ensure that the function properly reacts to data at and outside the boundaries of valid input. Besides equivalence partitioning and boundary value analysis, the path testing method can be applied to make sure that an individual path in the program is exercised, especially when the logical complexity of the function is high. In addition to these design methods, a technique, known as forced error testing suggested by Nguyen (2001, cited by Pressman 2010) can also be used to derive test cases that purposely drive the WebApp component into an error condition. The objective of this testing is to uncover errors that occur during error handling. Furthermore, because many WebApps are often interfaces with external databases, it is also recommended to include database testing as part of component-level testing.

After content, interface functionality, architectural design, and navigation are examined, the focus of the WebApp testing process shifts to tests that exercise technological capacities that are not always apparent to end users (Pressman 2010). The technological capacity tests typically include configuration testing, performance testing, and security testing. Configuration testing involves exercising a set of probable client-side and server-side configurations to ensure that the user will have similar experience across different configurations. The testing also involves isolating errors that may occur specifically when a particular configuration is applied. On the server side, configuration test cases are normally developed to ensure that the projected server configuration allows the WebApp to operate without errors. On the client side, configuration tests concentrate more on configuration that contain one or more permutations of

hardware, operating systems, browser software, user interface components, plug-ins, and connectivity (Nguyen 2001, cited by Pressman 2010). In addition to these components, Pressman also added that configurations of networking software, vagaries of the ISP, and applications running concurrently on the client side should also be taken into consideration.

As part of technical capacities testing, security testing aims to uncover weaknesses that can be exploited by those who wish to steal sensitive information, maliciously modify content, degrade performance, disable functionality, or embarrass a person, organization, or business. According to Pressman, security tests are designed to probe vulnerabilities of the client-side environment, the network communications, and the server-side environment. On the client side, vulnerabilities are frequently traced to preexisting bugs in browsers, e-mail programs, or communication software and to unauthorized access to cookies placed within the browser. For network communications, spoofing—which occurs when one end of the communication pathway is subverted by an entity with malicious intent—can be considered the most significant vulnerability. On the server side, vulnerabilities may include denial-of-service attacks, malicious scripts that can be passed through the client side or used to disable server, and unauthorized access to server-side databases. In order to protect against these vulnerabilities, Nguyen suggested that one or more of the security elements including firewall, authentication, encryption, and authorization must be implemented. Nevertheless, because the actual design of security tests requires in-depth knowledge of security elements and networking technologies, security testing in practice tends to be outsourced to the specialized firms (Pressman 2010).

In addition to configuration testing and security testing, performance testing is applied to uncover performance problems that can result from various causes such as lack of server-side resources, inappropriate network bandwidth, and inadequate database capabilities, and so on. As stated by Pressman, performance testing has two objectives. The first objective is to understand how the system responds as loading increases. The second objective is to collect metrics that will lead to design modifications to improve performance. To accomplish these objectives, two different performance tests can be applied: load testing and stress testing. According to Pressman, load testing is applied to examine real-world loading at a variety of load levels and in a variety of combinations and stress testing is applied to force loading to be increased to the breaking point to determine the capacity that the WebApp environment can handle.

To summarize, the information provided above can be seen as a practical guideline for the construction phase of the prototyping process discussed in this study. The construction phase essentially is composed of two tasks: coding and testing. For coding, this study aims to apply the coding principles discussed at the beginning of this subsection together with principles and standards that are specific for the programming language and environment implemented for the development of the prototype DSS. For testing, it is necessary to consider that the development of the prototype DSS adopts the prototyping process which concentrates more on identifying the requirements of the system rather than the quality. In

this study, the testing process therefore will be applied to ensure that the prototype DSS properly represents the web-based DSS for vegetated roofing system selection that would be fully developed with an eye toward quality and delivered to the end users in the near future.

3.2.6. Deployment, Delivery, and Feedback

Deployment is the last activity both in an incremental software engineering process before a new software increment is developed and in an evolutionary process before the process evolves into a new development cycle. Referring to Pressman, the deployment activity comprises three actions: delivery, support, and feedback. The delivery action provides an operational software increment that offers usable functions and features for the customer and end users. The support action supplies documentation and human assistance for all functions and features introduced during all deployment cycles up to the current cycle. The feedback action gives the software team the strategic direction for modifying the functions, features, and approach taken for the next increment. In order to effectively deliver a software package to the customer or end users, some basic principles that guide the deployment activity should be examined.

According to Pressman, the first principle is to manage customer expectations for the software. This principle aims to ensure that the software to be delivered is not under or over user expectations. The key is to carefully communicate with the customer or end users about what they should expect from the delivered software. The second principle is to make sure that a complete delivery package is well assembled and tested. It is important to provide the customer and end users with an operational software package that contains all executable software, support data files, support documents and other relevant information. The third principle is to establish a support system before the software is delivered. This may include developing a support plan, preparing support materials, and establishing appropriate record keeping mechanisms. The fourth principle is to provide appropriate instructional materials for the customer or end users. The materials may include training aids, troubleshooting guidelines, and a brief description of the differences between the current and previous increments of the software. The last principle is to deliver a high-quality product. It is critical to make certain that bugs are fixed before the software is delivered.

For this study, based on the principles discussed above, the research participants should be informed that the web-based DSS that they are going to experience is a prototype version of the DSS. By its very nature, the prototype may have limited functions and features, may contain uncovered errors, and may be discarded after system requirements are well understood. However, to ensure that the prototype properly represents the web-based DSS that would be fully developed in the near future, the prototype should be designed and built with an appropriate level of quality and should be well assembled and tested. Besides, the support system and instructional materials should also be appropriately provided. Furthermore, because the goal of the prototyping process is to determine what would be the appropriate functions and features, ease of use, reliability and other characteristics of software, the participants should be encouraged to give feedback on the characteristics of the prototype.

In addition to the discussed principles, when it comes to the development of DSS, it is essential to note that most of DSS projects are typically driven by "demand pull" or "visible, concrete needs" rather than "technology push" (Keen 1980). Therefore, the users' feedback obtained during this phase of the prototyping process tends to have considerable potential to help increase the success rate of the prototype DSS. During this phase, the users should be encouraged to participate in alpha tests if applicable. According to Pressman (2010), an alpha test is a set of test activities conducted on the developer's site by the users under controlled environments. Alpha tests frequently use observations as a technique to understand the problems encountered by the users during the time of use. The problems observed during the alpha test then can be brought to and tackled in the next prototyping cycle.

Once the prototype DSS is well developed, the research emphasis can shift from the prototype DSS development to the qualitative study on the usefulness of the prototype. From the software engineering standpoint, the qualitative study conducted as part of this study is quite similar to the beta test described by Pressman (2010), also referred to as an acceptance test. These two approaches are similar in the way in which the software application will be delivered to the users' sites, which allows the users to use the software application in open environments with high degrees of freedom. The difference between these two approaches is that the qualitative study is conducted using in-depth interviews whereas the conventional beta test is conducted using a customer report mechanism to understand the benefits and problems of the software application being studied. For this study, the understanding gained from the qualitative study is used to developed a guideline that can be used as a basis for the development of the full version DSS. The process for qualitative study conducted as part of this study is extensively discussed in the next section of this chapter.

3.3. Qualitative Research Process

3.3.1. Data Collection

3.3.1.1. Data Collection Techniques

Referring to the research organization discussed in Chapter 3.1, the second phase of this research implements the qualitative research process suggested by Groat and Wang (2002) to study the usefulness of the prototype DSS. This process begins with the data collection phase in which the three data-gathering techniques suggested by Rossman and Rallis (2003) can be applied: interviewing, observing, and documenting material culture. According to Rossman and Rallis, the use of data-gathering techniques primarily depends on a researcher's interests, a researcher's assumptions about epistemology and the social world, project feasibility, and research strategy, genre, and ethics. Besides these aspects of research, Rossman and Rallis suggested that the use of data-gathering techniques will also depends on the levels of depth, openness, and flow in which the techniques will be applied. In order to provide comprehensive information about the data collection activity performed in this study, this subsection discusses both the major decisions that guide the implementation of data-gathering techniques and the techniques implemented to collect data regarding the usefulness of the prototype DSS in detail.

According to Rossman and Rallis, selecting data-gathering techniques involves various decisions in which each decision focuses on a particular aspect of qualitative research. The decisions suggested by Rossman and Rallis seem to fall within three categories as shown in Figure 3-7: overall research characteristics, project practical considerations, and research design.

Over Research Characteristics

The first decision in the category of overall research characteristics is about the strategy of research. Among the research strategies listed in Figure 3-7, this study adopts evaluation research as a research strategy based on the type of information it provides. As described by Rossman and Rallis, evaluation research can provide either formative or summative information which can be used to define and evaluate the effectiveness of a program. Formative information is normally used to increase the effectiveness of the program. Summative information contributes to a final decision concerning the value and effectiveness of the program in producing intended changes. Based on these two types of information, this study intends to only focus on the summative information to determine the value and effectiveness of the prototype DSS in improving the quality of green roof design decisions.

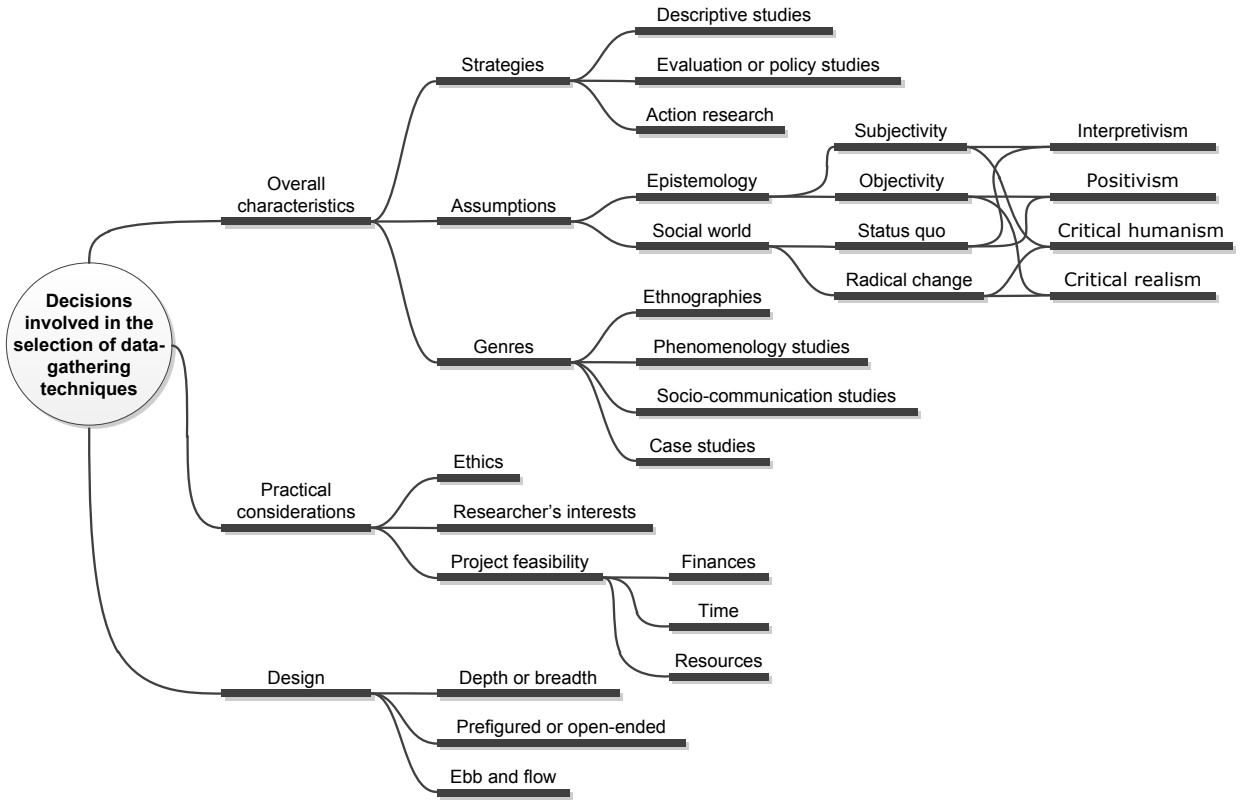


Figure 3-7: Decision involved in the selection of data-gathering techniques suggested by Rossman and Rallis (2003)

In addition to the research strategy discussed above, the second decision in this category is about the researcher's assumptions about epistemology and the social world. According to Rossman and Rallis, the assumptions about epistemology and the social world are the major factors that constrain the research

paradigm that, in consequence, shapes how research data should be collected at the tactical level. As suggested by Rossman and Rallis, the assumptions about epistemology can be defined within the subjectivity and objectivity continuum. Based on a similar approach, the assumptions about the social world can be defined within the status quo and radical change continuum. Essentially, this study aims to understand the usefulness of the prototype DSS as it is from the perspective of individual experience. With these assumptions in mind, it can be considered that this study holds status quo assumptions about the social world and subjective assumptions about epistemology. According to Rossman and Rallis, qualitative research that holds such a combination of assumptions typically falls within the interpretivism paradigm in which the humanistic methods for data gathering are applied. The humanistic methods fundamentally encompass direct interactions in the form of in-depth interviews, extended observations, or some combination of the two. For this study, the implemented humanistic method or technique for data gathering will be discussed soon in this subsection.

Besides the research strategy and researcher's assumptions about epistemology and the social world, the last decision in this category is about the genre of research. This study intends to understand the usefulness of the prototype DSS as perceived by a large group of prospective users through the intensive study of specific experiences drawn from the group. Based on this characteristic, this study seems to fall into the research genre in which Rossman and Rallis referred to as case studies[12]. According to Rossman and Rallis, case studies are descriptive, heuristic, and inductive in nature, which allow a larger phenomenon to be understood through intensive study of one specific experience. Corresponding to Rossman and Rallis, evaluation studies are frequently conducted as case studies in order to produce a rich illustration of program situations, to provide conceivable explanations, and to draw lessons learned. Furthermore, as suggested by Mile and Huberman (1994, cited by Rossman and Rallis 2003), when more than one case is studied, cross-case analyses can also be conducted for comparison purposes. Cross-case analyses allow a researcher to examine the commonalities and/or differences across cases while maintaining the integrity of each case. For this study, the instances drawn from the group of prospective users may include entries from licensed architects, licensed landscape architects, roofing consultants, or engineers who volunteer to evaluate the prototype DSS regardless of their experience in vegetated roofing system selection. These practitioners may be divided into two groups including practitioners who regularly use BIM tools to support their decision making and those who do not use BIM tools on a regular basis. Dividing the practitioners into two groups allows the researcher to conduct cross-case analyses as a means for understanding the usefulness of both the web-based DSS for vegetated roofing system selection with and without the BIM interoperable capability. The issue of participant selection will be discussed in detail in the next subsection.

[12] Although other scholars in the field of qualitative research such as Stake (2000, cited by Rossman and Rallis 2003) see case studies as an overall strategy rather than a research genre, Rossman and Rallis consider case studies a genre wherein qualitative research can be conducted.

Project Practical Considerations

The first decision in the category of project practical considerations is about the project feasibility. In general, Rossman and Rallis suggested that the feasibility of qualitative research significantly depends on three factors as listed in Figure 3-7: finances, time and resources. For the financial concerns, it is reasonable to address that this study is likely to be conducted with limited financial supports. The funds also need to be divided between the development of the prototype DSS and the study regarding the usefulness of the prototype. Because of financial limitations, the data-gathering techniques implemented in this study should be as inexpensive as possible. Similar to the financial concerns, this study is also likely to be conducted within a limited time frame. Thus, for the time concerns, it is necessary to choose data-gathering techniques that are highly adaptable to both research participants' and the researcher's schedules and that allow interactions between research participants and the researcher to occur as much as required. For the resource concerns, Rossman and Rallis suggested that the researcher's knowledge and skills are considered the most important resources that should be taken into consideration. For this study, it can be assumed that the researcher has considerable knowledge on the topics of architectural design and decision support systems and skills necessary for qualitative research. Based on these three factors, it is quite appropriate for the data-gathering activities to be conducted in the form of distance communications. This is because distance communication services are capable of supporting direct communications. They are also less expensive and more flexible than onsite meetings. Furthermore, most of computer-based communication services are equipped with useful tools that can be used for data gathering purposes.

In addition to the project feasibility, the second decision in this category is about the researcher's interests. For this study, the researcher is initially interested in the extension of BIM technology from the information domain toward the knowledge domain. The researcher sees this study as an opportunity to develop a prototype for a web-based, knowledge-driven DSS, which can be used as a system platform for implementing building information contained in BIM models in conjunction with other databases, analytical models, and expert knowledge involved in vegetated roofing system design. While there is a considerable potential for the prototype DSS to be built based on today's technology, the major remaining question is whether the system is useful for the end users in context of their practices. As a result, the researcher is intensely interested to learn more about the usefulness of the prototype DSS in improving the quality of vegetated roofing system design decisions as viewed by its prospective users through this qualitative study.

Besides the project feasibility and researcher's interests, the last decision in this category is about research ethics. For qualitative research, it is important for a researcher to maintain the balance between producing knowledge and protecting the humans studied (Rossman and Rallis 2003). According to Rossman and Rallis, every federally-funded university and organization tends to have an Internal Review Board (IRB), which reviews all proposals, codifies rules, and establishes procedures for research

involving human subjects. The IRB works to ensure that no humans or animals are put at risk and that particular research proposals contain necessary protections. For the reason that Virginia Tech is one of the universities that use the IRB, this study intends to follow the rules and procedures defined by the Virginia Tech's IRB to ensure that the implemented data-gathering techniques are ethical and appropriate for the purpose of this study.

Research Design

The first decision in the category of research design is the decision about the depth of the study. According to Rossman and Rallis, the level of depth can be defined within the depth and breadth continuum. For software evaluation, it can be considered that both broad and in-depth data about the software should be collected to ensure that it can satisfy a broad range of end user's needs. In general, prototype software tends to be reviewed by a small number of end users to gain an in-depth understanding of software functions, features, and requirements. After the software is fully developed, it can then be further evaluated by a large number of end users to gain a broad understanding of end users' needs. Because this study concentrates on the development of prototype DSS, it is appropriate to conduct an in-depth study on the usefulness of the prototype DSS based on a small number of end users. It is also appropriate to assume that, in the near future after the DSS is fully developed and released, a broad study on the system will be separately conducted to ensure that the system can fulfill a broad range of end users' needs.

In addition to the depth of study, the second decision in this category is about the openness of data gathering techniques. As suggested by Rossman and Rallis, the openness of research techniques can be defined within the prefigured and open-ended continuum. Between these two extremes, the techniques applied in this study tend to be prefigured, because this study has a specific focus on the usefulness of the prototype DSS. However, the applied techniques should be as open as possible to allow the end users' to fully share their perspectives and direct experiences of the system. The open-ended techniques allow an in-depth understanding of the usefulness of the prototype DSS to emerge from the data that derives from such end users' perspectives and experiences. As a result, the qualitative research techniques applied in this study should maintain the balance between the prefigured and open-ended natures of the techniques.

Besides the depth of the study and the openness of data-gathering techniques, the last decision in this category is about the flow between research techniques. As suggested by Rossman and Rallis, the flow can be defined by the mix between three research techniques: interviewing, observing, and documenting material culture. However, according to Rossman and Rallis, interviewing is the only technique that allows a researcher to understand individual perspective and suggested that, in practice, interviews may be supplemented by observations to help signal participants' emotions, attention and interest, authenticity, and fatigue. Based on observations, the researcher may decide to suggest questions not anticipated or yield topics that need to be explored. Based on Rossman and Rallis suggestions, this study intends to

apply interviews, supplemented by observations, as a data-gathering technique to collect data about the usefulness of the prototype DSS.

By combining all decisions together, the appropriate techniques for this study can be determined. This qualitative research begins with the researcher's interest to know more about the usefulness of the prototype DSS as viewed by prospective users. The researcher chooses evaluations as a research strategy based on the reason that it provides summative information that can be used to determine the value and effectiveness of the prototype DSS in improving the quality of green roof design decisions. For this study, evaluations are conducted in the form of case studies, which allow the researcher to understand the usefulness of the prototype as viewed by a larger group of end users from data drawn from individual instances. For the reason that this study aims to understand the usefulness of the prototype DSS as it is from the perspective of individual experience, the humanistic research techniques are applied. While humanistic research techniques may include in-depth interviews, extended observations, or some combination, this study chooses to apply in-depth interviews as the main research technique, supplemented by limited observations. It is believed that this implemented technique is sufficient because in-depth interviews allow the researcher to understand individual perspectives as well as to gain an in-depth understanding of the prototype DSS from a small number of end users. It is important to note that the in-depth interviews should maintain a balance between the prefigured and open-ended inquiries and should follow the university internal review broad rules and procedures. Based on the availability of finances, time, and resources, the in-depth interviews are quite feasible to be conducted in the form of distance communications using services such as online phone and meeting services.

While there are a significant number of in-depth interviewing approaches, this study intends to implement a specialized form of in-depth interviews, which Rossman and Rallis referred to as "elites" or experts interviewing. According to Rossman and Rallis, "elites" can be defined as individuals who are influential, prominent, and/or well informed in an organization or community and are selected based on their experience in areas relevant to the research. For this study, elites may include licensed architects, licensed landscape architects, and roofing consultants or engineers who are allowed to make decisions about the vegetated roofing system selection in their organization. In general, Rossman and Rallis suggested that the elite individual is quite astute and could be bothered by the restrictions of narrow or ill-phrased questions. As stated by Rossman and Rallis, elites tend to give good responses to inquiries about broad topics and to intelligent, provocative, open-ended questions that enable them to freely use their knowledge and imagination. While elites can provide valuable information for the research, they are frequently busy with their businesses. It can be considered that one of the most challenging tasks of interviewing elites is to find time to interview.

In conclusion, this subsection provides the comprehensive information about the major decisions involved in the selection of data-gathering techniques and the techniques applied to collect data about the

usefulness of the prototype DSS. Based on the decisions about the data-gathering techniques discussed in this subsection, this study intends to use in-depth interviews, in the form of elites interviewing, as a primary data-gathering technique, supplemented by limited observations. In general, the expected results of the interviews tend to be a large piece of data about the usefulness of the prototype DSS in improving the quality of vegetated roofing system design decisions, which needs to be reduced into manageable pieces that are then coded and categorized into various themes for the analysis purpose. The data reduction/coding phase of the qualitative research process will be discussed further in detail in the following subsection.

3.3.1.2. Research Participants

According to the previous subsection, the qualitative study conducted as part of this research aims to use the in-depth interviews as a research method to understand the usefulness of the prototype DSS as perceived by potential users, who are instances of a larger user population. In order to ensure that the understanding gained from the interview sample can be generalized to the larger population, it is essential to discuss possible strategies that can be used to select interview participants and to establish a research relationship between the researcher and the participants.

In order to select research participants, this study adopts the purposeful sampling method suggested by Patton (1989, cited by Seidman 2006) as a strategy for selecting interview participants. In a series of sampling techniques, Seidman (2006) suggested that the maximum variation sampling technique seems to be the most appropriate technique for selecting interview participants for interview studies. This is because the method provides the widest possibility to select an interview sample that well represents the larger population. The maximum variation sampling technique involves defining the maximum range of participant's settings/contexts, and then selecting research participants from the widest range of possible participants in such settings/contexts. For this study, the maximum variations of settings/contexts and potential participants can be summarized in Table 3-1.

Table 3-1: Possible groups of qualitative research participants

Experience of green roof design	**Use of BIM authoring tools**	
	BIM Users	**Non-BIM Users**
Experienced Green roof Designers	Group A	Group B
Inexperienced Green roof Designers	Group C	Group D

As shown in Table 3-1, the maximum range of participant settings/contexts can be defined by the use of BIM authoring tools and the experience of green roof design in conventional design practices. This study assumes that the practitioners' use of BIM authoring tools can range from "not used at all" to "used regularly". As well, this study assumes that the practitioners' experience of green roof design can range from "have never designed a green roof" to "have designed many green roofs". Based on the defined setting/context variations, the possible research participants can be categorized into four groups as shown in Table 3-1.

In Table 3-1, “Group A” participants are design practitioners who regularly use BIM authoring tools in their practices and have experience with green roof design. This group of participants can use the prototype DSS with the BIM support capability. The in-depth knowledge gained from this group of participants has considerable potential to help the researcher understand the usefulness of the prototype DSS with the BIM support capability as well as the validity of the DSS in solving the problem of the vegetated roofing system selection.

As presented in Table 3-1, “Group B” participants are design practitioners, who do not use BIM authoring tools in their practices but have experience with green roof design. This group of participants can also use the prototype DSS without the BIM support capability. The in-depth knowledge gained from this group of participants has considerable potential to help the researcher understand the usefulness of the prototype DSS without the BIM support capability as well as the validity of the DSS in solving the problem of the vegetated roofing system selection.

In Table 3-1, “Group C” participants are design practitioners who regularly use BIM authoring tools in their practices but have no experience with green roof design. This group of participants can use the prototype DSS with the BIM support capability. The in-depth knowledge gained from this group of participants has considerable potential to help the researcher understand the usefulness of the prototype DSS as perceived by practitioners who are new to the idea of using a BIM interoperable DSS for vegetated roofing system selection. The in-depth knowledge gained from this group of participants also has considerable potential to help the researcher understand the likelihood of new technology adoption influenced by the provided BIM support capability.

Lastly, “Group D” participants are design practitioners who do not use BIM authoring tools in their practices and have no experience of green roof design. This group of participants can use the prototype DSS without the BIM support capability. The in-depth knowledge gained from this group of participants has considerable potential to help the researcher understand the usefulness of the prototype DSS as perceived by practitioners who are new to the idea of using DSS for vegetated roofing system selection.

After the groups of participants are defined, it is essential to discuss the strategy used to establish a research relationship between the researcher and the interview participants. According to Grant (2007), the target users of the decision-making framework for vegetated roofing selection include licensed architects, landscape architects, and roof consultants or engineers. For this study, these practitioners can be seen as the entire population from which the interview participants are drawn.

In order to effectively establish a research relationship between the researcher and the participants, Seidman (2006) suggested that the researcher should contact the potential participants directly instead of accessing through the formal and informal “gatekeepers” such as supervisors at work or leaders in social

groups. This is because the gatekeepers tend to have certain influences on the participants' thoughts and feelings related to the topic of being studied, which may later affect the study results.

Based on Seidman's suggestion above, this study intends to search for participants from professional company directories provided on area specific Web sites such as Greenroof.com to build a pool of participants. After the participant pool is built, the contacts of design practitioners who work in companies found from the search may also be found from the professional social network Web sites such as LinkedIn.com. In addition to the contacts, such Web sites also provide the practitioners' areas of interests which could help filter the potential research participants out of the participant pool. As suggested by Seidman, the records of the companies' and potential participants' contacts should be kept for later use.

Once contacts are accessible, the researcher can begin to contact the potential participants and briefly introduce himself and briefly present the research. As suggested by Seidman, the researcher should not ask for participation from the potential participants during the first contact. Instead, the researcher should ask for permission to contact the potential participants at a later time via e-mail to provide more formal information about the research, especially the objectives and methodology of research and the commitments expected from the potential participants. After the potential participants have had a chance to read the document, the researcher can make a follow-up phone call to ask for a commitment to participation. If the potential participants are interested in participating in the study, they should then sign a consent form and schedule an interview.

When searching for research participants, Seidman suggested that using the in-depth interviews as a research method could be problematic if the number of interview participants is too small or too large. For this study, based on the valuations of participants and their settings/contexts presented in Table 3-1, problems may occur when there are only one or two participants in each group. For example, if there is only one participant in a group of participants, the in-depth understanding gained from the interviews may be led in only one direction. Again, if there are only two participants in a group of participants, there is about a thirty-percent chance that the interview data will split into two opposite directions, which would be difficult to interpret. Therefore, it is most reasonable to have at least three participants in each group of participations, which leads to the possible minimum number of twelve participants for the interview study.

In conclusion, this study adopts the purposeful sampling method suggested by Patton (1989, cited by Seidman 2006) as a strategy for selecting interview participants. This this strategy has considerable potential to ensure that the understanding gained from the interview sample can be generalized to the larger population. In addition, this study intends to directly contact the potential interview participants selected from the population of licensed architects, landscape architects, and roof consultants or engineers to establish research a relationship between the researcher and the interview participants.

3.3.2. Data Reduction/Coding

Continuing from the data collection phase, the data reduction or coding phase of the qualitative research process suggested by Groat and Wang (2002) focuses on reducing a large piece of data into manageable “chunks” by coding the “chunks” into various themes. For this study, this research activity encompasses four actions derived from the generic process for data analysis suggested by Rossman and Rallis (2003). The four actions include organizing the data, getting familiar with the data, generating categories and themes, and coding the data. This subsection will introduce the data analysis process and discuss each action of the data reduction activity in detail.

The data reduction activity performed in this study comprises a set of actions derived from the generic process for data analysis suggested by Rossman and Rallis (2003). According to Rossman and Rallis, data analysis can be seen as a process that encompasses seven steps: 1) organizing the data, 2) becoming familiar with the data, 3) generating categories and themes, 4) coding the data, 5) interpreting, 6) searching for alternative understanding, and 7) writing the final report. As viewed by Rossman and Rallis, each step of the process comprises data condensation whereby the large volume of collected materials is synthesized into manageable pieces with theme interpretation wherein meaning and insight are brought to participants’ words and actions. This seven-step analytical process seems to have parallels with the qualitative research process suggested by Groat and Wang (2002), which entails data collection, data reduction/coding, data display, and conclusion drawing and/or verifying, as shown in Figure 3-8.

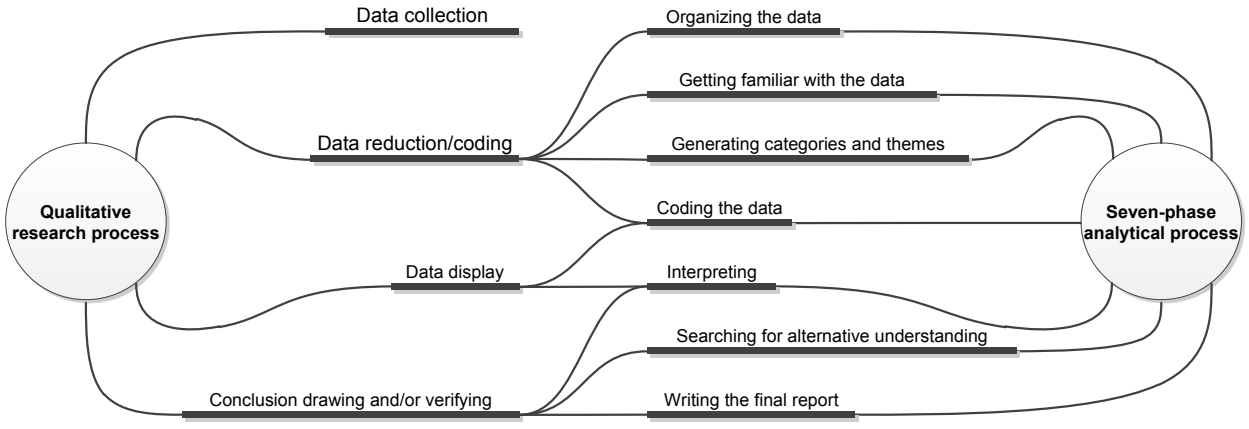


Figure 3-8: Parallels between the qualitative research process and the seven-phase analytical process

Referring to Figure 3-8, the data reduction phase of the qualitative research process begins after the data reduction phase of the process is completed. During the data reduction phase, a large piece of collected data can be reduced and coded into categories and theme through a set of actions that represent the first four steps of the seven-step analytical process: 1) organizing the data, 2) getting familiar with the data, 3) generating categories and themes, and 4) coding the data. Continuing from the data reduction phase, the data display phase of the qualitative research process moves the categorization and thematic analysis to a higher level of integration and synthesis, which leads to a greater level of interpretation. The data

display activity involves organizing and presenting the categorized and thematic data for analysis and communication purposes using various forms of data representation. Finally, the qualitative research process moves to the conclusion drawing and/or verification phase, which encompasses a set of actions that represent the last three phases of the seven-step analytical process: interpreting, searching for alternative understanding, and writing the final report. This subsection will continue to discuss the four actions of the data reduction activity in further detail and leave the data display and conclusion drawing and/or verifying activities to be discussed in later subsections.

Organizing the collected data is the first step of the data reduction activity. According to Rossman and Rallis, this step typically begins with cleaning up and making a few modifications to the collected data, which tends to be collected in a variety of forms and from various sources. For this study, the collected materials may primarily include notes, records, and transcriptions of in-depth interviews. The materials may also include hunches and analytic ideas that emerge during the course of study. To make them comprehensible and manageable, these materials should be cleaned up as soon as possible after they are collected. To make them retrievable, such materials may also be modified by adding information such as interview date and interviewee's identification to each item, if applicable. After the collected materials are cleaned up and modified, they can then be inventoried, sorted, and stored using several of the following tools: note cards, post-it notes, magic makers, file folders, and/or software programs. This study considers using software programs to inventory and manage the soft copies of interview notes, records, transcriptions, and other collected materials, and using file folders to manage the hard copies of such materials.

The second step of the data reduction activity, after the collected data is organized, is to become familiar with the data. According to Rossman and Rallis, this step is mainly about going through the collected data repetitively to gain an insight into the significant themes and meaning embedded in the data. For this study, the researcher can be seen as a primary instrument for transcribing all the interviews. After the interviews are transcribed, the researcher can then read through the transcriptions again and again until the themes and meaning embedded in the data are well understood. Besides using the researcher as an instrument, this study also considers using software programs, if they are affordable and applicable, to transcribe the interviews and to analyze the transcriptions. The benefit of using a software program for transcribing the interviews is that the researcher can compare two sets of transcriptions to ensure the accuracy of the transcriptions. And, the benefit of using a software program for analyzing the transcriptions is that the results generated by the program may provide an alternative understanding of the themes and meaning embedded in the data.

The third step of the data reduction phase is to generate categories and themes for data analysis. According to Rossman and Rallis, a category is a word or phrase that explicitly describes some segment of the data. Rossman and Rallis divided categories within the data into two types; the indigenous categories derived from participants' expressions and the analyst-constructed categories predetermined

by the researcher. In general, qualitative studies should maintain a balance between these two methods of categorizations. Based on these principles, this study intends to discover the indigenous categories through the analysis of participants' expressions as a means to understand the usefulness of the prototype DSS as viewed by the participants or end users. This study also considers using certain analyst-constructed categories to provide direction for data gathering. For example, the analyst-constructed categories may include the five characteristics of innovations discussed in Chapter 2.1.2 of this dissertation: relative advantage, compatibility, complexity, trialability, and observability. In addition to the term "category", Rossman and Rallis also defined a theme as a phrase or sentence that indirectly and implicitly describes some portion of the data. The themes typically emerge from the deep familiarity with the data gained from the process of categorization. As suggested by Rossman and Rallis, the researcher may define a theme by looking for recurring words and/or phrases that have similar and/or related meanings and then create a phrase or sentence to describe this group of words and/or phrases. For example, within the context of this study, the researcher may discover phrases as follows: "need more characters in the text fields" and "may add a custom green roof type to the dropdown lists". These phrases reveal the theme of improving form controls. Because of its simplicity, this study intends to apply this technique for theme generation.

The last step of the data reduction activity is to code the data into categories and themes. According to Rossman and Rallis, coding can be seen as a formal representation of categorizing and thematic analysis. In order to perform coding by hand, Rossman and Rallis suggested that the researcher start with formatting the word-processed data using wide margins on the right hand side. Next, the researcher can begin to code the data by bracketing chunks and writing a word or a symbol, which represents a category or theme, on the margin. For example, within the context of this study, the researcher may write down the abbreviation RA on the margin to represent the category of relative advantage. As suggested by Rossman and Rallis, the researcher may begin coding with a few broad categories and then refine these broad categories, or add new categories in the next iteration of coding. They also suggested that the researcher perform coding several times to gain a deeper understanding of the data. Furthermore, the researcher should keep a record of what has been coded in all materials in one place for retrieving purpose. Because Rossman and Rallis's hand coding technique seems to be very practical and straightforward, this study aims to use this technique as a basis for data coding.

To summarize, the reduction or coding phase of the qualitative research process, conducted as part of this study, consists of four steps. The first step is to organize the collected data into more manageable and retrievable parts. The second step is to become familiar with the data in order to gain insight into the salient themes and meaning embedded in the data. The third step is to generate categories and themes that explicitly and implicitly describe different pieces of data. The last step is to code the data according to the generated categories and themes. As suggested by Groat and Wang (2002), the coded data can be further organized and displayed in the form of charts, graphs, and tables, which can be useful for data

analysis and communication. The next subsection will discuss the data display phase of the qualitative research suggested by Groat and Wang in detail.

3.3.3. Data Display

The data display phase of the qualitative research process begins after the data reduction/coding phase is accomplished. According to Groat and Wang (2002), this phase of the process involves organizing and presenting the categorized and thematic data into the form of charts, graphs, or tables. Groat and Wang suggested that visualized forms of data representation can be beneficial for enhancing both data analysis and communication.

While there are many forms of data displays, this study intends to use tables for organizing and presenting the categorized and thematic data for various reasons. First, tables allow the researcher to organize and present texts, which can be considered the primary form of data to be analyzed and displayed. As well, tables allow the researcher to make comparisons between different pieces of data. Table 3-2, for example, represents one possible way to use tables for data analysis and display. Table 3-2 enables the researcher to conduct a cross-case analysis as a means of understanding the usefulness of the prototype DSS as viewed by the practitioners based on the four groups of practitioners discussed in Chapter 3.3.1.2. In Table 3-2, four groups of practitioners are displayed in four separate columns whereas the categories and themes are presented in separate roles.

Table 3-2: Data display for cross-case analysis

Category and Theme	Group A	Group B	Group C	Group D

After the data display phase is completed, the qualitative research process suggested by Groat and Wang then moves to the conclusion drawing and/or verifying phase of the process. The conclusion drawing and/or verifying phase concentrates on identifying patterns, providing explanations, and evaluating the findings. The conclusion drawing and/or verifying phase will be discussed in further detail in the next subsection.

3.3.4. Conclusion Drawing and/or Validating

Conclusion drawing and/or validating is the last activity of the qualitative research process suggested by Groat and Wang (2002). As mentioned in Chapter 3.3.2 of this dissertation, this phase of the qualitative research process encompasses three actions derived from the generic analysis process suggested by Rossman and Rallis (2003). The three actions include interpreting, searching for alternative understandings, and writing a report. This subsection discusses these three actions in detail.

Interpreting is the first action of the conclusion drawing and/or verifying phase of the qualitative research process implemented in this study. Fundamentally, “interpretation is storytelling” (Rossman and Rallis 2003, 288). Based on this fundamental concept, interpreting encompasses two major tasks. The first task

involves interpreting all the analyses that have been performed throughout the course of study. The second task involves putting together a story of what the researcher has learned from the participants' and from the researcher's own experiences and interpretations. In order to perform these tasks effectively, Rossman and Rallis suggested that it is important for the researcher to provide "thick description", which comprehensively explains the contextual clues (e.g., physical surroundings, actions, and/or events) used to interpret the meaning embedded in a particular piece of data. It can be considered that thick description makes interpretation possible and helps the audience of the research project make sense of the story put together by the researcher. Based on the fundamental principles suggested by Rossman and Rallis, this study intends to provide a story that is put together from the participants' hand-on experiences of the prototype DSS and their perspectives toward the usefulness of the prototype. As well, it intends to provide enlightenment regarding researcher's interpretations on thematic and cross-case analyses. Furthermore, instead of providing a single dimension of truth, the story put together by the researcher aims to cover different levels of understanding ranging from personal level to theoretical level, supported by necessary thick descriptions.

The second action of the conclusion drawing and/or verifying activity, after interpreting, is to search for alternative understandings. This second action focuses on verifying the story put together by the researcher to ensure that the story is well supported by the data, based on compelling logic, and apparently sensible to others. According to Rossman and Rallis, alternative understandings can be found in certain ways. One way is that the researcher may share the researcher's analyses and interpretations with knowledgeable persons such as research participants, colleagues, advisory committees and/or faculty advisors, and then ask for their understandings. Another way is that the researcher may find alternative understandings from the literature. Because these two methods are quite applicable, this study intends to use both methods to find alternative understandings that will be used to verify the research findings.

The last action of the conclusion drawing and/or verifying activity is to write the final report. According to Rossman and Rallis, after the findings are verified against alternative understandings, they can then be presented to the audience of the research project in various forms (e.g., writing material, film, and audio) depending on the audience types and presentation purposes. Because the audiences of this study are primarily in academia, and this study is conducted to primarily serve academic purposes, this study intends to present the findings in a form of formal writing. As suggested by Rossman and Rallis, there are a number of possible ways to write a formal report. However, most formal reports normally provide the researcher's interpretations on the performed analyses, supported by thick descriptions. Therefore, this study intends to provide these two elements in the final report presented in Chapter 5 of this dissertation.

In conclusion, within the context of this study, the conclusion drawing and/or verifying activity encompasses three actions: interpreting, searching for alternative understandings and writing a report. Through interpretation, this study aims to provide a story put together from the participants' experiences

and perspectives concerning the usefulness of the prototype DSS and from the researcher's interpretations on a variety of analyses conducted throughout the course of study. In order to ensure that the story put together by the researcher is sound, logical, and grounded on the collected data, the story will then need to be verified against alternative understandings. After the story is verified, the story can then be written in a form of final report provided in Chapter 5 of this document.

3.4. Author's Background

To identify any potential bias in the scope of research, a brief discussion of the author's background is offered here. The author is a registered architect in Thailand, with five years of architectural practice based in Bangkok (2001-2006). The author has considerable experience in designing conventional roofing systems, but does not have any experience in designing a vegetated roofing system for a building project. Therefore, the study of the DSS feature for vegetated roofing system selection conducted in this dissertation is not influenced by any previous familiarity with any particular vegetated roofing system design methodology. For DSS design and development, it should be noted that the author is quite proficient in programming languages and software development environments typically used for building DSS applications and has substantial experience with the design and development of small-scale DSS applications. However, because the researcher has no prior experience with the design and development of commercial proprietary DSS applications, the development of a prototype DSS conducted as part of this dissertation is not affected by any prior familiarity with any particular programming language, software development environment, or proprietary DSS application. It is hoped that these circumstances lend a degree of neutrality to this research effort.

Chapter 4

4. Prototype BIM Interoperable Web-Based DSS

4.1. Prototype Development

As mentioned in Chapter 1.3, one objective of this study is to examine the possibilities for developing a BIM interoperable, web-based DSS to be used by building design practitioners. To meet this objective, the first phase of this study entirely focuses on the development of a prototype DSS for vegetated roofing system (green roof) selection. Referring to the information given in Chapter 3.2.2, the stakeholders of this prototyping project includes the Center for High Performance Environments (customer), the author of this dissertation (developer), and the first phase research participants (end users). Derived from ideas given by the client and end users, the prototype DSS has three essential features: 1) decision support for green roof selection, 2) BIM compatibility, and 3) decision-making documentation and collaboration. To develop the prototype DSS, this study adopts the prototyping software engineering process, which comprises five steps: 1) communication, 2) quick plan, 3) modeling/quick design, 4) construction of prototype, and 5) delivery, deployment, and feedback. The whole prototype DSS development is divided into three prototyping cycles whereby Prototyping Cycle 1 focuses on the decision support feature, Prototype Cycle 2 focuses on the BIM compatibility feature, and Prototyping Cycle 3 focuses on the decision-making documentation and collaboration feature. In the following sections of this chapter, each prototyping cycle will be discussed in detail. Furthermore, the outcome of the prototype development, which is the BIM interoperable, web-based DSS for vegetated roofing system selection, will be presented at the end of this chapter.

4.2. Prototyping Cycle 1

4.2.1. Prototyping Cycle 1: Communication

For this study, the communications between the Center for High Performance Environments (CHPE) faculty researchers, the author, and the research participants began in early 2012. The purposes of the communications were to understand stakeholders' objectives for prototype DSS development and to gather requirements that help define DSS features and functions. To fulfill these two purposes, the author started the communications by arranging several meetings with the CHPE faculty researchers to discuss the objectives and requirements of prototype DSS development. The author then continued to broaden the understanding of the prototype DSS features and functions by inviting the Architecture, Engineering, and Construction (AEC) professionals to participate in the prototyping project. Meeting with both CHPE faculty researchers and AEC practitioners has significantly helped the author adequately understand the objectives and requirements of prototype development discussed in this section.

Based on the information gathered from the meetings between the CHPE faculty researchers and the author, the initial objective of prototype DSS development was to develop a prototype BIM-compatible, web-based DSS for vegetated roofing system selection, intended to be distributed through a website or to be a subsystem of a larger web-based decision support application for architectural design assistance. Corresponding to the initial objective, prototype DSS development should satisfy three high-level requirements as follows:

- The prototype DSS should well implement the VRSS Framework to provide decision support for vegetated roofing system selection.
- The prototype DSS should be compatible with major BIM-based tools employed by AEC practitioners.
- The prototype DSS should provide support for collaboration between project stakeholders during the design phases of the building lifecycle whereby decisions made during these phases have significant impact on the quality of buildings.

To effectively fulfill the three high-level requirements above, it is quite common to reduce the whole prototyping process into manageable pieces. Therefore, the prototyping process discussed in this chapter has been divided into three cycles whereby each cycle only focuses on one of the three requirements above.

The primary goal of Prototyping Cycle 1 discussed here was to develop a DSS that provides decision support for vegetated roofing system selection based on the VRSS Framework. Because the website or web-based application for distributing the prototype DSS has not been developed, the secondary goal of Prototyping Cycle 1 was to develop a demonstration website to show how the prototype DSS may be provided for the end users.

In addition to the objective and requirements above, it is important to gather detailed information about software operation from potential users of the DSS. To have the AEC professionals participate in prototype DSS development, the author followed the Virginia Tech Internal Review Board (IRB) approved procedures to conduct research involving human subjects as mentioned in Chapter 3.3.1. After the IRB approved the research protocol and other supporting documents, the invitations to participate in research were sent out to almost one hundred professionals in the forms of in-person communication, tri-fold brochure, and e-mail. From all of the professionals who received the invitations, there were nine AEC practitioners who voluntarily participated in the prototyping project as end users. Within this group, there were six practitioners, including four roofing consultants, one roofing researcher, and one roofing manufacturer representative, who participated in the communication phase of the prototyping process.

The communications between the author and the research participants were conducted in the form of a face-to-face meeting. To facilitate the communication, the author provided the research participants with information about research in general and the demonstration of the previously developed prototype DSS discussed in Appendix B. Based on the provided information and software demonstration, the research

participants were able to give valuable comments and feedback on the prototype DSS within the scope of Prototyping Cycle 1 as follows:

- Both the prototype DSS and the demonstration web site as a whole should be easy to navigate through and operate.
- The prototype DSS should provide detailed information about the VRSS Framework, especially the required decision-making variables and the models and algorithms used for deriving the attributes of design alternatives for each decision-making factor.
- The prototype DSS should provide assistance with the decision-making variables because end users tend to have different background knowledge and may not be familiar with some variables required by the DSS.

Throughout the communication phase of the prototyping process, the author learned about stakeholders' objectives for prototype development and requirements that help define system features and functions. To satisfy such objectives and requirements, Prototyping Cycle 1 focused on the development of a demonstration website and a prototype DSS for vegetated roofing system selection.

4.2.2. Prototyping Cycle 1: Quick Planning

4.2.2.1. Prototyping Cycle 1: Scope

According to Chapter 3.2.3, the planning phase of the prototyping process usually involves defining the scope of prototype development. As discussed in the previous section, the scope of Prototyping Cycle 1 was to develop a prototype web-based DSS for vegetated roofing system selection and a demonstration website for hosting the DSS. Within this defined scope, the possible, high-level use case diagrams of the demonstration website and the prototype DSS can be developed and are presented in this section.

As shown in Figure 4-1, the website is assumed to be accessed by two groups of users: building project stakeholders and DSS administrators. From the front end, the building project stakeholders including, but were not limited to, owners, designers, contractors, and managers, can navigate freely through the website to learn about the available DSS. The project stakeholders must register to use the DSS provided by the website. After they log into the system, project stakeholders can use multiple DSS provided on the website and can request for user support services from the DSS administrators. From the back end, the DSS administrators can access the website to maintain the DSS and to provide user support services for project stakeholders. To narrow the scope of prototype development, the prototype system developed as part of this study only focuses on the use cases that relate to the project stakeholders. As presented in Figure 4-1, the gray-colored use cases in the diagram will be further examined in future development.

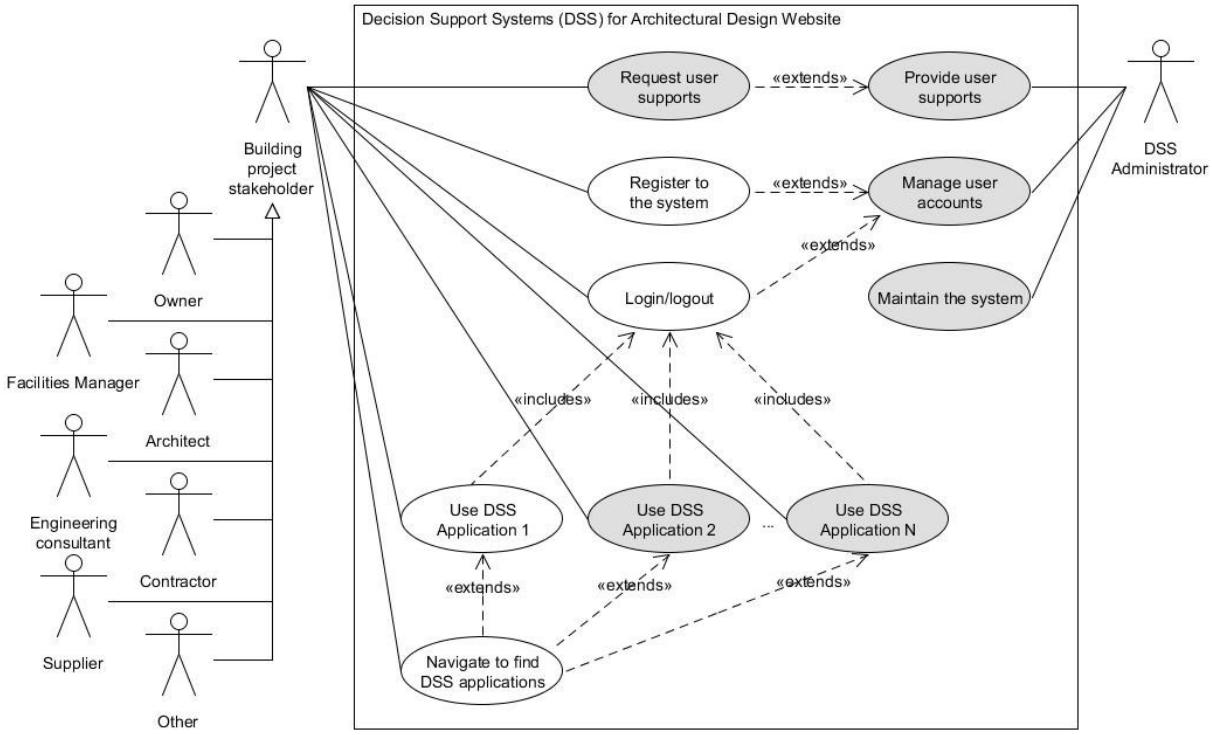


Figure 4-1: Use case diagram for the demonstration website

While the actual website or web-based application should host multiple DSS as presented in Figure 4-1, the demonstration website developed as part of this study has provided only one DSS, which is the Vegetated Roofing System Selection DSS (VRSS DSS). Based on the information gathered from the communication phase, Grant's (2007) dissertation, and the lessons learned from the previous prototype DSS discussed in Appendix B, the use case diagram of the VRSS DSS can be developed as shown in Figure 4-2.

In Figure 4-2, the end users can be divided into two groups: decision makers and information providers. The decision makers including, but not limited to, architects, engineers, and roofing consultants can log into the system to use the VRSS DSS for vegetated roofing system or green roof design assistance. The information providers, which include manufacturers and contractors, can log into the system to manage their products and/or profiles. To narrow the scope of development, the prototype VRSS DSS developed as part of this study has provided only the features and functions that will be used by the decision makers as shown in Figure 4-3. Within the scope of prototype development presented in Figure 4-3, Prototyping Cycle 1 only focused on the core functions of the VRSS DSS derived from the VRSS Framework as shown in Figure 4-4.

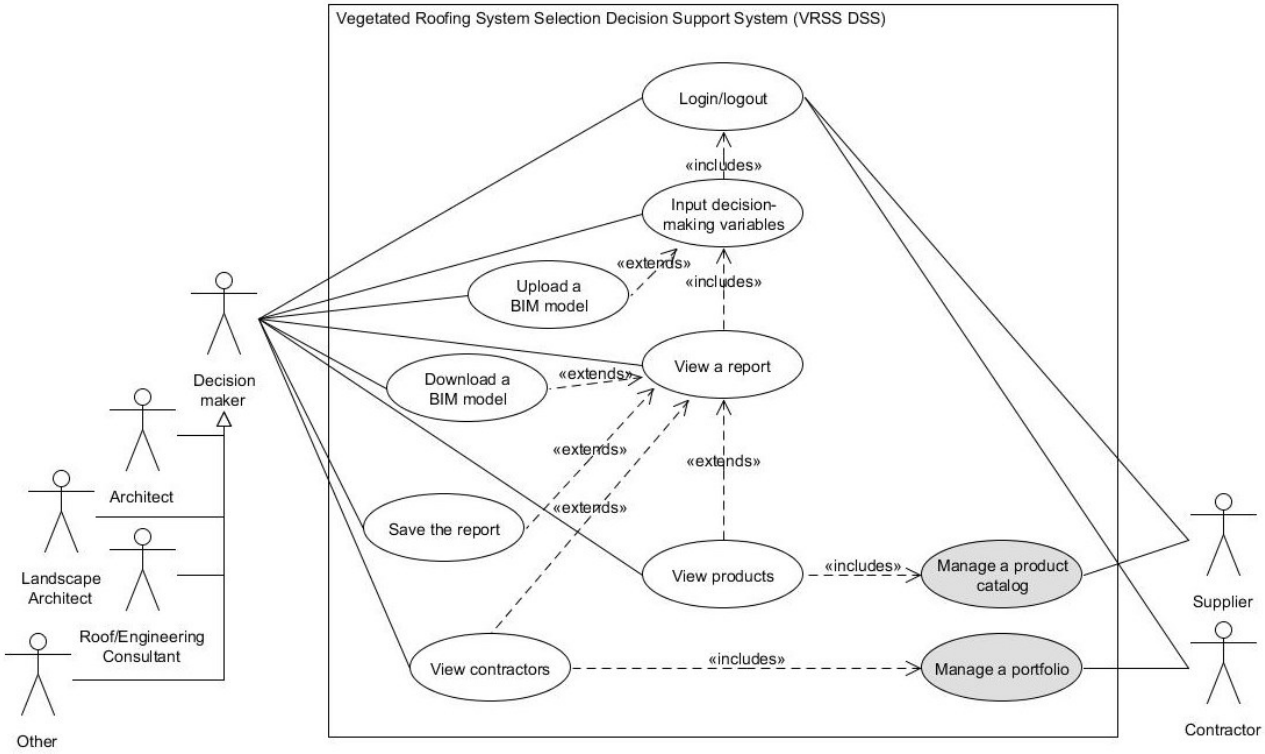


Figure 4-2: Use diagram for the VRSS DSS

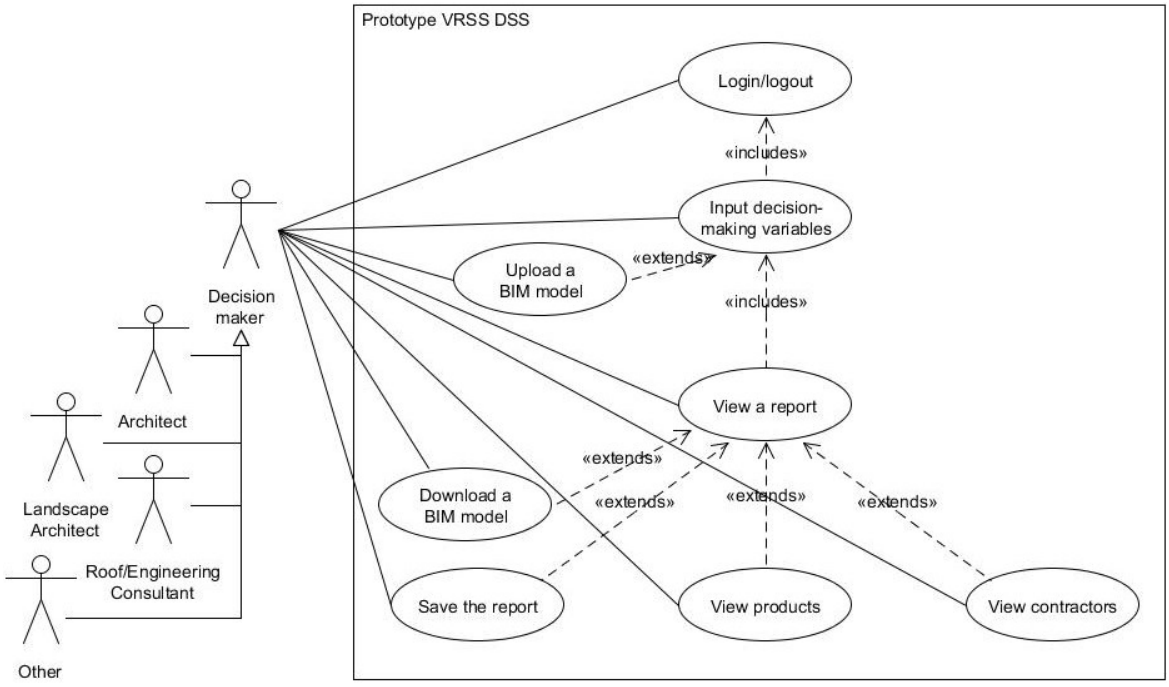


Figure 4-3: Use case diagram for prototype development

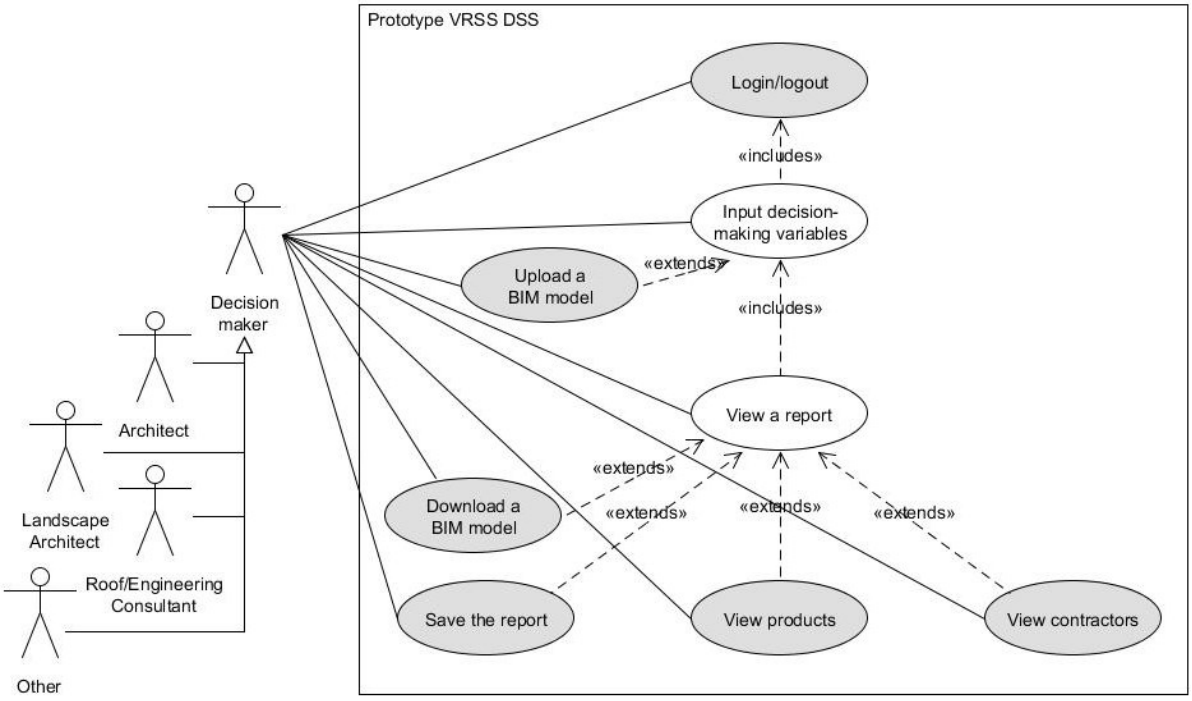


Figure 4-4: Scope of Prototyping Cycle 1

According to the scope of prototype development discussed above, Prototyping Cycle 1 involved developing the demonstration website for hosting the VRSS DSS. Prototyping Cycle 1 also included developing the prototype VRSS DSS that provides decision support information for vegetated roofing system selection. To narrow the scope of prototype development as a whole, it was assumed that both the demonstration website and the prototype DSS developed as part of this study will provide only the software features and functions that are observable by the decision makers. It was also assumed that the remaining software features and functions will be taken into consideration in the future development.

4.2.2.2. Prototyping Cycle 1: Feasibility Study

In addition to the scope of prototype development, the planning phase of the prototyping process often includes conducting a feasibility study on four dimensions: technology, finance, time, and resources. To ensure that the prototype DSS is feasible to be developed, a brief, but reliably comprehensive feasibility study was conducted as part of Prototyping Cycle 1. Said study is elaborated below.

Technology

The first dimension of project feasibility that should be taken into consideration includes the software technologies for DSS development. As mentioned in Chapter 2.2.3, the paradigm change from the procedural programming to object-based or object-oriented programming has a considerable impact on how web-based or web applications have been developed over the past decades. Object technology has enabled the development of sophisticated and complex software application and has become the dominant technology behind the majority of web applications available today. Based on object

technology, web-based DSS applications can be developed using client-side scripting languages alone or together with server-side scripting languages or with high-level, multipurpose programming languages. As discussed in Chapter 2.2.4, client-side scripting languages such as JavaScript tend to be appropriate for small-size DSS. Server-side scripting languages such as PHP and Ruby are widely used for developing large, data-driven DSS. Multipurpose programming languages such as Java and C# are very appropriate for developing model-driven and knowledge-driven DSS.

In addition to object technology, there are other technologies that should also be taken into consideration, especially agile software development, design patterns, open source software, and Ruby on Rails. According to the information provided by the Agile Alliance, found at www.agilealliance.org, "Agile" is a software development methodology that reveres four values: 1) individuals and interactions over process and tools, 2) working software over comprehensive documentation, 3) customer collaboration over contract negotiation, and 4) responding to change over following a plan. The idea of agile development has influenced other software practices such as Test-Driven Development (TDD) or later Behavior-Driven Development (BDD), which help a software team deliver working software by considering desired behaviors upfront. Another software practice that should be mentioned is the use of design patterns. As mentioned in Chapter 2.2.4 and 3.2.4, among a considerable number of design patterns, the Model-View-Controller (MVC) has become the most popular pattern for web application development because it fits well with the modularized or multi-layer nature of web applications. Due to the rapid growth of open source software communities, there are a growing number of open source software projects that have been involved in web application development. Among these projects, Ruby on Rails seems to have a significant impact on how web applications have been developed largely because it embraces the MVC pattern, works with HTTP protocol, promotes the convention over configuration approach, and integrates Object-Relational Mapping (ORM) into its core (Freeman and Sanderson 2011).

Based on the software technologies above, the author decided to mainly develop the prototype DSS using multipurpose languages. This is because the prototype DSS developed as part of this study fell within the category of knowledge-driven DSS as discussed in Chapter 2.3. In addition to the multipurpose languages, client-side scripting languages may be used to reduce web application server workload and to enhance users' experience. On top of programming languages, web applications are often developed using a web application framework. Like the phenomenally popular framework, Ruby on Rails, the frameworks that are widely adopted by web application developers seem to comply with agile, TDD, or BDD practices, embrace the MVC pattern, work well with HTTP protocol, and require minimum configuration. These features can be used as criteria for choosing an appropriate web application framework for this prototyping project.

Although the author has been familiar with some multipurpose software programming languages and development environments as mentioned in Chapter 3.4, the author still believes that choosing web programming languages and frameworks can be an interesting challenge. Prior to the prototyping effort

discussed in this chapter, the author has learned to develop web-based DSS using Servlets and JSP, which is one of the frameworks built upon the Oracle's Java Enterprise Edition (Java EE) platform. The author is also familiar with the ASP.NET Web Forms, which is one of the frameworks built upon the Microsoft's .NET platform, which the author used for developing the second throwaway prototype presented in Appendix B. From the author's experiences, these two frameworks are not likely to fit well with the previously discussed criteria. As a result, the author decided to adopt a new web application framework as a platform for developing the prototype DSS discussed in this chapter.

To select a web application framework and its associated programming languages, the author started by searching information about the popular web application frameworks on the Internet. The author found interesting information from several websites that have conducted surveys on web technologies such as www.netcraft.com and www.w3techs.com and from some articles such as the one posted by Zorgdrager (2010), which discussed the detailed comparison of six popular Java EE web application frameworks. Such information considerably helped the author narrow the choices of web application frameworks to some of Java EE and .NET frameworks. The author then continued to search for learning aids focusing on these frameworks such as tutorials and published documents. Due to the greater availability of learning aids, the author decided to spend a considerable amount of time learning about two of the following frameworks: JavaServer Faces (JSF) and ASP.NET Model-View-Controller (ASP.NET MVC). Based on the author's learning experience, the features of these two frameworks that correspond to the criteria discussed earlier can be summarized in Table 4-1.

Table 4-1: Feature of JSF and ASP.NET MVC

Feature	JSF	ASP.NET MVC	Note
Comply with agile, TDD, or BDD practices	Yes	Yes	JSF applications may need multiple units testing APIs[a] to test all major components of the applications whereas ASP.NET MVC applications require only one API for unit testing.
Embrace the MVC pattern	Yes	Yes	Both frameworks fully embrace the MVC pattern; however, ASP.NET MVC application structure better represents the MVC pattern than JSF.
Work well with HTTP protocol	Yes	Yes	Both frameworks well implement the HTTP protocol, which allows the client and server computers to communicate economically with minimal interruption.
Require minimum configuration	Unlikely	Yes	Although several IDEs[b] used for developing JSF and ASP.NET MVC applications provide automated configuration functions, ASP.NET MVC configuration seems effortless because of its convention over configuration approach, which is not fully implemented in JSF.

a. Application Programming Interface (API)
b. Integrated Development Environment (IDE)

As summarized in Table 4-1, both frameworks seem to be appropriate to for web-based, knowledge-driven DSS development in general. However, the author has preferred to use the ASP.NET MVC framework for prototype DSS development for several reasons. First, using ASP.NET MVC makes prototype DSS development easily manageable, especially when it comes to coding, testing, and deployment. Second, ASP.NET MVC very well implements the Representational State Transfer (REST) architecture, which helps web applications effectively handle user's requests and server responses. Third, the IDEs used for developing ASP.NET MVC applications also provide templates that can be easily modified using Cascading Style Sheet (CSS) and a built-in authentication system that can be used

instantly. Although ASP.NET MVC is appealing to be used for prototype DSS development, it is important to study other dimensions of project feasibility, including finance, time, and resources, to ensure that this framework is appropriate for prototype DSS development.

Finance

The second dimension of project feasibility is the project finance. Because the prototype DSS was developed as part of academic research, it was important to keep the prototype development cost as low as possible. This can be done by using free and/or inexpensive DSS development tools and relying on available resources. In general, both JSF and ASP.NET MVC are open source web application frameworks, which are free to use. Furthermore, the major IDEs that support these frameworks are free of charge. Therefore, the financial feasibility of the project has become heavily dependent on the available resources, which will be discussed further in this section.

Time

The third dimension of project feasibility is the time required for prototype development. Due to the reason that the prototype DSS has been developed in a low-risk, non-competitive setting, the time was not a major project constraint. This gave valuable opportunities for the author to get familiar with emerging technologies including, but not limited to, programming languages, web application frameworks, and development tools that seem to be more appropriate for developing the prototype DSS than those used for developing the throwaway prototypes discussed in Appendix B. Such opportunities are not common for highly competitive software prototyping projects whereby software developers may sometimes use inappropriate, but highly familiar languages, frameworks, or tools to quickly complete prototyping projects. Once the author has developed new knowledge and skills through Prototyping Cycle 1, the prototype DSS should be developed in a timely manner during Prototype Cycle 2 and 3.

Resource

The last dimension of prototyping project feasibility comprises the available resources. According to Pressman (2010), the resources involved in software projects can be divided into three categories: human, reusable software, and environment. To ensure the feasibility of a prototyping project, it is important to examine the available resources based on these three categories in detail.

For human resources, the prototype development team comprised the CHPE (customer), the author (developer), and the research participants (end users), as mentioned in the previous section. As part of the team, the CHPE faculty researchers were responsible for defining project objectives and requirements related to software features and functions. The author was responsible for gathering information about the project from the customer and end users, designing, building, testing, and deploying the prototype DSS, and acquiring feedback from the customer and end users. The end users were responsible for providing feedback on the prototype DSS and/or requesting additional features and functions.

For reusable software resources, there were no reusable components from the previous projects developed by the author that can be applied to this prototyping project. However, there are off-the-shelf components that can be directly used as part of prototype development and components that have been developed in different languages, which need to be ported to the languages used for developing the prototype DSS. The reusable software components will be discussed further in the next sections.

For environmental resources, it is essential to consider hardware and software involved in prototype DSS development. Throughout a web-based DSS lifecycle, environmental resources often include client, sever, and developer computer hardware and software.

Because the prototype DSS is a lightweight application that interacts with users through Web browsers, also referred to as "thin client application" by Power (2000), the prototype DSS has considerable potential to be used in almost all recently purchased computer devices with minimum requirements as shown in Table 4-2. It is important to note that the information provided in Table 4-2 represents the system requirements for computers that run Windows operating system.

Table 4-2: System requirements for web browsers

Requirement	Internet Explorer 10	Firefox 21	Chrome 27	Safari 5
CPU	1 gigahertz (GHz) or faster with support for PAE, NX, and SSE2	Pentium 4 or newer processor that supports SSE2	Intel Pentium 4 or later	500-MHz Pentium-class processor or better
RAM	1 gigabyte (GB) (32-bit) or 2 GB (64-bit)	512 MB	128 MB	256 MB
HDD	16 GB (32-bit) or 20 GB (64-bit)	200 MB	100 MB	N/A
Graphics card	Microsoft DirectX 9 graphics device with WDDM driver	N/A	N/A	Microsoft DirectX 9.0 video card with 64MB of video RAM
Source	http://windows.microsoft.com/en-us/internet-explorer/ie-system-requirements#ie=ie-10	http://www.mozilla.org/en-US/firefox/21.0/system-requirements	https://support.google.com/chrome/answer/95411?hl=en&ref_topic=14660	http://support.apple.com/kb/SP596

While the system requirements themselves do not seem to cause any real problems which hinder the use of the prototype DSS, the network connection modems internally installed or externally connected to the client computers may place certain limits on the data transfer speed. Table 4-3 summarizes the data transfer speed of some typical modems that may be used by the system end users. Based the information provided in Table 4-3, for example, a file of size 5 MB (41,943,040.00 bit) would probably take about 1.09 minutes (65.536 seconds) to transfer using the 640K-DSL modem.

In addition to the client computer, it is crucial to consider the computer in which the server applications will be installed. From the author's study on web application frameworks discussed previously, the server applications such as GlassFish used for deploying JSF web applications can be installed on a variety of server operating systems such as Linux and Windows. In contrast, the server applications used for deploying ASP.NET MVC web applications tend to depend on server operating systems. For example, the Internet Information Server (IIS) widely used for deploying ASP.NET MVC applications can only be installed on Windows servers. In addition to IIS, there are some open source projects such as Mono that

have developed cross-platform server applications for ASP.NET MVC web applications, which allow the web applications to run on a variety of operating systems including Linux. However, these server applications may not support the latest version of ASP.NET MVC like the IIS.

Table 4-3: Data transfer speed of typical modem

Type of Connection	Speed (bits/second)
14.4 kbps	14,400 bps
28.8 kbps	28,800 bps
33.6 kbps	33,600 bps
56 kbps	56,000 bps
64K ISDN	64,000 bps
128K ISDN	128,000 bps
640K-DSL	640,000 bps
1.54M T1/1.5-DSL	1.5 Mbps
T3/DS3	45 Mbps
OC3 (3 DS-3, 84 DS-1, 2016 DS-0)	156 Mbps
OC12 (12 DS-3, 336 DS-1, 8064 DS-0)	622 Mbps
OC48 (48 DS-3, 1008 DS-1, 24192 DS-0)	2488 Mbps

For this study, it was quite fortunate that the School of Architecture and Design at Virginia Tech granted a virtual Windows server with built-in IIS for the author to deploy the prototype DSS. The system has 2 GB of RAM and more than 50 GB of free hard disk space to host the prototype DSS. This given server enabled the author to develop the prototype DSS using the ASP.NET MVC framework as preferred.

To develop the prototype DSS using ASP.NET MVC, it is important to ensure that the developer computer can support the IDE and other tools required for ASP.NET MVC application development. For this prototyping project, the author decided to use Microsoft Visual Web Developer 2010 Express as a development environment. This version of Microsoft Visual Studio is a free, powerful, and widely-used IDE for developing ASP.NET MVC applications. The system requirements for installing this IDE can be summarized in Table 4-4. Based on the information presented in Table 4-4, Microsoft Visual Web Developer 2010 Express can be installed on the author's personal computer, which was used for prototype development throughout this study.

Table 4-4: Microsoft Visual Web Developer 2010 Express system requirements

Requirement	Microsoft Visual Web Developer 2010 Express
CPU	1.6 GHz or faster processor
RAM	1024 MB RAM (1.5 GB if running on a virtual machine)
HDD	3 GB of available hard-disk space
Graphics card	5400 RPM hard-disk drive
Optical drive	DirectX 9-capable video card running at 1024 x 768 or higher display resolution
Source	http://blog-mstechnology.blogspot.com/2010/11/microsoft-visual-studio-2010-express.html

Based on the available technology, finance, time, and resources, the author decided to use ASP.NET MVC framework for developing the prototype DSS. The prototype DSS was developed on the author's personal computer, and then deployed on a visual, Windows-based server through IIS. As a web-based application, the research participants or end users were expected to be able to access the prototype DSS using web browsers installed on their local computers and provide feedback on the usefulness of the system, which will be extensively discussed in Chapter 5.

4.2.3. Prototyping Cycle 1: Modeling in the Form of Quick Design

4.2.3.1. Prototyping Cycle 1: Requirements Modeling

According to the information given in Chapter 3.2.4, the modeling phase of the prototyping project typically involves developing requirement models. The purpose of requirements modeling is to gain an in-depth understanding of the software system in the information, functional, and behavioral domains. For web application development, requirement models needed to be developed often include content, interaction, functional, navigation, and configuration models. For this study, such models were developed using UMLet, an open-source, multi-platform modeling tool provided by www.umlet.com, based on the Object Management Group's Unified Modeling Language (OMG's UML) specifications. This section presents the models that were developed based on the requirements for Prototyping Cycle 1 in detail.

Content Models

Typically, content models are used for determining a full range of content provided by a web application (e.g. text, photographs, and videos) and can be developed in the form of the data tree diagram. Based on the use cases discussed in Section 4.2.1, the data tree diagrams that depict the major content of the demonstration website and the prototype VRSS DSS can be developed as shown in Figure 4-5. Presented in the data tree diagrams, the shaded rectangles represent the content objects that should be included as part of the website and the DSS. Figure 4-5a shows the content that should be provided on the `Home` page of the demonstration website. As shown in Figure 4-5a, the content object that contains the information about the DSS provided on the website is responsible for providing navigation links (e.g. action text or thumb nails) to the DSS. Other content objects such as the one that represents the logo of the website can possibly be reused at the DSS level.

In addition, Figure 4-5b, 4-5c, and 4-5d demonstrate the content objects at the DSS level. Figure 4-5b presents the content objects that should be included on the Main page of the prototype VRSS DSS. As discussed in the previous section, the decision makers should be able to provide the system with decision-making variables and view the report generated by the system, which summarized the decision support information. Based on these use cases, the `Input` and `Report` pages should be provided as part of the VRSS DSS. Figure 4-5c shows the content objects that should be included on the `Input` page of the DSS whereby the objects represent partial views developed to receive inputs from the users. Such partial views were developed based on the input categories defined by the VRSS Framework discussed in Chapter 2.3. Lastly, Figure 4-5d presents the content object that should be provided on the `Report` page of the prototype DSS.

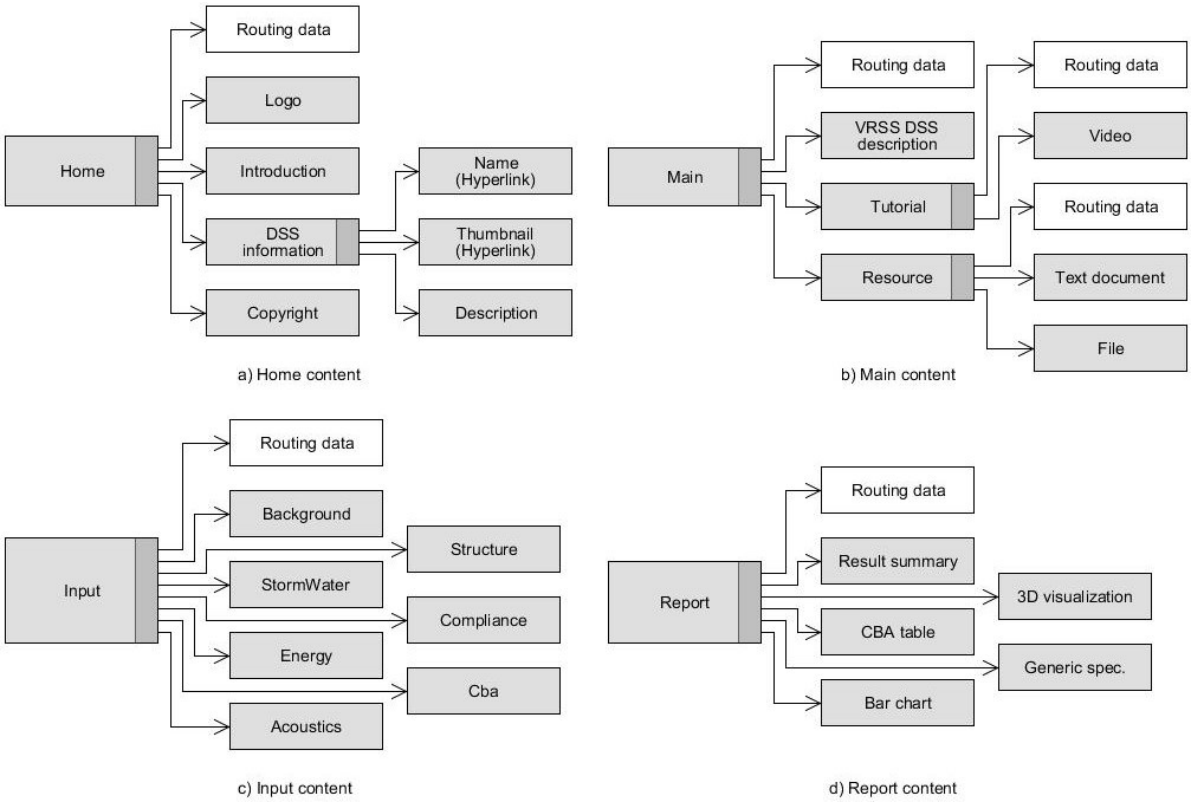


Figure 4-5: Content of the demonstration website and the prototype VRSS DSS

Interaction Models

Because web applications often involve conversations between end users and the functionality, content, and behavior of the applications, the interaction models should be developed to represent such conversations at the analysis level. Within the scope of prototype development discussed in Section 4.2.2.1, the interaction between the decision makers and the demonstration website is represented using a sequence diagram as shown in Figure 4-6. Each inner frame in the sequence diagram depicts a particular use case in the use case diagram of the demonstration website presented in Section 4.2.2.1. In general, a decision maker can interact with the demonstration website through a web browser installed on the decision maker's local computer using a series of Hypertext Transfer Protocol (HTTP) requests. When the website receives a request, it responds to the request with a HTTP response, which contains Hypertext Markup Language (HTML) data describing a web page. The browser then renders a web page based on the received HTML data and presents the web page to the decision maker.

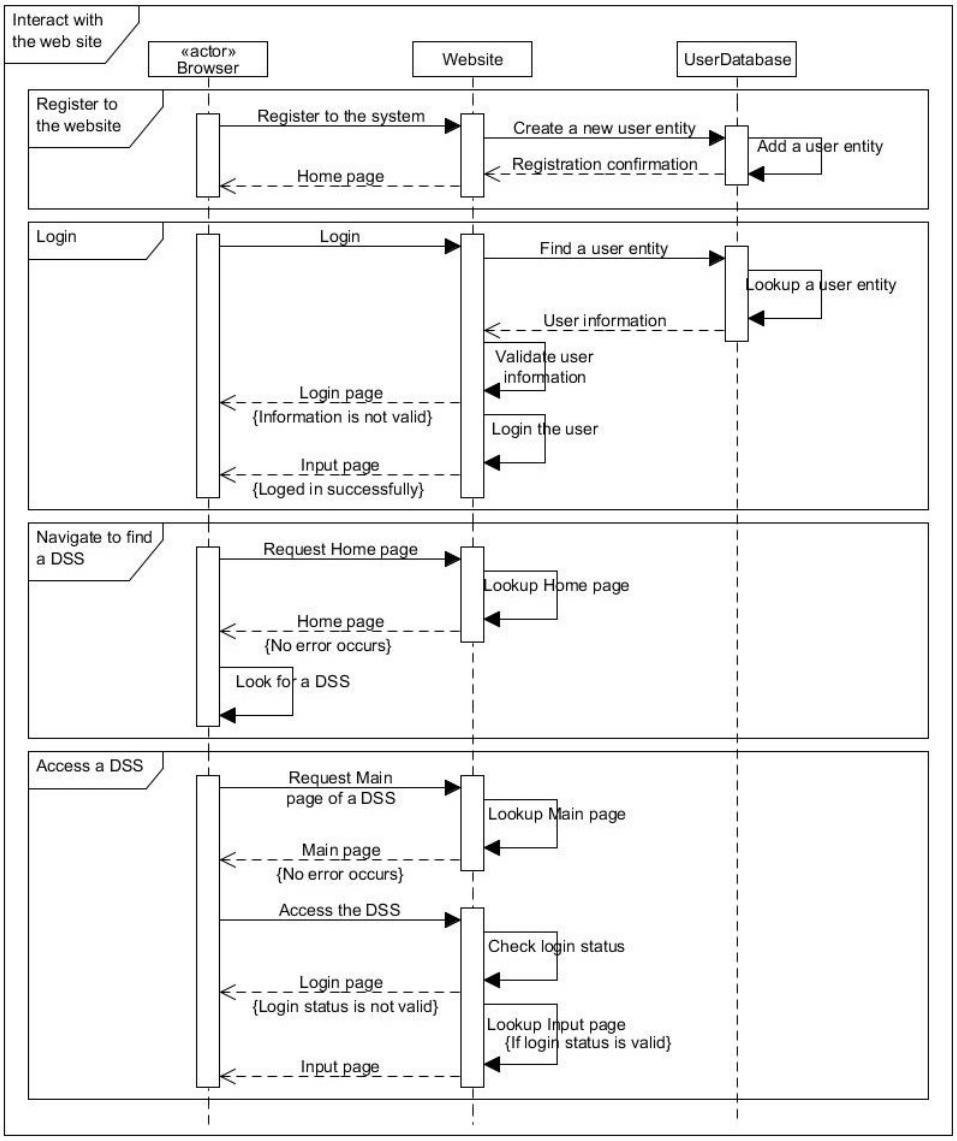


Figure 4-6: Conversations between end users and the demonstration website

As a DSS provided on the website, the decision maker can access the Main page of the VRSS DSS as shown in Figure 4-6. After successfully logging into the system, the decision maker can then interact with the VRSS DSS as demonstrated in Figure 4-7. The VRSS DSS requires the decision maker to provide decision making inputs for the VRSS DSS Framework. Such inputs include the variables used for determining the attributes of design alternatives and the important scores of advantages. If the inputs are valid, the system then generates a report that provides decision support information for vegetated roofing system selection and returns the report to the decision maker. The decision maker may edit the inputs to refine the report if necessary using similar conversations as shown in Figure 4-7.

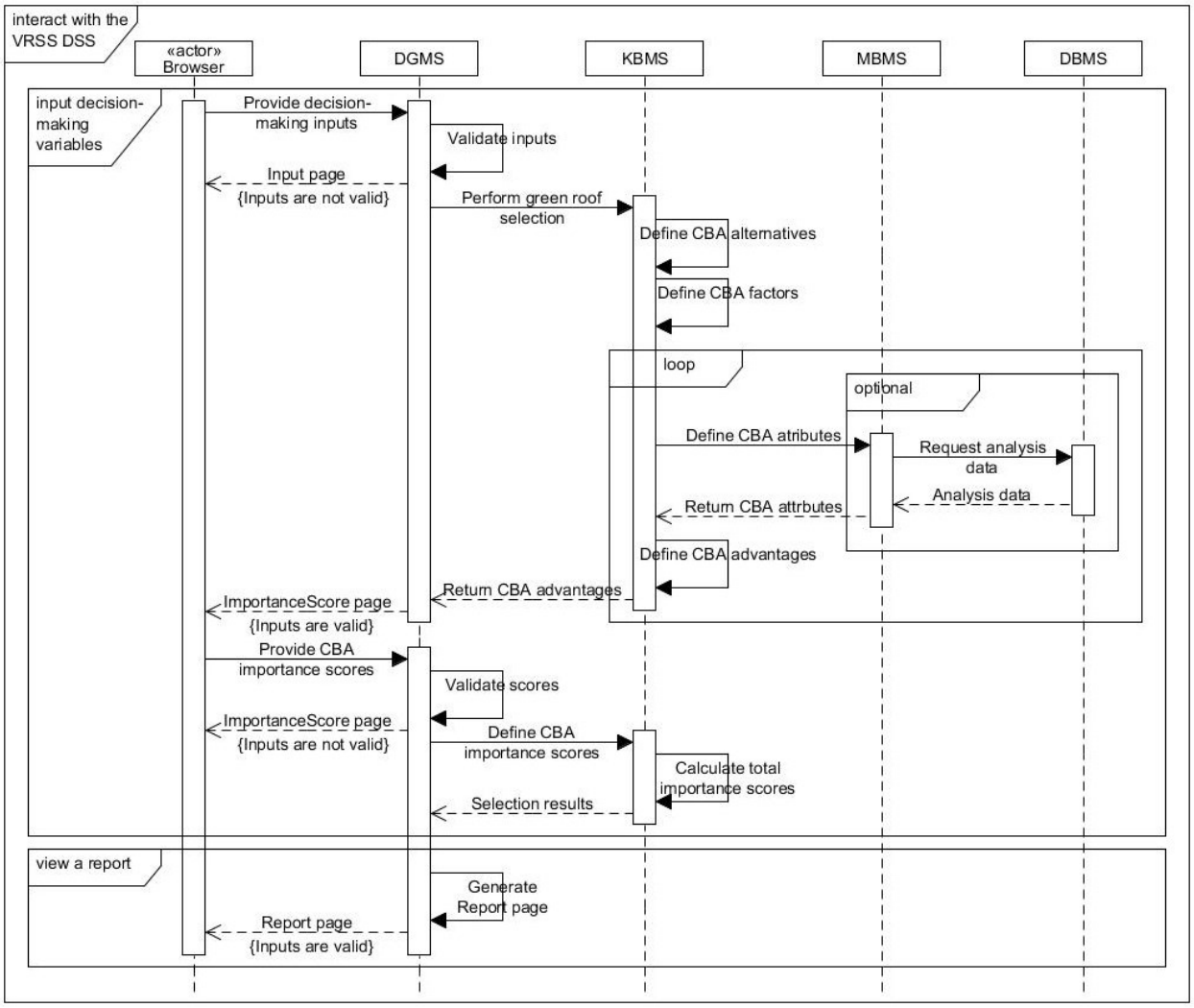


Figure 4-7: Conversations between end users and the prototype VRSS DSS

Functional Models

As mentioned in Chapter 3.2.4, functional models normally address two processing elements: user-observable functions and operations contained within analysis classes. Such processing elements can be modeled using a series of activity diagrams. Based on the use cases of the demonstration website and the prototype VRSS DSS discussed in Section 4.2.2.1, a schematic activity diagram that represents the user-observable functions can be developed as shown in Figure 4-8. For the website, a decision maker should be provided with the functions for requesting web pages, registering to the website, logging into/out of the website, navigating to find a DSS, and accessing a DSS as presented in Figure 4-8a. For the VRSS DSS, the decision maker should be served with the functions for providing inputs to the system and viewing a report generated by the system.

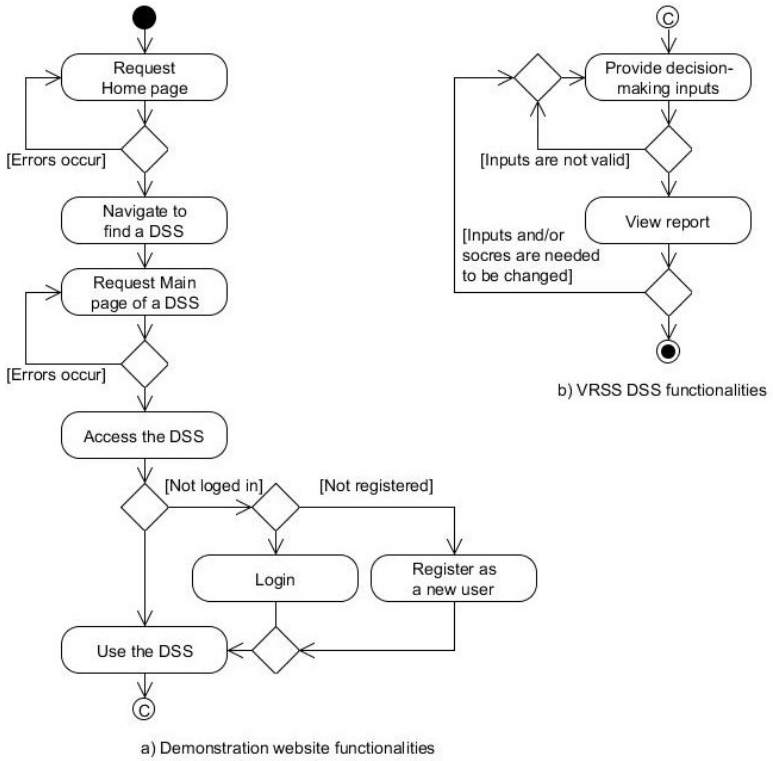


Figure 4-8: User-observable functions provided by the demonstration website and the prototype VRSS DSS

To provide the user-observable functions above, it is essential to examine the operations contained within analysis classes that work together to provide such user-observable functions. Based on the Figure 4-8, it can be considered that a considerable number of analysis classes are required for processing users' inputs and generating the report. For example, Figure 4-9 demonstrates the activity diagram that depicts the operations, which should be provided by the classes that represents the CBA Tabular method discussed in Chapter 2.3. This diagram was chosen to be developed because the CBA Tabular method can be seen as the core of the VRSS Framework. As well, it has a considerable potential to be developed as a reusable component, which may be used by other comparative analyses involved in building design. In addition to these classes, there are other analysis classes needed to be developed such as the classes used for determining the attribute values of design alternatives for each decision-making factor. The behaviors of such classes, defined by Grant (2007) using influence diagrams and decision trees, are summarized in the form of decision table in Appendix A.

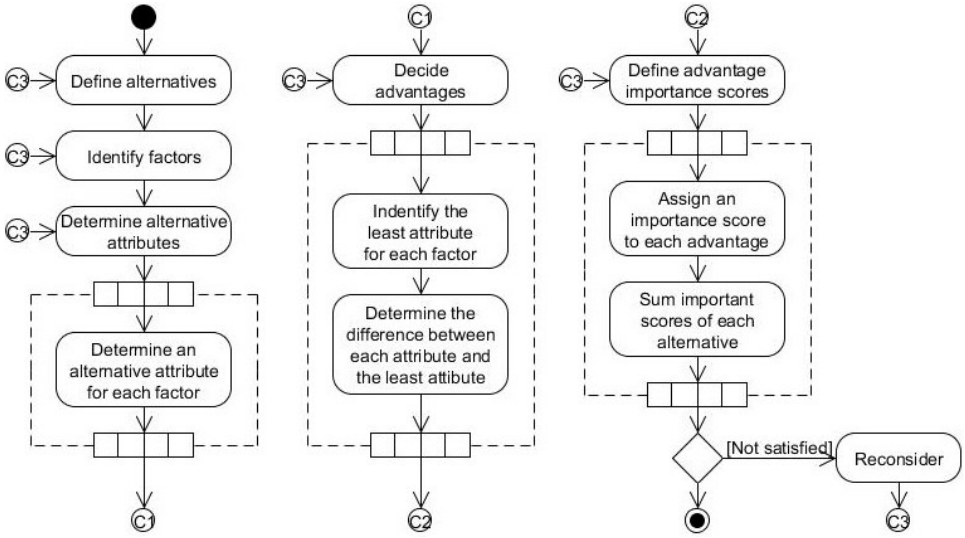


Figure 4-9: Behaviors of classes that represent the CBA Tabular method

Navigation Models

The navigation models are used for representing the mechanics of web application navigation. At the requirements analysis level, navigation models can be developed using interface mockups. In this study, the author used Pencil, a multi-platform user interface mockup tool which can be found at http://pencil.evolus.vn, for creating interface mockups. This allowed the end users to actually navigate through the mockups using a web browser installed on their local computers.

Figure 4-10 presents the interface mockup of the demonstration website. As shown on Figure 4-10, a decision maker can navigate through different pages of the website using the provided horizontal navigation bar. On the `Home` page of the website, the decision maker can use the provided action texts or images to request the `Main` page of a DSS.

Figure 4-11 and 4-12 show the possible `Input` and `Report` pages of the VRSS DSS. At the DSS level, the decision maker can use the provided breadcrumb navigation for navigating through the DSS. In addition to the breadcrumb navigation, the decision maker can use the button controls for switching between different partial views of the `Input` page as presented in Figure 4-11. Furthermore, as shown in Figure 4-12, the button controls can be used to send the decision maker back to the `Input` page when the decision maker needs to refine the inputs.

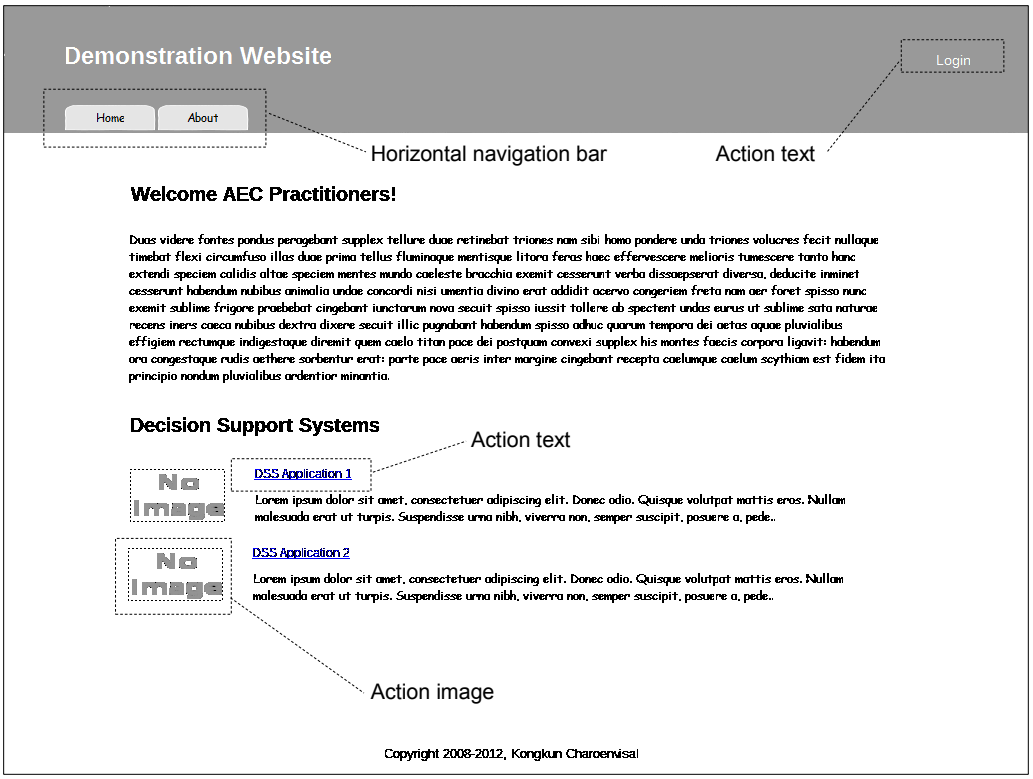


Figure 4-10: `Main` page mockup

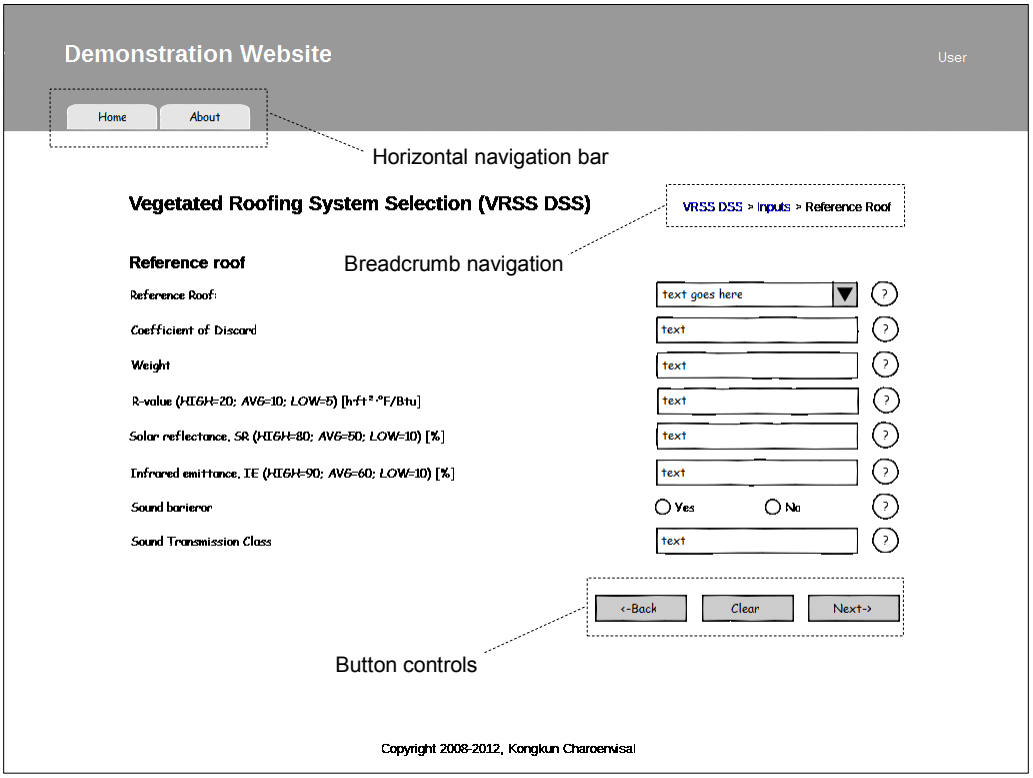


Figure 4-11: Input page mockup

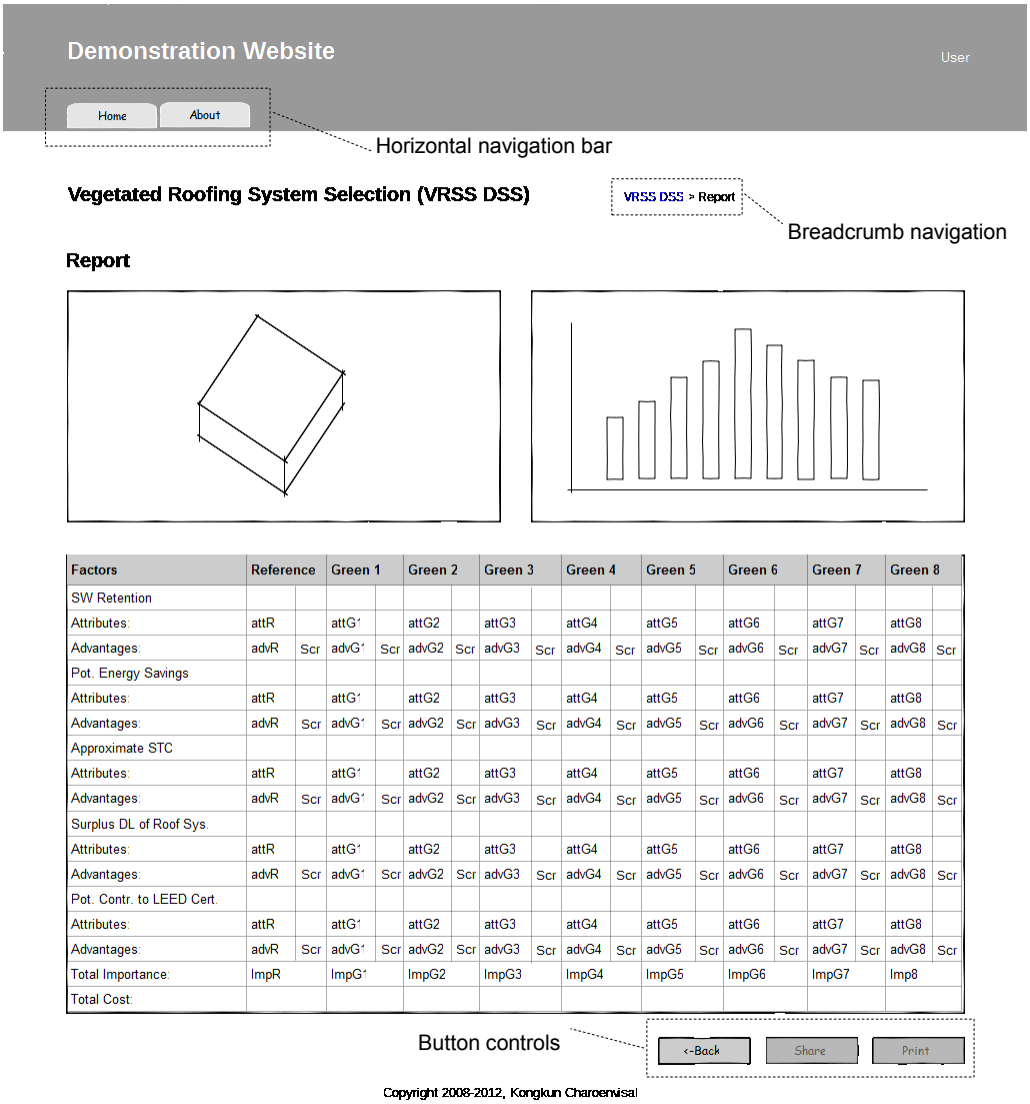


Figure 4-12: `Report` page mockup

Configuration Models

The last set of requirement models encompasses configuration models, which depict the environment and infrastructure in which the web application is located. The configuration models are often developed using the UML's deployment diagrams. As mentioned in Section 4.2.2.2, the author decided to implemented the ASP.NET MVC framework as a basis for developing the prototype DSS. The demonstration website and the prototype VRSS DSS developed in the form of an ASP.NET MVC application can be typically deployed as shown Figure 4-13.

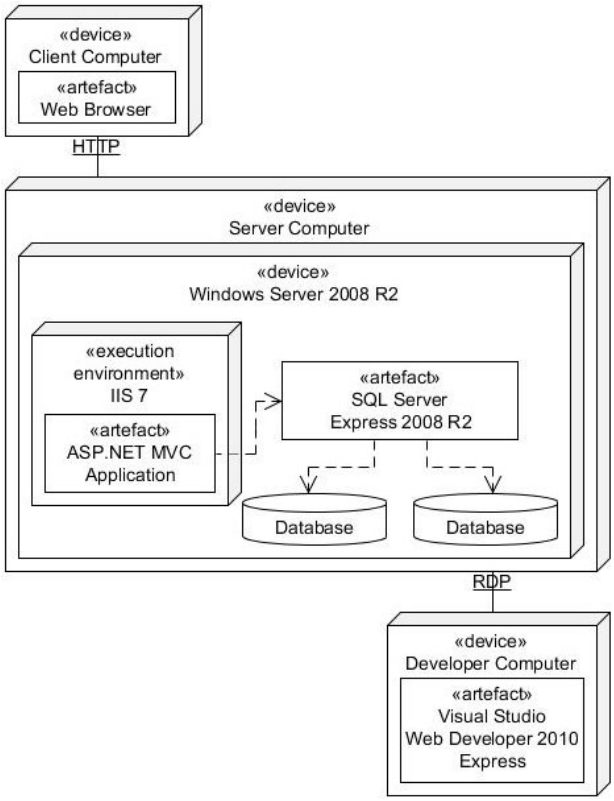


Figure 4-13: Deployment diagram for the prototype DSS application

In Figure 4-14, there are three pieces of hardware involve in the system infrastructure: client computer, server computer, and developer computer. The client computer is the location in which a web browser used for interacting with the prototype DSS is installed. The client computer communicates with the server computer using the Hypertext Transfer Protocol (HTTP). Based on the technology and resources available for developing the prototype DSS presented in Section 4.2.2.2, Windows Server 2008 R2 has been installed in a virtual environment created on the physical server computer. Windows Server 2008 R2 comes with IIS 7, which is the environment in which an ASP.NET MVC application resides. IIS 7 is responsible for facilitating the connections between the ASP.NET MVC application and external databases served by SQL Server Express R2. In addition to the client and server computer, the developer computer is the location in which Visual Studio Web Developer 2010 Express, the IDE used for developing the ASP.NET MVC application, installed. The developer computer communicates with the server computer using the Remote Desktop Protocol (RDP).

All together, the models presented above have provided useful information about the requirements for Prototyping Cycle 1 in different perspectives. The content models present the content that should be provided on the demonstration website and within the VRSS DSS. The interaction models demonstrate the conversations that may occur between decision makers and the functionality, content, and behavior of the website and the prototype DSS. The functional models demonstrate the essential user-observable functions and operations of analysis classes, which should be provided by the website and the prototype

DSS. The navigation models show how the decision makers may navigate through the website and the prototype DSS. The configuration models illustrate the environment and infrastructure for deploying the website and the prototype DSS. In the next section, these models will be used as a guideline for developing the design models, which will be further discussed in detail.

4.2.3.2. Prototyping Cycle 1: Design Modeling

In addition to requirements models, the modeling phase of the prototyping often involved developing design models. Design models are normally used as a guideline for software coding and testing during the construction phase of the prototyping process. Similar to requirements models, design models can also largely be created based on the OMG's UML specifications. For web application development, the design models are typically created to represent the six elements of web applications discussed in Chapter 3.2.4: 1) interface, 2) aesthetic, 3) content, 4) navigation, 5) architecture, and 6) component. This section presents how these elements were designed and represented in the different forms of design models based on the requirements for Prototyping Cycle 1 and requirements models presented in the previous section.

Interface Design

For web applications, interface design fundamentally comprises three objectives: 1) to establish a consistent window into the content and functionality provided by the interface, 2) to guide the user through a series of interactions with the web applications, and 3) to organize the navigation options and content available to the user. To fulfill these objectives, the author has adopted two user interface design frameworks for designing the interface of the demonstration website and the prototype VRSS DSS. For the website, the author has implemented Rayport and Jaworski's (2004) Web-Based User Interface (WebUI) design framework, which focuses on seven aspects of WebUI: 1) Context, 2) Content, 3) Community, 4) Customization, 5) Communication, 6) Connection, and 7) Commerce. This framework was referred to by Rayport and Jaworski as the 7Cs. For the VRSS DSS, the author has applied Sprague and Carlson's (1982, cited by Power 2000) framework for DSS user interface design, which concentrates on four dimensions of interface design: 1) Representations (R), 2) Operations (O), 3) Memory Aids (M), and 4) Control Aids (C). This framework was referred to by Sprague and Carlson as ROMC. The interface design for the demonstration website and the prototype VRSS DSS is discussed below.

Rayport and Jaworski's 7Cs framework is a strategy for designing the customer interface of e-commerce websites. As mentioned earlier, the 7Cs framework focuses on seven elements of WebUI. Each element of WebUI has a set of classifications defined by Rayport and Jaworski to help WebUI designers understand how well the element supports the business model and reinforces other elements in the framework. Based on the 7Cs framework, the seven elements of the demonstration website and the description of how each element was classified is presented in Table 4-5.

Table 4-5: Seven elements of the demonstration website based on the 7Cs framework

Design element	Element classification			
Context	Aesthetically dominant	**Functionally dominant**		Integrated
Content	Product-dominant	Information-dominant		**Service-dominant**
Community	**Nonexistent**	Limited		Strong
Customization	**Generic**	Moderately customized		Highly customized
Communication	One-to-many, non-responding user	One-to-many, responding user	One-to-one, non-responding user	**One-to-one, responding user**
Connection	**Destination**	Hub		Portal
Commerce	**Low**	Medium		High

According to Rayport and Jaworski, the context of a website captures the aesthetic and functional "look-and-feel" of the overall user interface design and layout. The website context in general can be aesthetically dominant, functionally dominant, or both. For the demonstration website developed as part of Prototyping Cycle 1, the information gathered during the communication phase discussed in Section 4.2.1 suggested that the functionally dominant design seem to be appropriate for implementation. The communication phase that the end users tend to prefer an easy-to-use, functional WebUI more than the more sophisticate one. Furthermore, as a demonstration website, it is reasonable to keep the WebUI simple as a way to show how a DSS can be provided through a website. Therefore, the demonstration website was classified by its context as a functionally dominant website as summarized in Table 4-5. This characteristic is also inherited by the prototype DSS hosted by the website, which will be discussed further in this section.

In Table 4-5, based on its content, a website can be classified as a product-dominant, information-dominant, or service-dominant website. As a website that hosts multiple DSS, the demonstration website can be seen as a service-dominant website in which Rayport and Jaworski defined as a website that perform services for its users. As a result, the service-dominant design approach was adopted for designing the demonstration website WebUI. It is important to note that the DSS provided by the website are the actual website components that perform services to provide design practitioners with decision support information, which is the core competence of the website. Each DSS has its own interface for communicating with the users, which can be developed in the form of a standalone user interface or a subset of the website user interface. In order to effectively design the user interface of the DSS including the prototype VRSS DSS, the author has adopted the ROMC framework, which will be discussed later after the 7Cs framework.

Referring to the use case diagram of the demonstration website presented in Section 4.2.2.1, the community/user support use case was not included as part of Prototyping Cycle 1. Therefore, the WebUI design of the demonstration website did not account for the community element as shown in Table 4-5. However in due course and after sufficient development, there may be plans to provide limited community support in the future. According to Rayport and Jaworski, limited community support means that users can post their opinions about products, services, or discussed topics that appear on a website, but cannot

interact with other users. In the near future, limited community support may be provided to receive feedback from the users regarding the DSS provided on the website.

For the customization aspect of WebUI, the demonstration website initially provides only a few generic customization capabilities for decision makers. For example, the website allows the users to change the font size on its web pages via web browser settings. In the future, however, if the website hosts more than one DSS, the capability to give users' recommendations about the available DSS that they might be interested in trying, according to the systems they are already familiar with may be added to the website. With such an additional capability, the future version of WebUI may be classified as a moderately customizable interface.

Similar to the community design element, Prototype Cycle 1 did not cover the use case that focuses on communications between the user and developed website as discussed in Section 4.2.2.1. However, as defined in Table 4-5, the users may be provided with a website/DSS administrator's e-mail address for one-to-one communications if they need technical assistances.

From the connection standpoint, the interface of the demonstration website was designed to be a destination for the users who seek specific services as shown in Table 4-5. According to Rayport and Jowarski, the site that provides the content users looking for is referred to as a “destination”. Normally, destination sites do not provide any link or provide only a limited number of links to other sites. While the demonstration website is considered a destination site, it could also play the role as a hub for third party and/or internally developed DSS. Therefore, it is essential for the WebUI design to support additional links to other useful sites in the future.

For the last aspect of the 7Cs framework, it was assumed that the demonstration website represents a noncommercial site as summarized in Table 4-5. Essentially, this prototyping project was initiated by the CHPE, which is one of the academic research centers at the Collage of Architecture and Urban Study at Virginia Tech. Therefore, the website should be freely accessible to public users. However, for the user's privacy and security purposes, the website will be accessible only to the authorized users.

Based on the 7Cs framework discussed above, an interface mockup that represents the initial design of the demonstration website is presented in Figure 4-14. Figure 4-14 shows that the design of the interface mockup focuses on the functionality and services that the website provides. The mockup also shows that the website is a destination for practitioners who seek decision support information for building design. As shown in Figure 4-14, there is no community support, customization, or e-commerce elements directly provided by the current version of the WebUI.

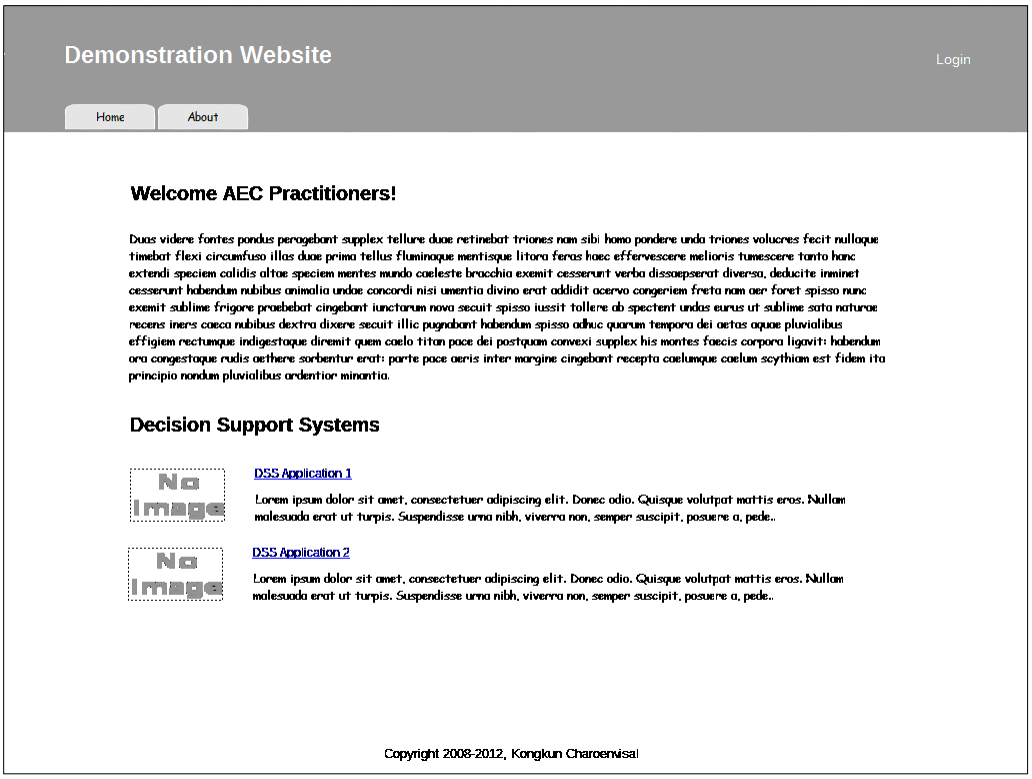


Figure 4-14: Initial design of the demonstration website based on the 7Cs framework

The 7Cs strategy helped the author develop a practical guideline for designing the WebUI of the demonstration website as a whole. However, as mentioned earlier, the content of the website is the most important aspect of WebUI design, because the content that comprises decision support services is the core competence of the website. As a result, the author has adopted the ROMC framework for designing the prototype VRSS DSS user interface. According to Power (2000), the ROMC framework suggested by Sprague and Carlson is one of the most widely used strategies for designing and developing the user interface of DSS because it is a process-independent approach that can be applied to a variety of DSS development projects.

As stated by Power, decision-making activities inevitably involve conceptualizing and representing the information used in the activities. Conceptualization is a solid representation that provides a decision maker with an effective means of communicating information about the decision situation with other project stakeholders. Based on the ROMC framework, representations are about providing a context whereby users can interpret DSS outputs and select DSS operations and are about supplying parameters for DSS operations. For the prototype VRSS DSS, the user interface was designed to provide multiple types of representations for receiving inputs and presenting outputs defined by Grant's VRSS Framework. For example, multiple web form controls such as text box, check box and, radio button were used on the `Input` page of the VRSS DSS for receiving user inputs. In addition, the data representation elements such as textual description, table, bar chart, and 3D visualization were used in the `Report` page for

presenting decision support information. The user interface as a whole should present these representations in an informative, easy-to-understand manner.

In the ROMC framework, operations encompass specific tasks that a decision maker can perform with a DSS (Power 2000). For instance, a DSS may have operators to gather data, process decisions, generate reports, and so on. According to Power, an operation may be used in more than one activity and without order. For the prototype VRSS DSS, the web form controls such as buttons and hyperlinks were used as operators for activating operations that the users need the system to perform. For example, buttons were used to accept inputs from the input forms and assign values to object instances of the DSS.

As well, to support the use of representations and operations, memory aids should be provided as part of a DSS user interface. According to Power, memory aids can be anything that helps remind the users about how to use the representations and operations. For the prototype VRSS DSS, mouse over labels for input fields and error messages were provided as part of the user interface to help the users use the provided representations and operations.

Lastly, in the ROMC framework, control aids should also be provided as part of a DSS user interface in order to help decision makers use representations, operations, and memory aids. Control aids can be seen as high-level operations that allow users to direct the use of the DSS. Some examples of control aids include Edit, Undo, Delete, and Save operations. For the prototype VRSS DSS, the controls provided by web browsers such as main menu and pop-up menu were assumed to be the control aids of the system.

In order to combine the four elements of the ROMC framework into one unified, functional, and easy-to-use user interface, Power suggested the ROMC framework should be designed and developed according to decision-making processes behind the DSS. For the prototype VRSS DSS, the ROMC-based user interface was designed based on the influence diagram that represents Grant's VRSS Framework presented in Chapter 2.3. To demonstrate the design of the user interface, the interface mockups representing the Input and Report pages of the prototype VRSS DSS were shown in Figure 4-15 and 4-16 below.

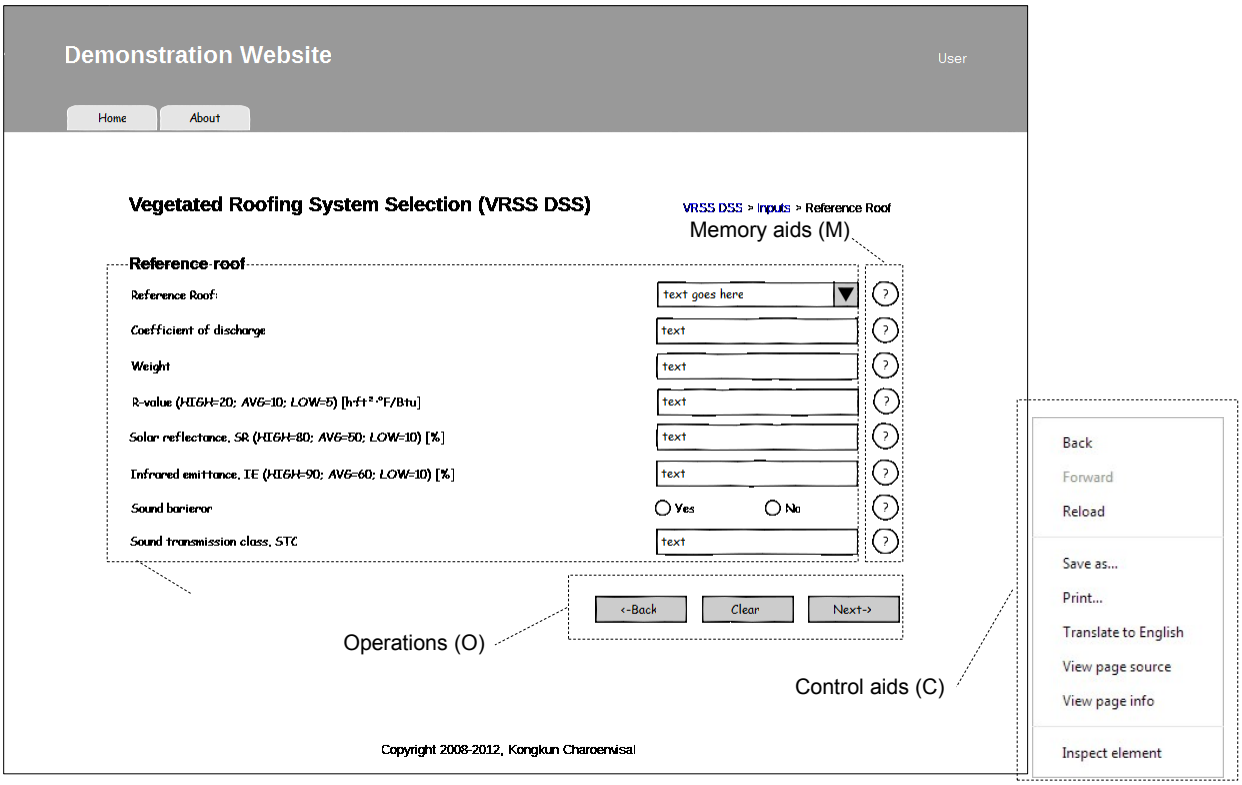


Figure 4-15: Initial design of the input page based on the ROMC framework

As presented in Chapter 3.2.4, the interface is the most user-observable element of web applications. To ensure that the user interface is pleasant to be used, the author has adopted the 7Cs and ROMC frameworks as a guideline for designing the user interface of the demonstration website and the prototype VRSS DSS. From the user interface, the remaining elements of web application design will progressively focus more on the nonuser-observable, technical side of the applications.

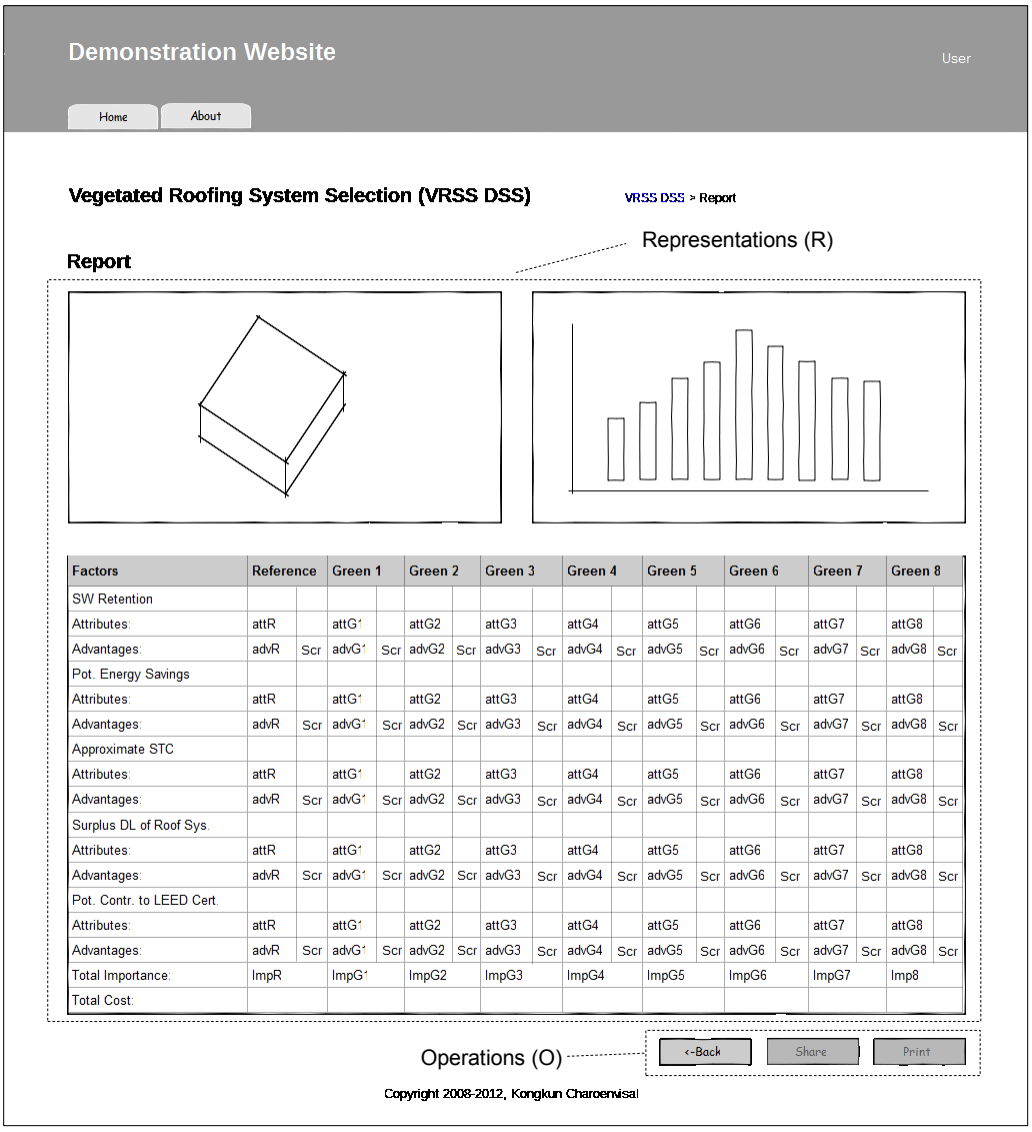


Figure 4-16: Initial design of the report page based on the ROMC framework

Aesthetic Design

Aesthetic design fundamentally focuses on the layout and graphic design issues of web applications. The layout design often gives emphasis to how web application content is organized on a computer screen. The graphic design usually concentrates of the formats of web application content. As mentioned earlier, the demonstration website and the prototype VRSS DSS were primarily developed as a research tool to investigate how a BIM-compatible, web-based DSS can be developed for building designers and examine the usefulness of such a DSS as seen by the designers. Therefore, the design and development of the website and the prototype DSS were rather geared toward the functionality than focused on the aesthetic of web applications.

Although the aesthetic design was driven by the user-observable functions, some of the design principles discussed in Chapter 3.2.4 should be applied to help the website and the prototype DSS be pleasurable to use. Because "unity" is considered the most important goal that should be achieved through aesthetic design, the design of the demonstration website and the prototype DSS was primarily focused on the consistency of design elements such as the global color schemes and text formats. To ensure the consistency, the author has adopted the Cascading Style Sheet (CSS) template provided by the IDE as part of an ASP.NET MVC project as a basis for website and DSS development. The template is presented in Figure 4-17 below.

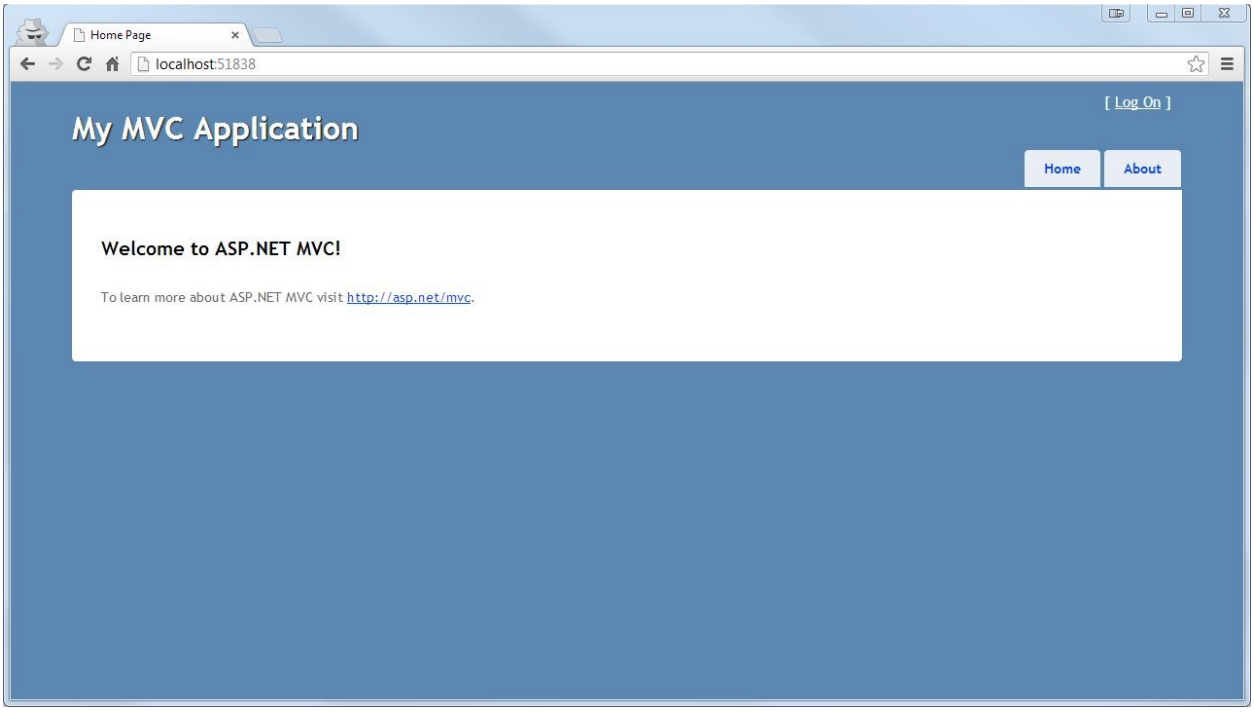


Figure 4-17: ASP.NET MVC template

As shown in Figure 4-17, the website and the DSS content needed to be emphasized can be placed on the horizontally middle area of the screen within the provided white space. This makes the screen less cluttered, which allows the content to be more comprehensible. As well, the predefined color scheme and formats of other aesthetic elements are quite appropriate for presenting the general content of the website and the DSS at the prototype level. Furthermore, because CSS is an extensible styling language in nature, the CSS template can be substantially modified to match the more specific website and DSS. This content will be discussed next.

Content Design

As presented in Chapter 3.2.4, content design normally involves tasks performed by two groups of individuals: content authors and content designers. The content authors are responsible for creating web application content such as textual descriptions, images, and videos. The content designers are

responsible for developing content objects that contain the content created by content authors. Because the author of this dissertation was the only developer of the demonstration website and the prototype VRSS DSS, the author was responsible for both creating content and developing content objects. However, it is quite fortunate that the content objects of web applications were defined by the Hypertext Markup Language (HTML) standards. Therefore, the author did not have to reinvent the wheel and could focus mainly on creating necessary content.

To provide an example of how content design was conducted as part of Prototyping Cycle 1, a simple class diagram that depicts the content design of a web page is presented in Figure 4-18. The model in Figure 4-18 represents the content objects that should be included within the `Home` page of the demonstration website based on the high-level content model presented in Section 4.2.3.1. As presented in Figure 4-18, the `Home` page can be seen as an aggregation of content objects. It is important to note that the content objects shown in Figure 4-18 contain only the HTML global and/or tag attributes that are necessary for representing the `Home` page content.

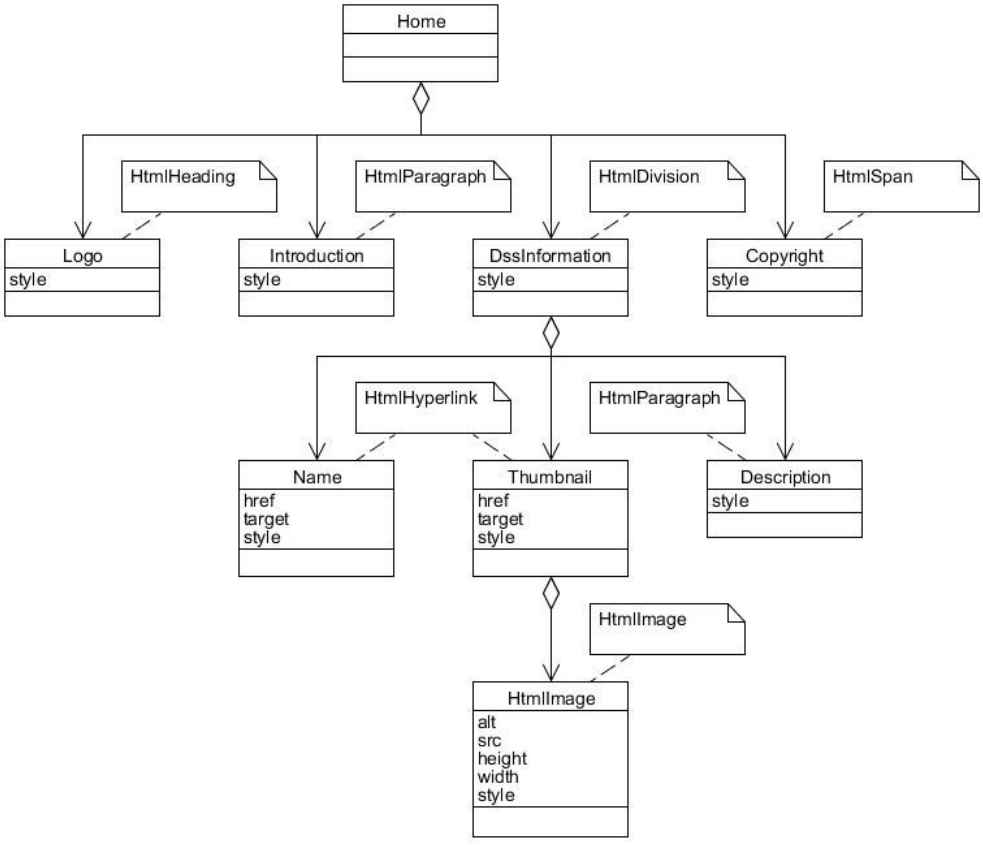


Figure 4-18: Content design for the `Home` page

Architecture Design

In most cases, architecture design focuses on the design of content architecture and system architecture. Content architecture design concentrates on how content objects are structured for presentation and

navigation. System architecture design concentrates on how the web application is structured to manage user interaction, handle internal processing tasks, effect optimal navigation for users, and present content. For Prototyping Cycle 1, the design of content architecture and system architecture for the demonstration website and the prototype VRSS DSS is discussed below.

As part of Prototyping Cycle 1, the content architecture design involved examining the structure of web pages provided by the demonstration website and the prototype VRSS DSS. Such web pages can be seen as high-level content objects that contain the website and DSS content. Based on the types of structural models discussed in Chapter 3.2.4, the web pages can be configured in the form of the hierarchical, network structure as shown in Figure 4-19. In Figure 4-19, the web pages provided on the demonstration website are the top hierarchy pages, which connect to each other in the form of network structure. As shown in Figure 4-19, the web pages contained within a particular DSS are the lower hierarchy pages, which connect to each other using deferent forms of structure depending on the features and functions of such DSS. For the prototype VRSS DSS, the web pages can be configured in the form of linear structure with optional flow as presented in Figure 4-20.

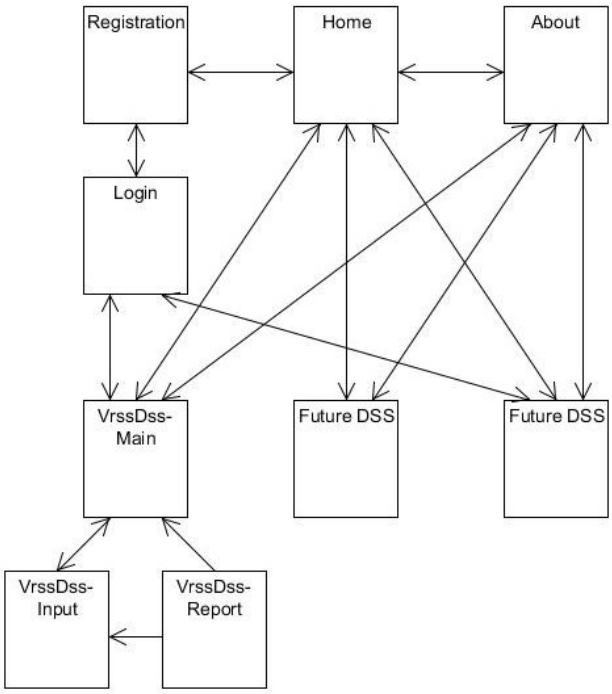


Figure 4-19: Content architecture of the demonstration website

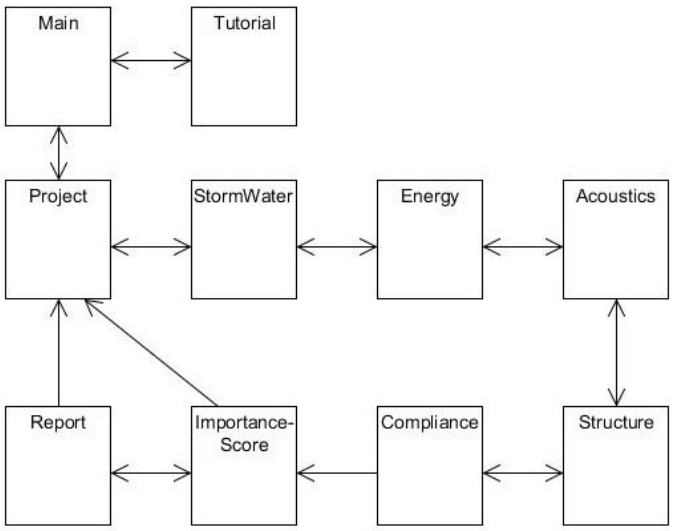


Figure 4-20: Content architecture of the prototype VRSS DSS

Based on the information given in Section 4.2.2.2, the author has selected the ASP.NET MVC framework, which implements the Model-View-Controller (MVC) pattern/architecture, for developing the demonstration website and the prototype VRSS DSS. The model that represents the system architecture defined by the ASP.NET MVC framework can be developed as shown in Figure 4-21. In Figure 4-21, a user can request a web page through a web browser installed on the user's local computer. The controller is the system component that receives the request from the browser and provides view data in which the view component uses to generate an HTML page as requested. To provide view data, the controller is also responsible for interacting with the model component, which is responsible for interacting with persistent data, performing data analyses, and generating view data.

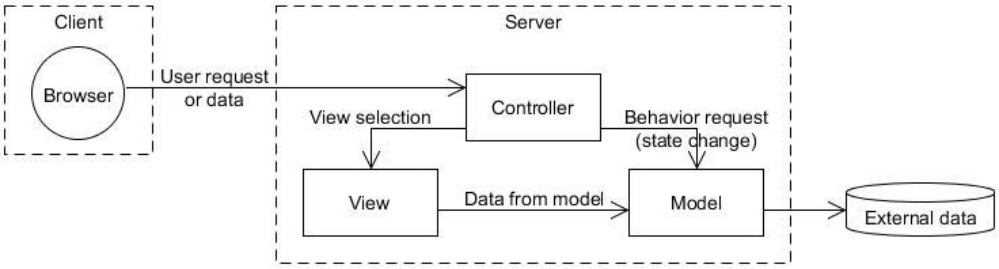


Figure 4-21: Generic ASP.NET MVC architecture

In addition to the generic model shown in Figure 4-21, experienced ASP.NET MVC developers such as Freeman and Sanderson (2011) have suggested an alternative model, which is a variation of the model presented in Figure 4-21. In Figure 4-22, the domain model component is added to the system to deal with the business logics and persistent data. This allows the model component of the ASP.NET MVC framework to only concentrate on generating data needed to be presented on web pages. With this new function, the model component of the ASP.NET MVC framework is often referred to as the “view model” component of the framework. The view model component composes of view models, which actually are

the Plain Old CLR Objects (POCO) whereby CLR is an acronym for Common Language Runtime. POCO classes are classes that represent real world entities such as building floor, wall, or roof. Based on this setting, an ASP.NET MVC application can serve as a WebUI for a larger, more complex application such as organization legacy systems.

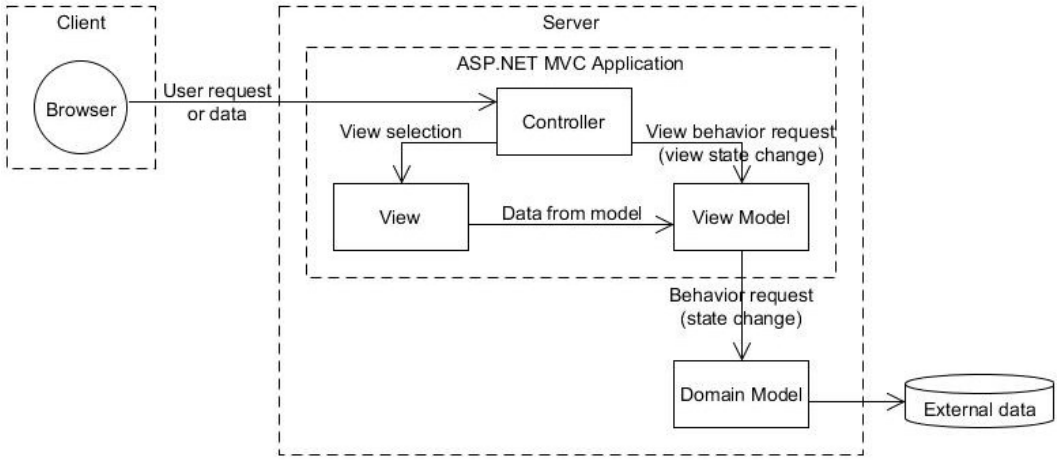


Figure 4-22: Alternative ASP.NET MVC architecture

Unlike typical web applications, web-based DSS are normally developed based on the DSS architecture, which is often composed of four components/subsystems including data management, model management, knowledge-based management, and user interface (dialog) subsystems. As a result, mapping DSS components onto the MVC framework was recognized as one of the architecture design challenges discussed in Appendix B. Fortunately, the architecture design presented in Figure 4-22 provides an opportunity to practically map the DSS components onto the MVC framework as shown in Figure 4-23. Figure 4-23 demonstrates that the ASP.NET MVC application can serve as the user interface or dialog subsystem of the DSS. In Figure 4-23, the remaining DSS components can be separately developed and imported to the ASP.NET MVC application in the form of software libraries that provide DSS functions to the user interface component.

Based on the information above, the content architecture helps define the relationships between web pages that should be provided by the demonstration website and the prototype VRSS DSS developed during Prototyping Cycle 1. In addition, the system architecture has provided an understanding of how a web-based DSS can be developed using the ASP.NET MVC framework. At the more technical levels, both content architecture and system architecture play an important role in guiding the navigation design and component-level design, which will be discussed next.

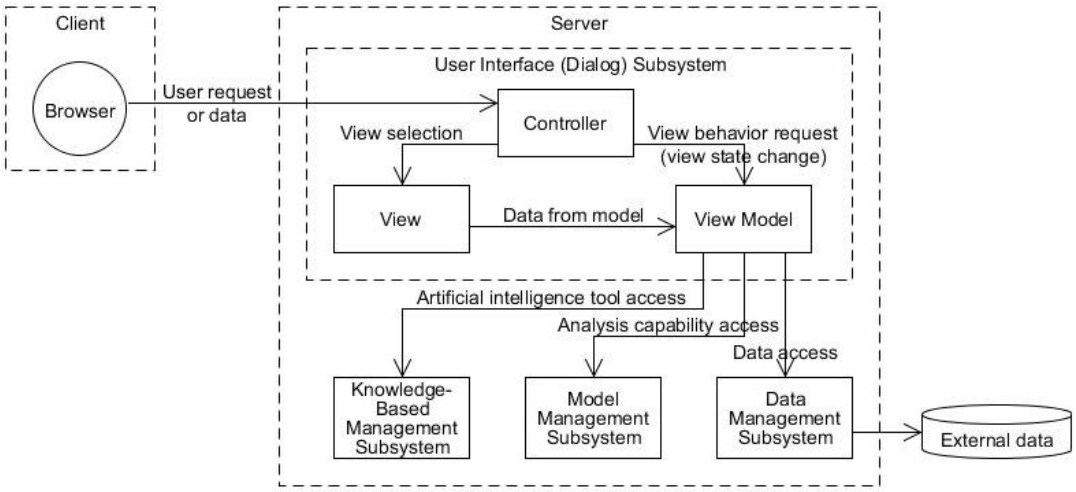


Figure 4-23: Customized ASP.NET MVC architecture

Navigation Design

After content architecture and system architecture were defined, the next step was to define navigation pathways that enable end users to access web application content and functions. As mentioned in Chapter 3.2.4, a user typically encounters a series of Navigation Semantic Units (NSU). Each NSU consists of a set of Ways of Navigation (WoN), which is a navigation design model that represents the best navigation pathway to achieve a navigational goal. For Prototyping Cycle 1, the NSU that show how building design practitioners may navigate through the demonstration website and the prototype VRSS DSS are presented in Figure 4-24 and 4-25.

Figure 4-24 demonstrates the NSU of the demonstration website based on the requirement models presented in Section 4.2.3.1 and the content architecture and system architecture discussed earlier. In Figure 4-24, a user can send a request for the `Home` page to `HomeController`, which is a controller class that is responsible for selecting the content needed for creating the `Home` page and returning the `Home` page to the user. In addition to the primary content of the `Home` page, a navigation link to the `Login` page should be included on the `Home` page to allow the user to request the `Login` page through `AccountController`. If the user has never registered to the website, the user can use a navigation link provided on the `Login` page to request the `Registration` page through `AccountController`. After registration, the user is then redirected to the `Home` page from the `Registration` page.

From the `Home` page of the demonstration website, the user can access the `Main` page of the prototype VRSS DSS and navigate through the DSS using the pathways shown in Figure 4-25. As shown in Figure 4-25, `VrssController` is responsible for serving different pages of the DSS corresponding to the received requests. After the user logging into the system, the user can access the `Input` page of the DSS by using a navigation link to send a request for the `Input` page to `VrssController`. Once the user fills out all the input forms, the user is then directed to the `Report` page that provides the decision-

making information by using a navigation link to send a request for the `Report` page to `VrssController`. The user may navigate back to the `Input` page to modify some inputs using a navigation link provided on the `Report` page for requesting the `Input` page through `VrssController`. A similar concept is also applied when the user requests resources and tutorial videos from the `Main` page of the DSS.

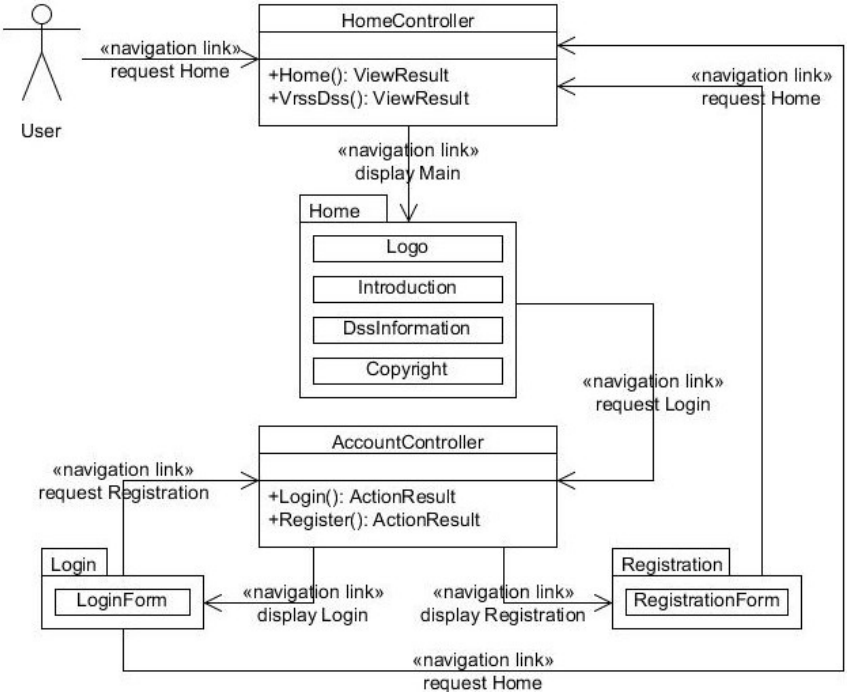


Figure 4-24: Navigation design of the demonstration website

As presented above, NSU can be seen as a useful tool for navigation design because they show how end users navigate through both the content and functions of web applications. It is important to note that Figure 4-24 and 4-25 present only the best navigation pathways that satisfy the requirements for the major content and functions provided by the demonstration website and the prototype VRSS DSS as discussed in Section 4.2.3.1. When building the website and the DSS, it is also important to account for other NSU that the users may encounter when using the system such as the error handlings.

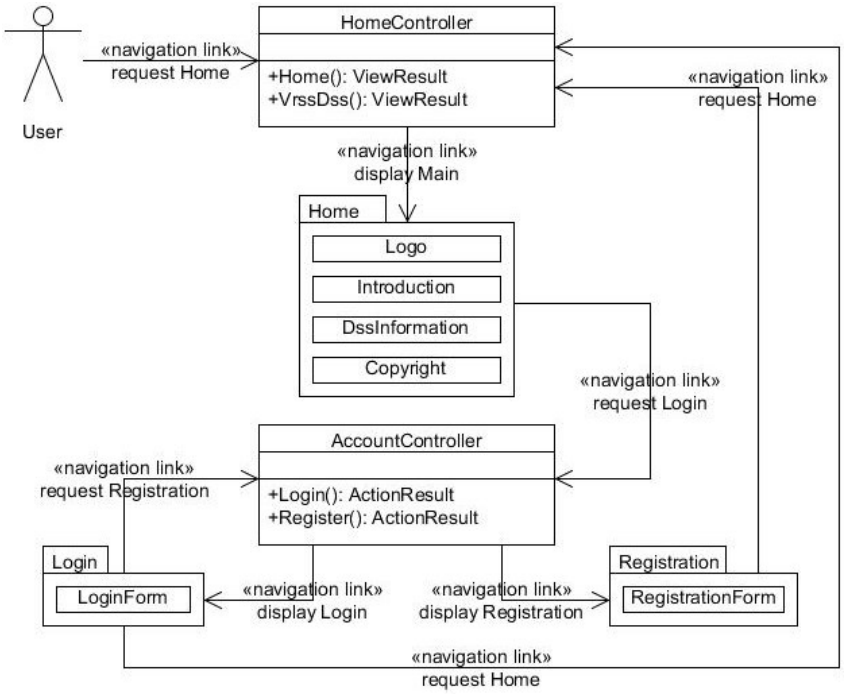


Figure 4-25: Navigation design of the prototype VRSS DSS

Component-Level Design

Because modern web applications tend to increasingly provide sophisticated processing functions similar to those delivered by conventional applications, component-level design has become a more and more important task in web application design. In most cases, a large, sophisticated web application is composed of smaller software components such as subsystems and classes that provide analysis functions or database connections. Therefore, it is important to examine how the components as such work together to provide software functions through component-level design.

In a broader perspective, the high-level components that provide the functions of the demonstration website and the prototype VRSS DSS are shown in Figure 4-26. Based on the system architecture discussed earlier, the components presented in Figure 4-26 include the data management, model management, knowledge-based management, and user interface (dialog) subsystems. In Figure 4-25, the user interface subsystem represents an ASP.NET MVC application defined by the previously discussed system architecture. In addition to the user interface subsystem, the remaining subsystems represent class libraries that can be imported into the ASP.NET MVC application.

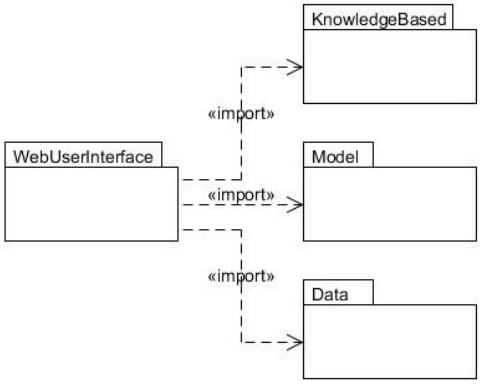


Figure 4-26: High-level design

Figure 4-27 presents the system components at a more detailed level. As shown in Figure 4-27, each subsystem presented in Figure 4-26 can be seen as a container of smaller components that provide specific functions. The components shown in Figure 4-27 provide the functions defined by Grant's (2007) VRSS Framework mentioned in Section 4.2.3.1, which are the core functions of the prototype VRSS DSS. These components are independent from each other, allowing a particular component to be flexibly modified or replaced. To establish independent relationships between them, the components have to be connected to each other through interfaces. For example, the `VrssFramework` component is connected to the `VrssKnowledge` component through the interface named `IVrssKnowledge` as shown in Figure 4-27.

It is important to note that by breaking a large software system into smaller, manageable chunks it is quite possible to identify the components that may be reused as part of other applications in the same domain. For example, the `CbaTabular` component, which provides the CBA Tabular decision-making method, has a considerable potential to be reused in applications that require this comparative analysis capability. Therefore, component-level design not only promotes the independency between the components, but also the reusability of the components that work together to fulfill the functional requirements for the demonstration website and the prototype VRSS DSS developed during Prototyping Cycle 1.

In summary, this section discusses the different aspects of demonstration website and the prototype VRSS DSS design from the user-observable to the technological. Based on the requirement models provided in Section 4.2.3.1, a series of design models were developed to support the different design tasks presented in this section. The models were used as a guideline for constructing the website and the prototype DSS, which will be further elaborated in the next section.

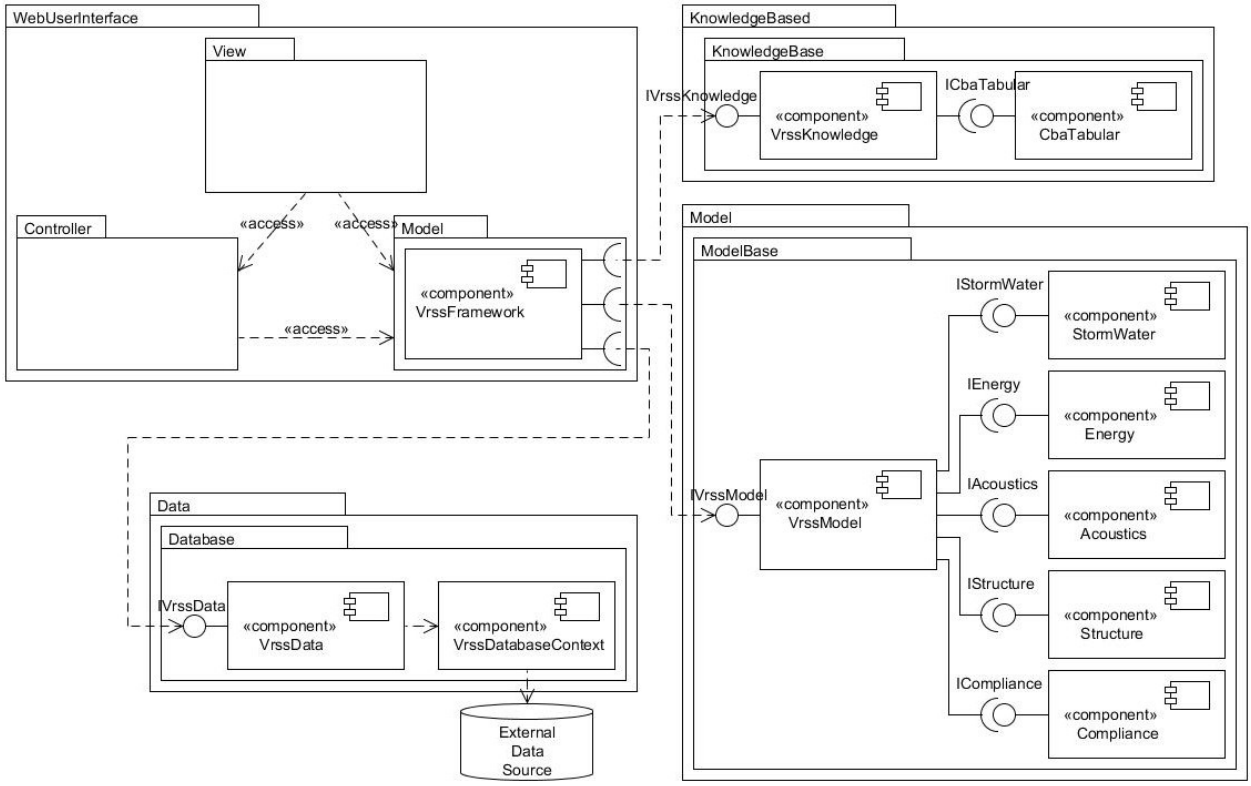


Figure 4-27: Detailed component-level design

4.2.4. Prototyping Cycle 1: Prototype Construction

After the requirement and design models were developed, the next step of the prototyping process is to construct the prototype software. The construction phase of the prototyping process involves writing the programming language source code (coding) and uncovering errors in the written code (testing). For constructing the demonstration website and the prototype VRSS DSS, the author adopted an integrated coding and testing paradigm known as Test-Driven-Development (TDD). TDD was used as a basis for building the major components of the data management, model management, knowledge-based management, and user interface subsystems previously presented in Section 4.2.3.2. This section discusses the TDD and the construction of the aforementioned major components in detail.

Test-Driven Development

Test-Driven Development (TDD) is an integrated coding and testing approach that allows programmers to define software behaviors upfront and then develop the source code that defines the behaviors. In general, the TDD can be conducted using IDEs that are equipped with unit testing tools. However, Microsoft Visual Web Developer 2010 Express, the IDE used for developing the demonstration website and the prototype VRSS DSS, did not come with a built-in testing tool. Therefore, the author decided to use NUnit, which is a widely used testing tool provided by www.nunit.org, for testing. The author also adopted a procedure suggested by experienced programmers such as Allen (2011), known as Red-Green-Refactor (RGR), to enhance the construction of the website and the DSS discussed in this section.

Fundamentally, the TDD begins with developing a series of test fixtures. Such test fixtures can be developed using C# (pronounced C sharp), which is the primary language for developing the analysis and POCO classes of an ASP.NET MVC application. A test fixture is actually a C# class that has a set of tests or operations similar to the one shown in Figure 4-28. Figure 4-28 presents the test fixture named `CbaTabularTest` developed for testing the `CbaTabular` class, which can be seen as the main class of the `CbaTabular` component discussed in Section 4.2.3.2. In Figure 4-28, `CbaTabularTest` connects to `CbaTabular` through the `ICbaTabular` interface.

The test fixture presented in Figure 4-28 includes a number of tests that represent the operations that should be provided by `CbaTabular`. To create a particular test in the test fixture, the author implemented the Arrange/Act/Assert (A/A/A) pattern recommended by Freeman and Sanderson (2011). This pattern involves setting up test conditions (Arrange), performing the test (Act), and verifying the test result (Assert). For example, the simplified pseudo code of the `TestRemoveFactor` test developed based on the A/A/A pattern is presented in Box 4-1. As shown in Box 4-1, three decision-making factors are initially added to the CBA analysis to set up test conditions. To perform the test, `factor3` is removed from the CBA analysis. After `factor3` is removed, it is essential to verify that there are only two remaining factors included in the analysis. For the more complex test fixtures, the mocking tools such as Moq, found at http://code.google.com/p/moq, may be used for arranging the test conditions of a particular test instead of the standard way shown in Box 4-1.

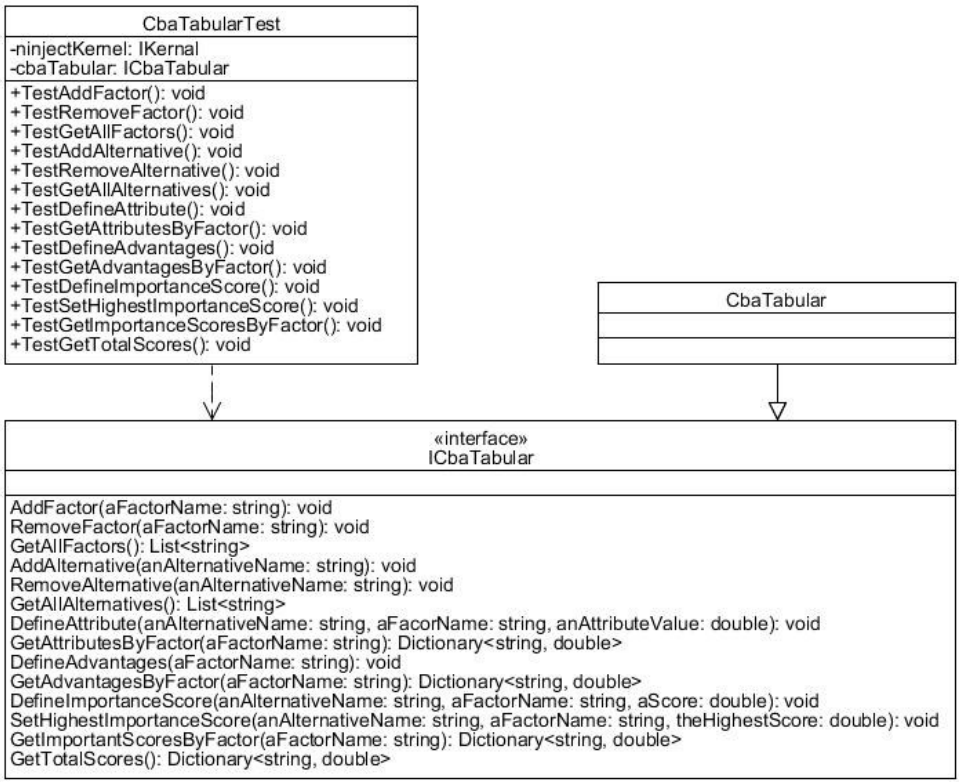


Figure 4-28: Class diagram of a test fixture

Box 4-1: Pseudo code developed based on the A/A/A pattern

```
//Arrange
Add a factor named "factor1"
Add a factor named "factor2"
Add a factor named "factor3"

//Act
Remove the factor named "factor3"

//Assert
Is the number of factors equal to 2
```

After a test fixture is developed, the source code of classes can be written and tested against the developed test fixture. For instance, Figure 4-29 demonstrates the initial run of the test fixture presented in Figure 4-28 before the source code is written to fulfill the requirements for `CbaTabular`. As shown in Figure 4-29, the status bar on the NUnit screen is colored in red because the class tested against the test fixture does not exhibit the expected behaviors defined by the test fixture. This situation represents the "Red" in the RGR procedure.

From the point discussed above, the source code needs to be written to ensure that each operation of `CbaTabular` displays the expected behaviors defined by a particular test in `CbaTabularTest`. Figure 4-30 presents the testing results after the code for the `RemoveFactor` operation of `CbaTabular` is written. This time the status bar is colored in green because `RemoveFactor` shows the expected behaviors defined by `TestRemoveFactor`. This situation represents the "Green" in the RGR procedure.

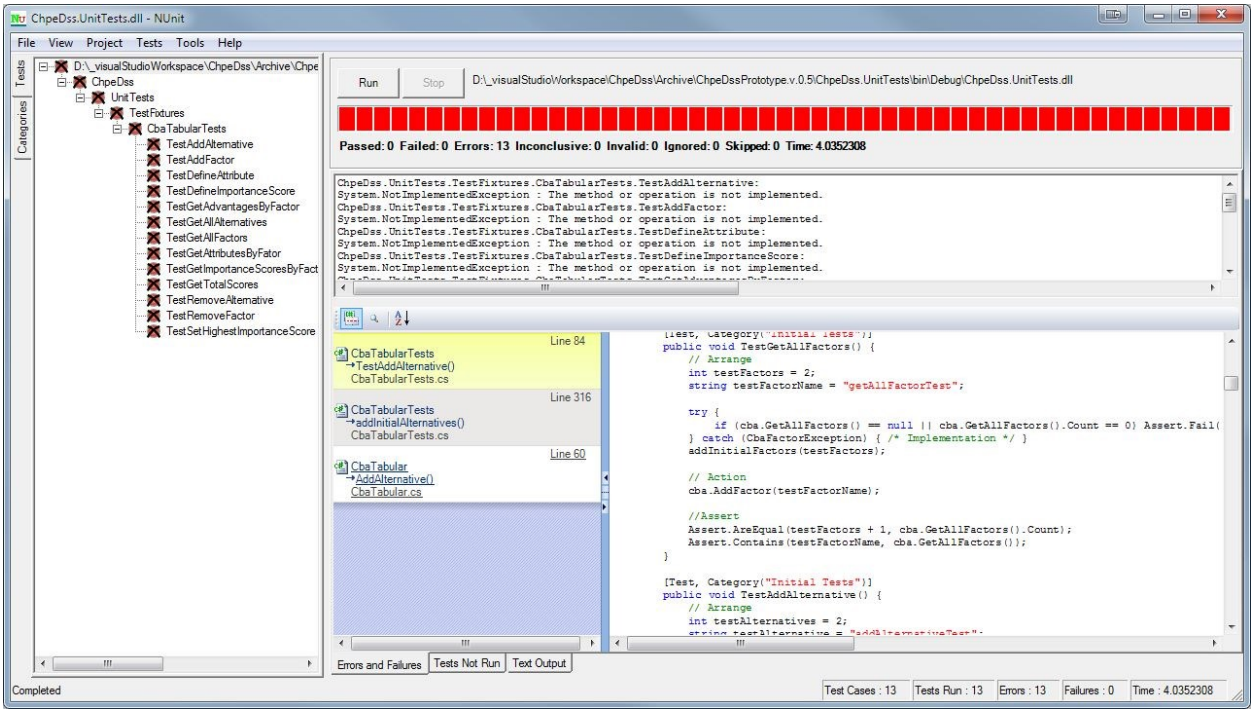


Figure 4-29: Initial run of the test fixture

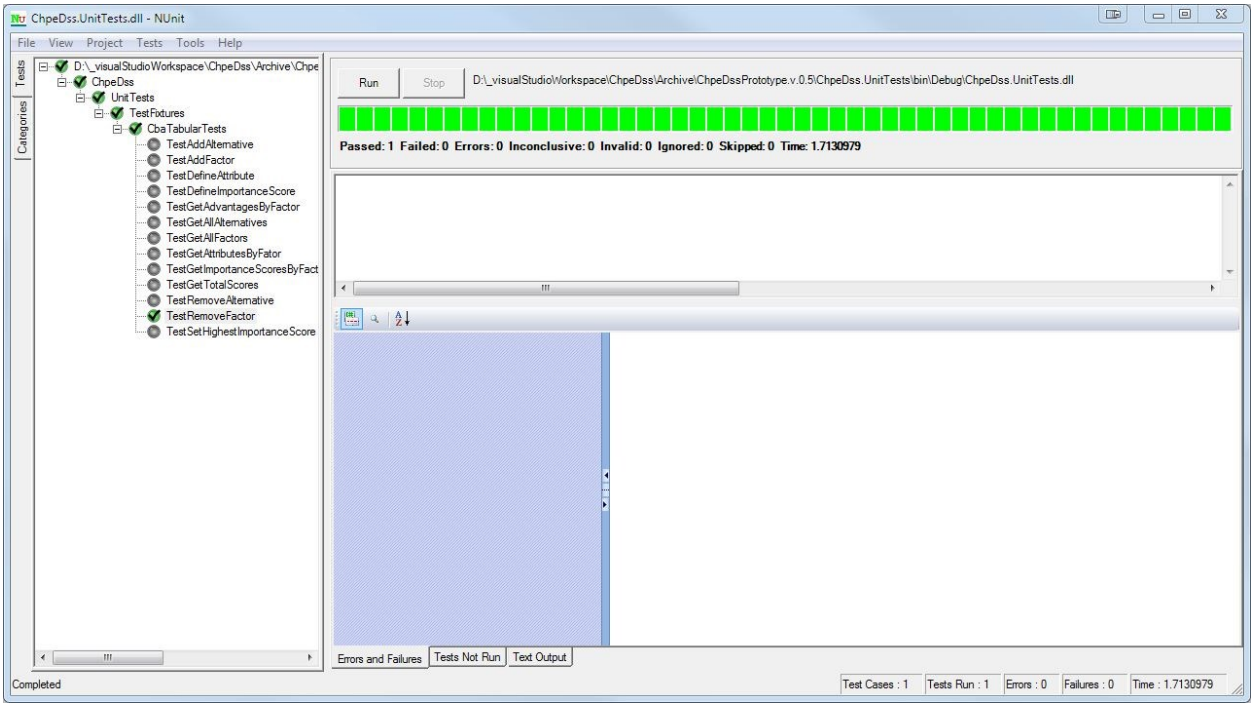


Figure 4-30: Test results after the code for the `TestRemoveFactor` operation is developed

Once `CbaTabular` is fully developed to satisfy all of its requirements, it should pass all the tests when tested against `CbsTabularTest` as shown in Figure 4-31. From now on, the RGR procedure can move towards the "Refactor", which focuses on improving the quality of the source code. For example, as mentioned in Chapter 3.2.5, the visual layout of the code may be improved to help make the code easier to read and comprehend.

Based on the information above, the TDD in general can be seen as a practical guideline for coding and testing activities. Because the ASP.NET MCV framework highly embraces the TDD, the TDD was used as a basis for developing the source code for the demonstration website and the prototype VRSS DSS. For Prototyping Cycle 1, the TDD was adopted for constructing the major components of the data management, model management, knowledge-based management, and user interface subsystems, which will be discussed further in this section.

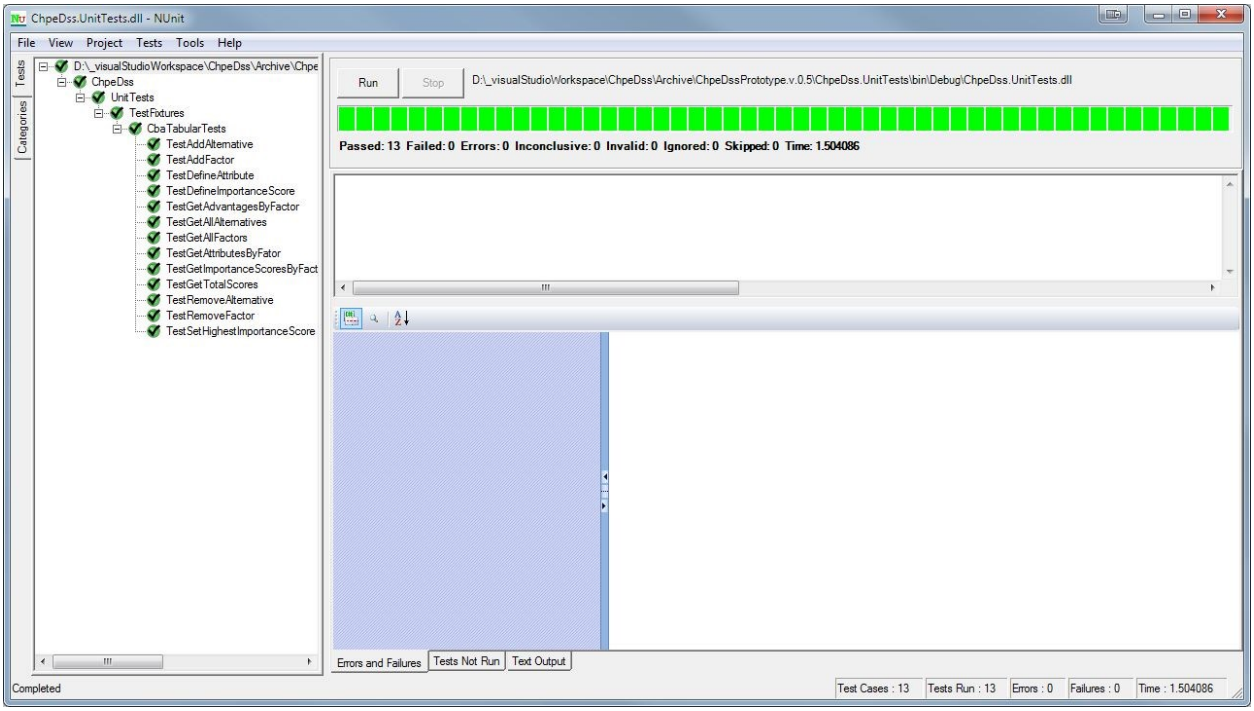


Figure 4-31: Final test results

Knowledge-Based Subsystem

To effectively build the prototype DSS application, it is important to breakdown the application building work to smaller, more manageable tasks. For the construction of the knowledge-based management system, the essential tasks involved creating the software components that work together to provide knowledge for enhancing the decision-making processes and storing the created components in the knowledge base. The components, including interfaces and classes, that were created during Prototyping Cycle 1 are shown in Figure 4-32.

Figure 4-32 demonstrates the knowledge components of the prototype VRSS DSS extended from the component design model presented in Section 4.2.3.2. As mentioned earlier, Suhr's (1999) CBA Tabular Method can be seen as the core of Grant's (2007) VRSS Framework and has a considerable potential to be developed as a reusable component. Therefore, the interfaces and classes that provide the CBA Tabular Method functions were independently built and stored in the `ChoosingByAdvantages` package as shown in Figure 4-32. Because the CBA decision-making system comprises a variety of methods for different decision-making tasks, `ChoosingByAdvantages` may contain additional CBA methods developed in the future. Within `ChoosingByAdvantages`, the `CbaTabular` class was built to provide the major functions of the CBA Tabular Method originally defined by Suhr. This makes `CbaTabular` reusable by other DSS and extensible by other classes to perform additional tasks if necessary.

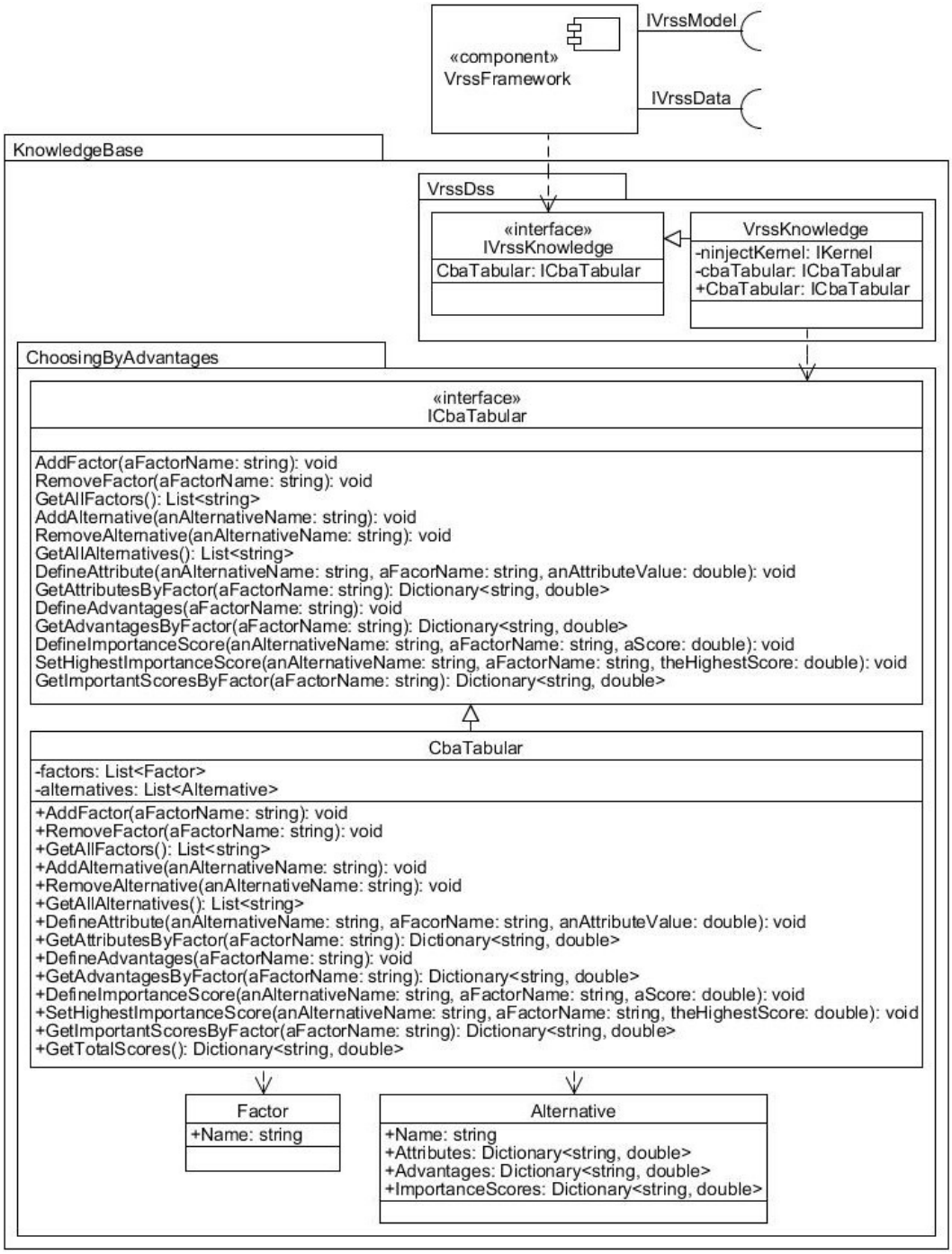


Figure 4-32: As built class diagram of knowledge components

As shown in Figure 4-32, the classes that utilize `CbaTabular` such as `VrssKnowledge` can access `CbaTabular` through the interface named `ICbaTabular`. This allows `CbaTabular` and the classes that utilize it to be highly independent from each other. To effectively create a loosely-coupled relationship between two classes through an interface, the author has adopted a technique called Dependency Injection (DI), also known as Inversion of Control (IoC), recommended by Freeman and Sanderson

(2011). This technique allows objects that implement a particular interface to be obtained without creating the implementing object directly. To implement DI, the author also used Ninject, which is an open source DI container provided by www.ninject.org, for initiating the implementation of an interface, passing it to the constructor of the assessor class, and then returning the result as the implementation of the interface.

To provide an example of the DI concept above, Box 4-2 demonstrates how the DI was implemented in `VrssKnowledge`. As shown in Figure 4-32, `VrssKnowledge` requires an object instance of `CbaTabular`, which implements `ICbaTabular`. This dependency can be injected to the constructor of `VrssKnowledge` shown in Box 4-2 using the Get<T>() operation. This dependency injection method is often referred to as "constructor injection". Furthermore, to support DI, all operations of `CbaTabular` were coded to accept only value types (e.g., integer, bool, and string) as their parameters. This reduces the needs to instantiate objects of concrete classes and to use them for exchanging data between `CbaTabular` and the classes that utilizes it such as `VrssKnowledge`.

Box 4-2: Partial code for dependency injection

```
// Obtaining an object instance of CbaTabular which implements ICbaTabular
public class VrssKnowledge : IVrssKnowledge {
     private IKernel ninjectKernel;
     private ICbaTabular cbaTabular;

     public VrssKnowledge() {
          ninjectKernel = new StandardKernel(new KnowledgeBaseModule());
          cbaTabular = ninjectKernel.Get<ICbaTabular>();
     }

     ...
}

// Binding ICbaTabular to CbaTabular that implements it
public class KnowledgeBaseModule: NinjectModule {
     public override void Load() {
          Bind<ICbaTabular>.To<CbaTabular>();
     }
}
```

After it was built and stored into the knowledge base, `VrssKnowledge` that utilizes `CbaTabular` became available through the knowledge-based subsystem. Other components such as `VrssFramework` can access `VrssKnowledge` through `IVrssKnowledge` as shown in Figure 4-32. It is important to note that DI can also be implemented to create a loosely-coupled relationship between `VrssFramework` and `VrssKnowledge` and in other similar situations whereby this kind of relationship is required. Once the CBA Tabular Method functions become available, the next step is to create analytical models for defining attribute values of decision-making alternatives corresponding to different decision-making factors. The construction of such models will be discussed next.

Model Management Subsystem

Based on the component design diagram presented in Section 4.2.3.2, the construction of the model management subsystem primarily involved creating a set of analysis models and storing them in the

model base. During Prototyping Cycle 1, the models were created based on the influence diagrams and decision tree algorithms developed by Grant for defining alternative attributes corresponding to a set of decision-making factors. The created models are discussed below.

In her study, Grant (2007) has identified eleven relevant decision-making factors involved in vegetated roofing system selection. These factors can be categorized into six categories: 1) storm water, 2) energy, 3) acoustics, 4) structure, 5) compliance, and 6) cost. Among the identified factors, there are five factors in which Grant was able to develop influence diagrams and decision tree algorithms for determining alternative attributes. These five factors are as follows: 1) storm water retention, 2) potential energy savings, 3) approximate sound transmission class, 4) surplus dead load of roof system, and 5) potential contribution to LEED certification. In this study, the models developed by Grant were used as a basis for creating the software components shown in Figure 4-33.

As presented in Figure 4-33, the analysis classes that were created based on Grant's influence diagrams and decision tree algorithms include `Retention`, `PotentialSavings`, `ApproximateStc`, `SurplusDeadLoad`, and `LeedContribution`. These analysis classes implement interfaces, which makes the analysis classes loosely coupled with the classes that utilize them. The interfaces created to specify the functions of five analysis classes shown in Figure 4-33 include `IRetention`, `IPotentialSavings`, `IApproximateStc`, `ISurplusDeadLoad`, and `ILeedContribution`. These interfaces implement `IAttributeIdentifier` that specifies the main function of the analysis classes, which is to identify the alternative attribute values for a particular decision-making factor summarized in Appendix A. For example, Box 4-3 demonstrates a code written as part of `SurplusDeadLoad`, which implements `ISurplusDeadLoad` that implements `IAttributeIdentifier`. The code in Box 4-3 was written to determine the alternative attributes for the surplus dead load of a roof system factor.

Also shown in Figure 4-33, the classes and interfaces discussed earlier were stored in separate directories whereby each directory represents a particular category of decision-making factors defined by Grant. This provides opportunities for new analysis classes to be added to directories in the future. For instance, the storm water category initially comprises three decision-making factors: 1) storm water retention, 2) storm water pollutant control, and 3) runoff warming. Between these three factors, storm water retention has been the only factor in which the influence diagram and decision tree algorithm could be developed. In the future, if the influence diagram and decision tree algorithm for identifying alternative attributes corresponding to the storm water toxicity factor can be defined, `Toxicity` can be a new class added to the storm water category.

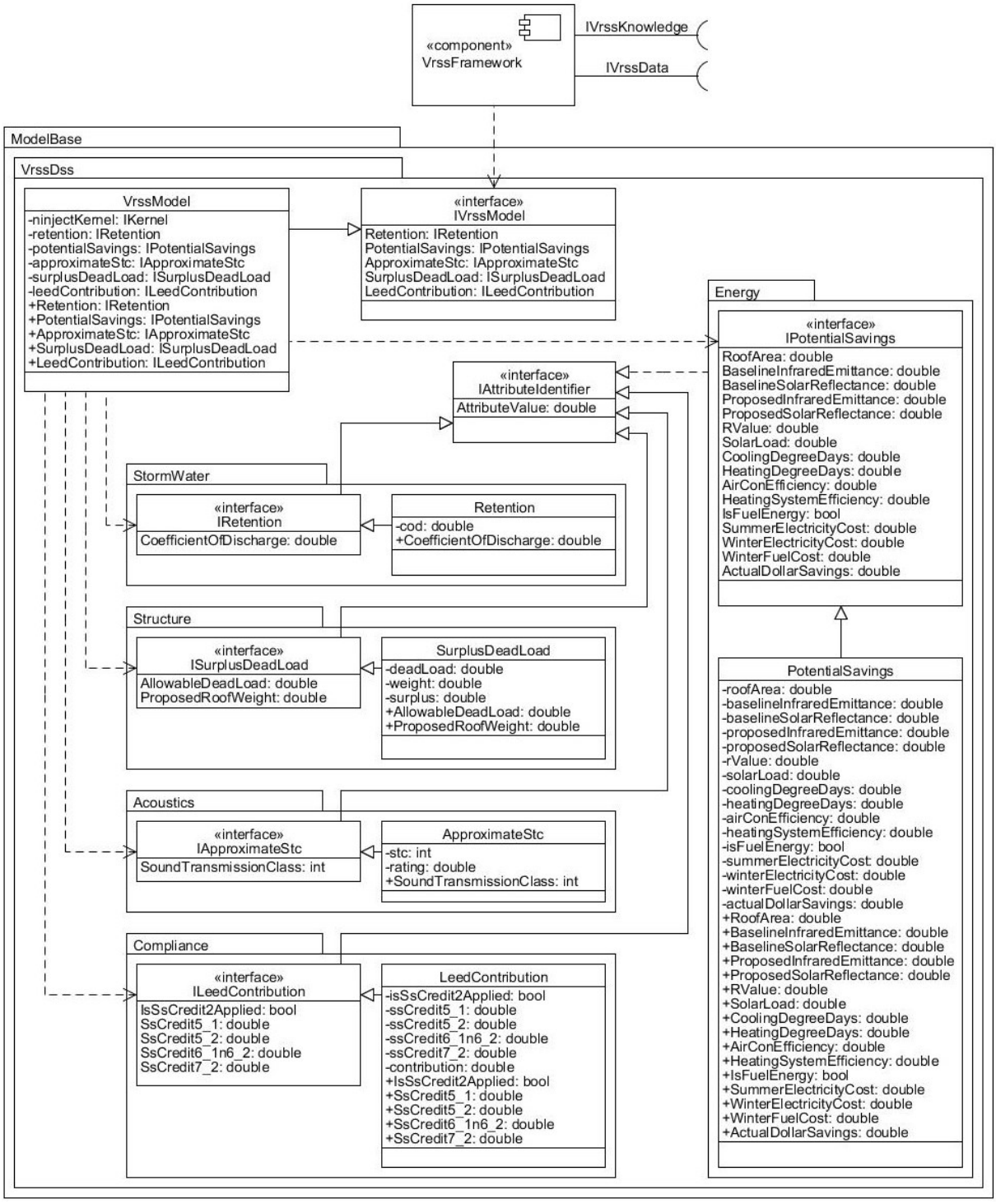


Figure 4-33: As built class diagram for model components

Box 4-3: Partial code for `SurplusDeadLoad`

```
public class SurplusDeadLoad : ISurplusDeadLoad {

    ...

    // Return the attribute value for the surplus
    // dead load of roof system factor
    public double AttributeValue {
     get {
          if (deadLoad >= weight) {
              surplus = 1.0;
          } else {
              surplus = 0.0;
          }
          return surplus;
       }
     }
   }
}
```

In addition to the information above, it is important to note that the current version of the VRSS Framework adopts DOE Cool Roof Calculator, a public domain JavaScript program developed by the U.S. Department of Energy's Oak Ridge National Laboratory, for calculating energy savings gained from proposed roofs. As a result, a large portion of `PotentialSavings` code was ported from JavaScript to C#, which was the primary language used for writing the software code in this prototyping project. Although `PotentialSavings` was created based on DOE Cool Roof Calculator and has a considerable potential to be reused by other DSS, the class was not built for reuse in this version of prototype DSS.

The analysis classes built during Prototyping Cycles 1 play an important role in defining the alternative attribute values corresponding to different decision-making factors. To identify such attribute values, these classes require a considerable amount of data from an external database as defined by the system architecture presented in Section 4.2.3.2. The construction of the database and the software components created for retrieving data will be discussed next.

Data Management Subsystem

According to the system architecture presented in Section 4.2.3.2, the data management subsystem is the system component responsible for transforming data stored in data sources into decision support data. To fulfill this function, the construction of a data management subsystem often encompasses creating data classes that describe database entities and their relationships as well as building a database for storing data in data storage hardware. Therefore, it is important to discuss how the data classes and database were built during Prototyping Cycle 1 in detail.

Figure 4-34 demonstrates the data components created during Prototyping Cycle 1. In Figure 4-34, `VrssData` is responsible for defining database entities (POCO classes) that provide data for the previously discussed analysis models. `VrssData` utilizes `VrssDatabaseContext` which is responsible for extracting data from `VrssDatabase` and creating memory-resident representations of data. `VrssData` can be accessed through `IVrssData` as shown in Figure 4-34.

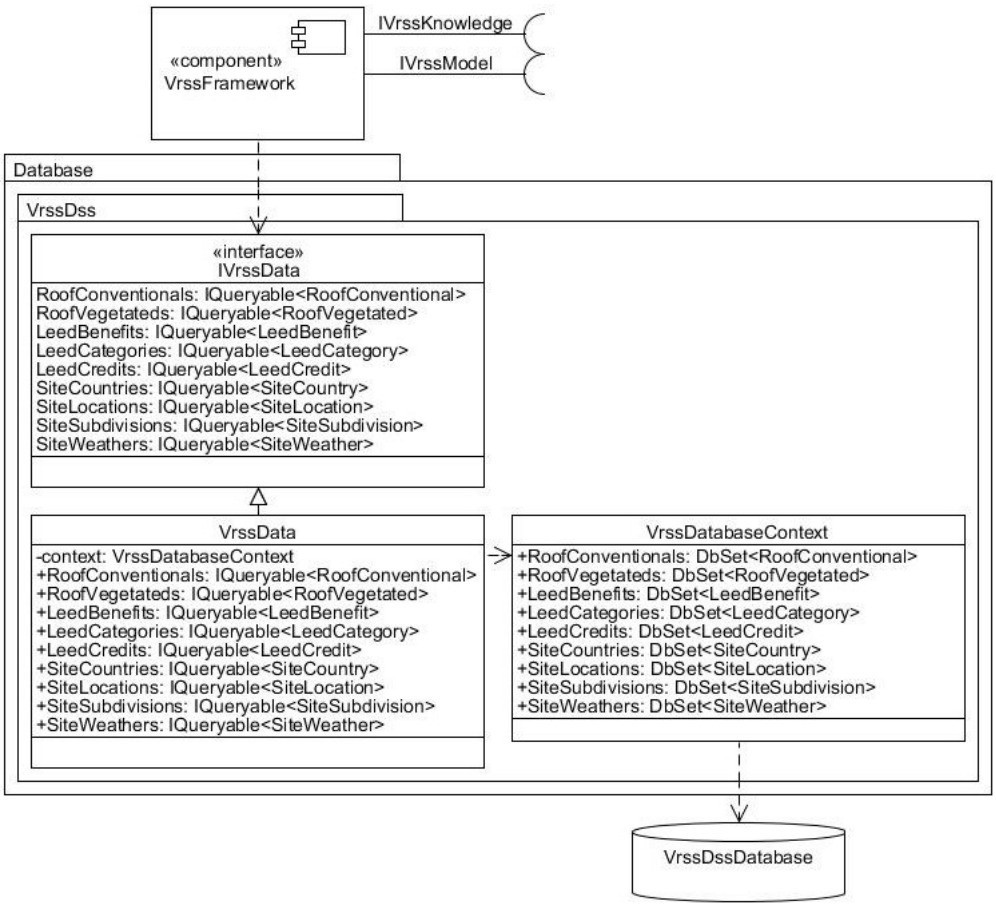


Figure 4-34: As built class diagram for data components

To effectively create the data components presented in Figure 4-34, the author adopted the Entity Framework which allows the author to use POCO classes to work with persistent data. Entity Framework is an Object-Relational Mapper (ORM) framework built upon Microsoft's .NET data access component known as ADO.NET, the combination of the ActiveX Data Objects (ADO) and Object Linking and Embedding for Database (OLE-DB) technologies. Entity Framework handles the low-level code that interfaces with an ADO.NET data provider used for communicating with SQL Server, the Database Management System (DBMS) implemented in this prototyping project. The DBMS is a piece of software used for creating, accessing, and updating the database. For DSS applications, the DBMS is also used for managing the data management subsystem as mentioned in Chapter 2.2.4.

Entity Framework allowed the author to write the code for POCO classes and map POCO classes to database tables extracted from an existing database. This workflow is called the “Code First” with an existing database approach. For example, Box 4-4 demonstrates the code written for `SiteCountry`, which is the data class that describes a database table that contains data required by `PotentialSavings`. Because Entity Framework adopts the convention over configuration paradigm similar to ASP.NET MVC, the property name that represents the primary key of a database table must

end with "ID" or "Id". For example, ID is the property that represents the primary key of the `SiteCountry` as shown in Box 4-4. In addition, `SiteSubdivisions` in Box 4-4 demonstrates that the primary key of `SiteCountry` is included in another table, `SiteSubdivisions`, as a foreign key. The relationship between `SiteCountry` and `SubDivisions` is presented in Figure 4-35, which represents all database tables that provide data for `PotentialSavings` together with their relationships.

Box 4-4: Partial code for `SiteCountry`

```
public class SiteCountry {

     // Properties (Columns)
     public int ID { get; set; }
     public string Name { get; set; }
     public string Alpha_2 { get; set; }
     public string Alpha_3 { get; set; }
     public string UnNumberic_3 { get; set; }
     public string Fips10_4 { get; set; }

     // Tables that have SiteCountryID as a foreign key
     public virtual ICollection<SiteSubdivision> SiteSubdivisions { get; set }
}
```

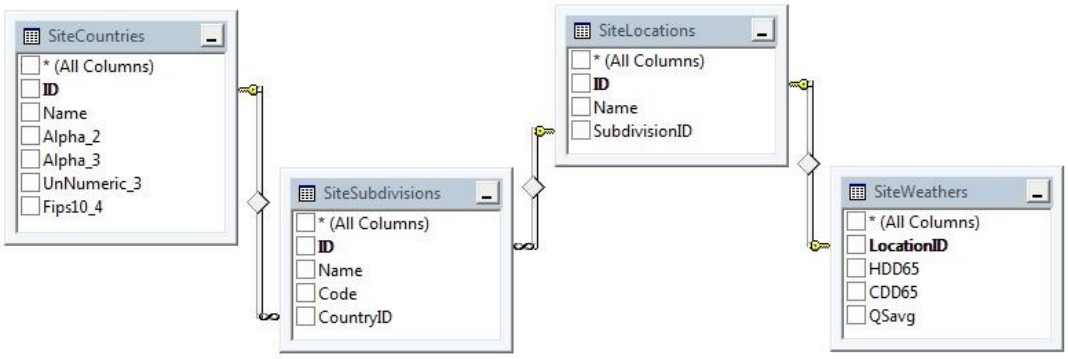


Figure 4-35: Relational diagram for weather data

In addition to the relational diagram presented in Figure 4-35, Code First was used to create data classes that represent the data of roof systems shown in Figure 4-36. The relational diagram in Figure 4-36 shows that `RoofConventionals` is the table that contains the data about conventional roofs. This table was created to support the future extension in which the DSS may provide a list of conventional roofs for DSS users to compare with green roofs. As shown in Figure 4-36, `RoofVegetateds` contains the data specific for green roofs, which actually are conventional roofs with more layers and properties. It is important to note that the database created during Prototyping Cycle 1 was normalized to meet the requirements of the third normal form (3NF); meaning that the database must contain no transitive dependencies. As a result, data tables that contain the data about LEED certification were separated from the tables that contain data about roof systems.

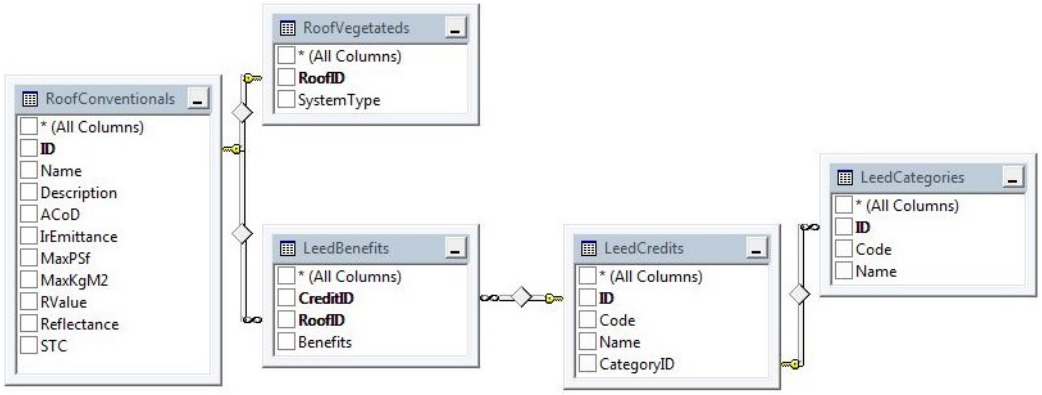


Figure 4-36: Relational diagram for roof systems

During Prototyping Cycle 1, the data classes and database were created to provide decision support data for the analysis models to define the alternative attribute values for a series of decision-making factors, which are the important inputs for the CBA Tabular Method. To create data components discussed above, the author has implemented the Entity Framework that played an important role for mapping data classes to an existing relational database managed by the DBMS (SQL Server). Because the created data classes are in fact the POCO classes, they can be easily used in conjunction with other classes built to fulfill the functions of the prototype application as a whole.

User Interface (Dialog) Management Subsystem

Once the back-end components including data, model, knowledge-based management subsystems have been built, the next step is to construct the user interface (dialog) subsystem. Based on the system architecture presented in Section 4.2.3.2, the user interface subsystem represents a whole ASP.NET MVC application, which is composed of three major components: model, view, and controller. Therefore, the main purpose of the user interface subsystem construction performed during Prototyping Cycle 1 was to build an ASP.NET MVC application that works as a web-based user interface or WebUI of the prototype DSS application.

Basically, when creating an ASP.NET MVC project in Microsoft Visual Web Developer 2010 Express, the project comes with prebuilt models, views, and controllers. Some of the prebuilt components are presented in Figure 4-37. These prebuilt components were considered adequate for creating the demonstration website with the login/out and registration functions corresponding to the requirements discussed in previous sections. Based on the components shown in Figure 4-37, the `Home` page of the demonstration website is presented in Figure 4-38.

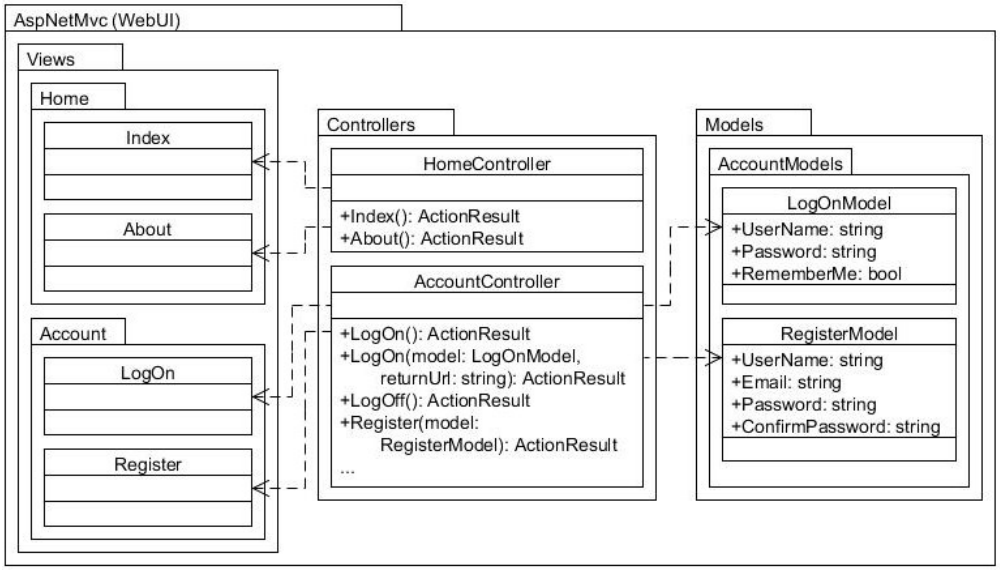


Figure 4-37: Prebuilt components

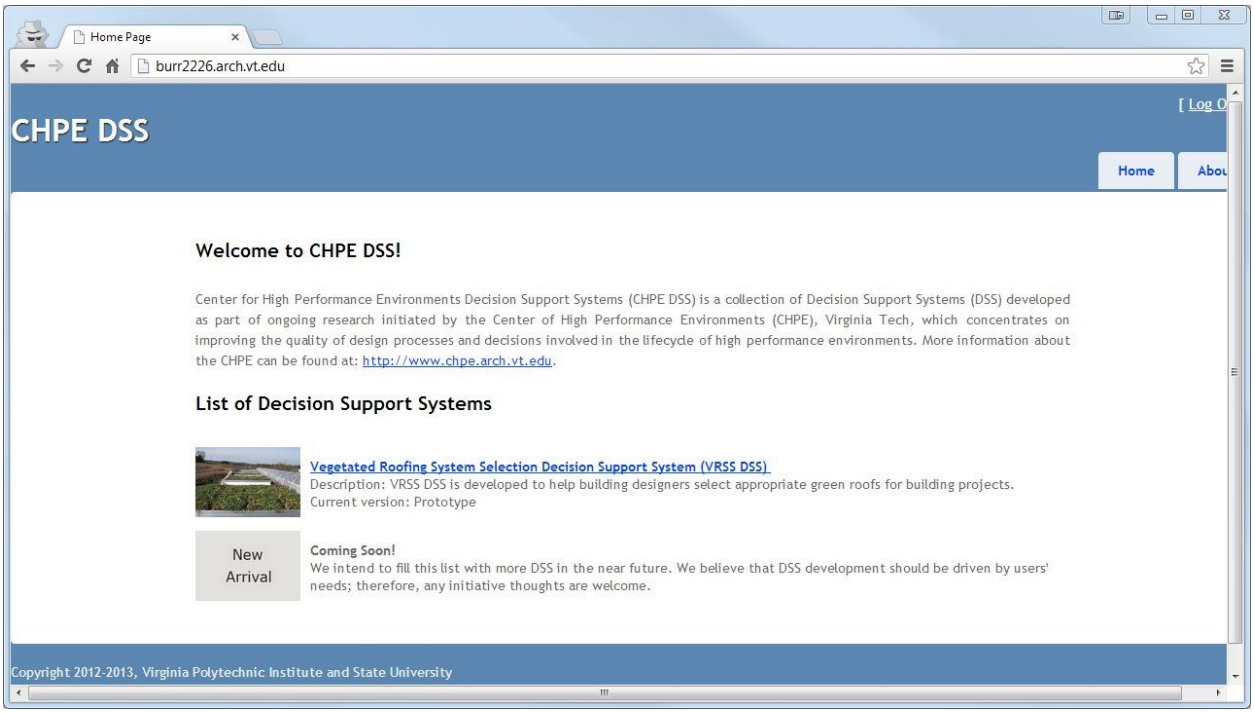


Figure 4-38: `Home` page

The prebuilt components discussed above not only can be used for creating the demonstration website, but also provides a pattern or workflow for building the WebUI of the prototype VRSS DSS and future DSS. Based on Figure 4-37, the construction of a new WebUI may begin with creating models (e.g., `LogOnModel` and `RegisterModel`) that provide business logics and interact with other DSS subsystems. After models are created, the WebUI construction can focus on building a controller (or controllers) that uses these models to generate content; then, the controller chooses a view to present the

generated content. Based on this workflow, it is highly possible to add a significant number of WebUIs into a single environment. Alternatively, for larger systems, WebUIs in the form of ASP.NET MVC applications may be developed separately, and then deployed on different servers. For this scheme, the `Home` page shown in Figure 4-38 can be simply used as an access point to multiple DSS.

Figure 4-39 below demonstrates how the concept above was implemented to create the WebUI of the prototype VRSS DSS. In Figure 4-39, `VrssFramework` is the model class that interacts with other subsystems of the DSS to provide DSS functions. In addition to `VrssFramework`, `VrssInput` and `VrssReport` serve as view models that take care of data content to be presented to users or decision makers. `VrssDssController` is the controller class responsible for processing user's requests, interacting with `VrssFramework`, and choosing views and return them to users. The views that `VrssDssController` can choose from includes `Main`, `Input`, and `Report` as shown in Figure 4-39.

As mentioned earlier in Section 4.2.2.2, ASP.NET MVC adopts the convention over configuration paradigm. For example, the name of controller classes such as `VrssDssController` must end with `Controller` to make controller classes recognizable by the routing system. As well, the name of controller methods should match the names of the views. For instance, the method named `Background()` should match with the `Background` view as presented in Figure 4-39.

To provide an example, Box 4-5 shows the partial code written for model, view, and controller classes shown in Figure 4-39. `VrssInput` presented in Box 4-5 was created in the form of the POCO class that has a set of class properties. Such properties contain input data for `VrssFramework`. In Box 4-5, `Input` was created as a view that provides multiple partial views that provide input elements defined by `VrssInput`. For instance, Box 4-5 shows the code for the partial view named `StormWater` written using Razor syntax, a technology implemented by ASP.NET MVC for integrating C# code into HTML code. When a user send a request for the `StormWater` view to `VrssDssController`, the `Input` method of `VrssDssController` is responsible for returning the web page containing an input form shown in Figure 4-40 to the user.

After the user fills in and submits the form, the `StormWater` method of `VrssDssController` is then responsible for assigning the inputs (method parameters) to the object of `VrssInput` and returning the next partial view, in this case the `Energy` view, to the browser. The browser then displays the `Energy` view to the user as presented in Figure 4-41. In contrast, if the user submits the form with invalid inputs, `VressInput` is responsible for identifying such invalid inputs and sending error messages back to the controller. The `StormWater` method then returns the web page containing error notifications presented in Figure 4-42

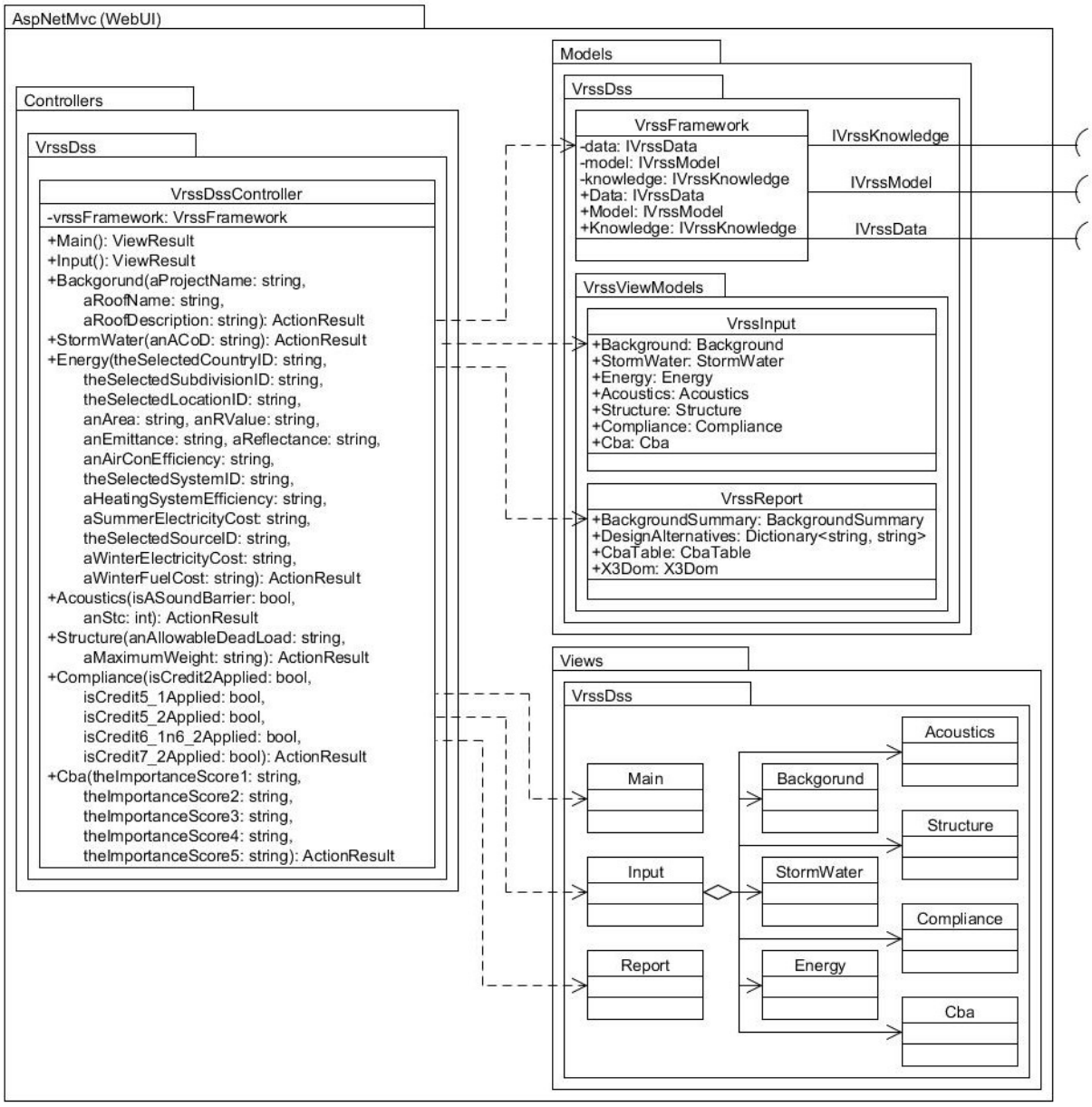


Figure 4-39: Class diagram for user interface subsystem

It is important to note that ASP.NET MVC fundamentally provides multiple ways to extract input data from HTML requests. However, the author decided to extract input data via controller method parameters. This approach helps simplify the testing procedure discussed earlier in this section by allowing the author to directly create a series of tests for different controller methods.

Box 4-5: Partial code for the user interface

```
// VrssInput (model)
public class VrssInput {
   ...
   public StormWater StormWater { get; set }
   ...
}
public class StormWater {
   public Retention Retention { get; set }
}
public class Retention {
   public double? ACoD { get; set }
}

// Input (view)
<div>
   @{ var partialView = ViewBag.PartialView as string;
      ...
      switch (partialView) {
         ...
         case "StormWater"
            <div>@Html.Partial("PartialViews/StormWater")</div>
         ...
      }
   }
</div>

// StormWater (partial view)
...
<tr>
   <td>
      @Html.LabelFor(m => m.StormWater.Retention.ACoD,
      "Annual coefficient of discharge, Ψa (0.0≤ Ψa≤1.0)")
   </td>
   <td>
      @Html.EditorFor(m => m.StormWater.Retention.ACoD)
   </td>
   ...
</tr>
...

// VrssController (controller)
public class VrssController : Controller {
   ...
   public virtual ActionResult Input() {
      var partialView = TempData["PartialView"] as string;
      ...
      ViewBag.PartialView = partialView;
      return View("Input", vrssInput);
   }
   ...
   public virtual ActionResult StormWater(
      [Bind(Prefix = "StormWater.Retention.ACoD")] double? anACoD) {
      ...
      this.TempData["PartialView"] = "StormWater";
      return RedirectToAction("Input");
   }
   ...
}
```

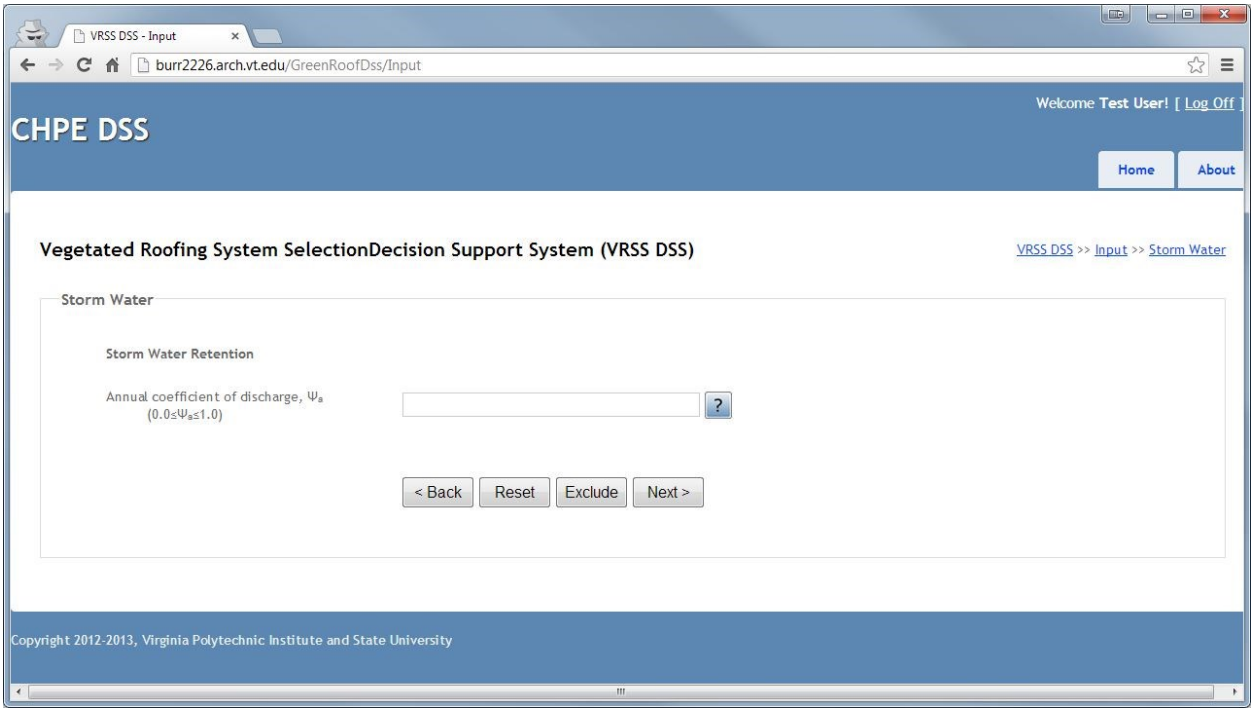


Figure 4-40: `StormWater` page

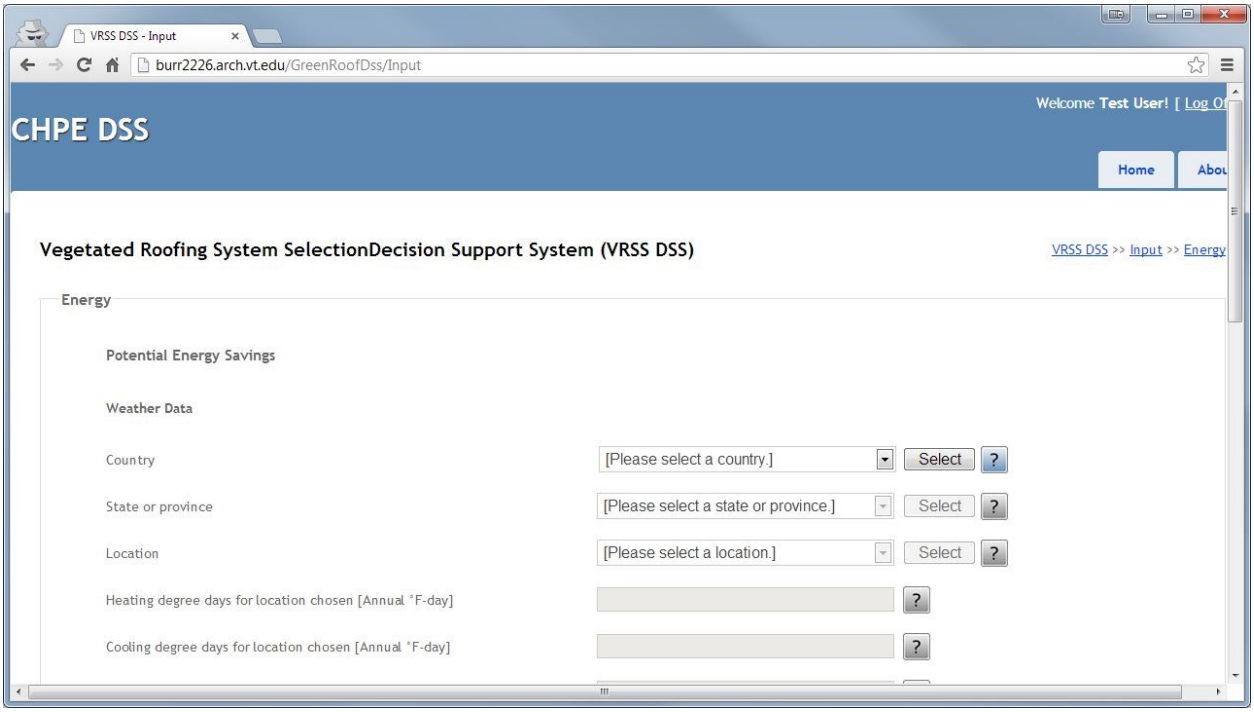


Figure 4-41: `Energy` page

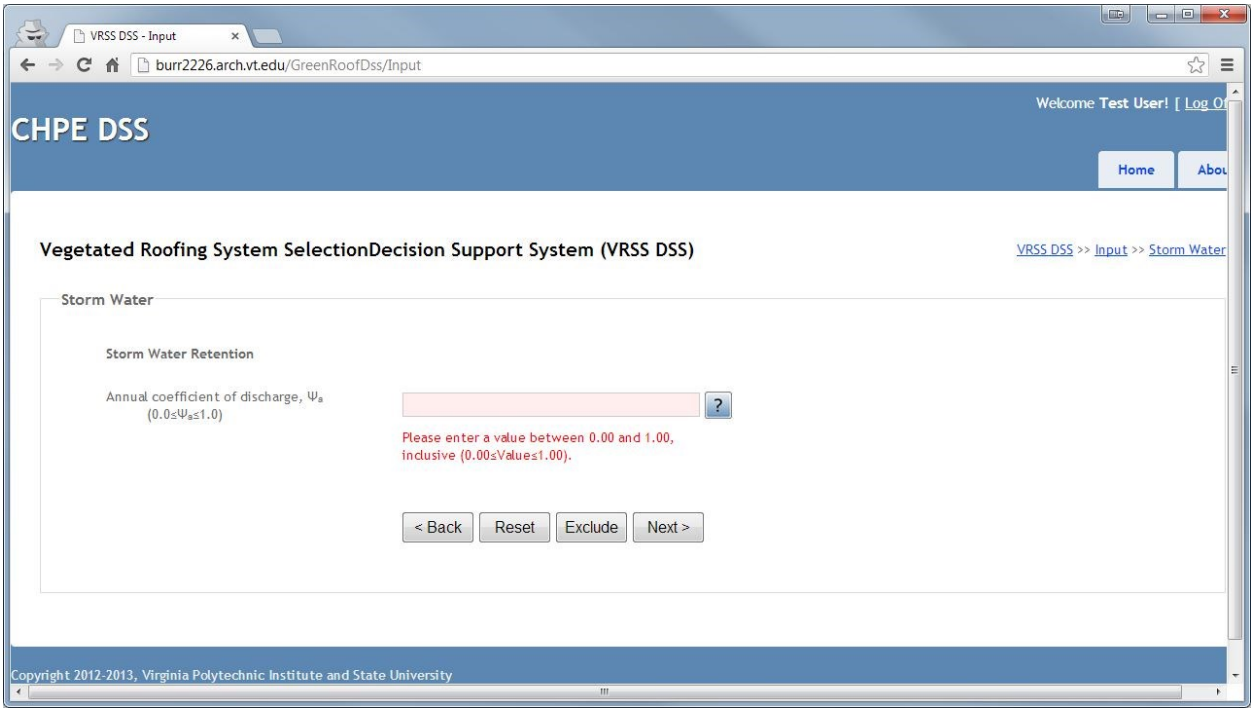


Figure 4-42: `StormWater` page with an error message

Based on the information above, the construction of the user interface (dialog) management primarily involved building an ASP.NET MVC application that represents the WebUI of the prototype DSS application. This ASP.NET application was created to serve as a WebUI for both the demonstration website and the prototype VRSS DSS. By implementing the Model-View-Controller pattern, the ASP.NET application built during Prototyping Cycle 1 is highly feasible to be extended to support other additional DSS that will be developed in the future.

In conclusion, the construction phase of the prototyping process encompasses building software components for the knowledge-based management, model management, data management, and user interface subsystems. To effectively create such software components, the author has adopted the Test-Driven Development framework for software coding and testing. The components discussed in this section were mostly created to be loosely-coupled with each other to make them flexible to change. Once all the components were built, the prototype DSS application was then delivered to the research participants to receive feedback on the features and functions of the system, which will be extensively discussed in the next section.

4.2.5. Prototyping Cycle 1: Delivery, Deployment, and Feedback

After a software application is constructed, it is essential for the application to be evaluated by the customer and end users. Customer and end users' evaluations typically help address the changes that have business values and bring awareness to issues that should be taken into consideration in the next prototyping cycle. As a result, the deployment phase of a prototyping process often involves delivering the

application to, providing support for, and collecting feedback from the customer and end users. This section discusses the three deployment tasks performed during the deployment phase of Prototyping Cycle 1 in detail.

As mentioned in Section 4.2.1, the customer and end users included the CHPE faculty researchers and research participants who participated during the first phase of this study. From the nine AEC practitioners who voluntarily participated in the prototyping project as end users, there were three participants who participated in this phase: two architects and one BIM coordinator.

To deliver the prototype DSS and collect feedback, the author arranged a project meeting with the CHPE faculty researchers and separate face-to-face meetings with research participants, both in person and via video conferences. The author deployed the prototype DSS using IIS Express installed on the author's laptop computer and gave a brief demonstration of the prototype DSS as part of the meetings. During the meetings, the CHPE researchers and research participants were free to have a hands-on experience of the prototype DSS and to give feedback on the features and functions of the system. As well, the CHPE researchers and research participants were informed that the prototype DSS had limited features and functions and were therefore provided with the author's technical assistance.

Based on the information gathered from the meetings, the CHPE researchers and research participants were satisfied with the software functions and decision support information provided by the prototype DSS. From their familiarity with the VRSS Framework and CBA process, the CHPE researchers verified that the functions and decision support information provided by the prototype DSS were accurate enough to be carried on to the next prototyping cycle. In addition to its accuracy, the research participants confirmed that the prototype DSS provides potential benefits to their business. According to the research participants' perspective, the prototype DSS offers a structured framework for identifying the benefits of vegetated roofing systems and for selecting an appropriate system based on its advantages. The prototype DSS also provides the research participants with useful decision support information for vegetated roofing system selection. Furthermore, the prototype DSS allows the research participants to communicate their decisions on vegetated roofing system selection and to show the reasoning behind their decisions to their project team. These benefits indicate the potential for the prototype DSS to be developed further in next prototyping cycle.

While the CHPE researchers and research participants were satisfied with the prototype DSS, there were certain issues that should be taken into consideration before the prototyping project moves toward the next prototyping cycle. The issues raised by the CHPE researchers and research participants are as follows:

- The spelling errors found on several web pages generated by the prototype DSS application should be examined.
- The terms used for describing the inputs should be less technical as much as possible.

- The mouse-over help messages for the inputs should be easy to understand and provide some common values for the users to compare against.
- The summary of the vegetated roofing system selection should provide a few alternative systems with their total importance score of advantages for an instant comparison.
- The system should be updated in a backward-compatible fashion to ensure that the decision support information generated by the updated system is consistent with the information generated before the system is updated.

Although the issues discovered during the deployment phase listed above provides valuable information for system improvement, the author decided to tackle only the issues that fall within the scope of prototype development presented in Section 4.2.2.1. Therefore, the backward compatibility was not taken into consideration in the prototype DSS development discussed in this chapter. Nevertheless, this issue is very important and will be revisited when designing and developing the full version system in the future.

The deployment phase of Prototyping Cycle 1 provides opportunities for the prototype DSS application to be evaluated by the CHPE researcher and research participants. The evaluations indicated that the prototype DSS not only fulfills the operational requirements, but also provides accurate decision support information and potential business values. The evaluations also reveal some improvement issues that should be taken into consideration as part of the next prototyping cycle. The information gathered during the deployment phase helps ensure that the prototype development can move toward Prototyping Cycle 2, which will be extensively discussed in the next section.

4.3. Prototyping Cycle 2

4.3.1. Prototyping Cycle 2: Communication

Continuing from Prototyping Cycle 1, Prototyping Cycle 2 aims to fulfill the second high-level requirement for prototype DSS development initially discussed in Section 4.2.1 whereby the DSS should be compatible with major BIM authoring tools. To understand how the DSS should be developed to meet this requirement, the author was able to collect useful information from the communications between the author and the research participants established in early 2012 and during the deployment phase of Prototyping Cycle 1. The collected information is presented in this section.

As mentioned in Section 4.2.1, there were nine research participants involved in the first phase of this study, concentrating on the prototype DSS development. These participants represent the end users of this prototyping project. In early 2012, the author first had opportunities to meet six participants including four roofing consultants, one roofing researcher, and one roofing manufacturer representative to gather information about how the DSS might be operated. This group of participants was provided with a brief presentation on this study and the second throwaway prototype as previously discussed in Section 4.2.1. In late 2012, the author then had opportunities to meet the remaining three research participants including

two architects and one BIM coordinator to deliver the prototype DSS developed during Prototyping Cycle 1 and collect the feedback on its features and functions.

Based on the information gathered from the communications between the author and the research participants, all research participants were familiar with the concept of building information modeling. From the nine participants, there were five participants who have used at least one BIM-based tool for their projects and four participants who occasionally worked with colleagues and/or clients who have used at least one BIM-based tool for their projects. According to the research participants, Autodesk Revit has been used in their projects as a major BIM authoring tool. When it comes to the exchanging of data between BIM-based tools, most of the participants often use Revit (.rvt) and Green Building XML (.gbxml) file formats for data exchange, but not regularly use Industry Foundation Classes (i.e. .ifc and .ifcxml) for this purpose. However, the research participants were interested to see how Industry Foundation Classes (IFC) can be implemented as part of their BIM processes after the author addressed the potential benefits of using standard formats previously discussed in Chapter 2.1.2 and 2.2.3.

In addition to the information presented previously, the research participants were able to provide valuable comments and feedback on the BIM compatibility feature and its possible functions as follows:

- The prototype VRSS DSS should provide a list of roof properties when the user uploads a BIM model to the system.
- The prototype VRSS DSS should provide a BIM model that represents the selected green roof system for the user to download and import to a working BIM project.
- The plug-ins for different BIM tools should be developed to facilitate the exchange of data between the prototype VRSS DSS and BIM tools.

To narrow down the scope of prototype development during Prototyping Cycle 2, the author decided to focus on the first two issues listed above and leave the last issue for future development. Although the availability of plug-ins would make the DSS more appealing to the users, plug-in development can be seen as another software project by its own right. Nevertheless, the issue concerning plug-ins development should be revisited in the future when the DSS becomes more mature.

The information collected from the communications between the author and the research participants helped define the new software functions that should be taken into consideration during Prototyping Cycle 2. The information given in this section was used as basis for prototype DSS development throughout Prototyping Cycle 2, which will be further discussed in the following sections.

4.3.2. Prototyping Cycle 2: Quick Planning

4.3.2.1. Prototyping Cycle 2: Scope

According to the information given in Section 4.3.1, the scope of Prototyping Cycle 2 was to make the prototype DSS compatible with BIM authoring tools employed by AEC practitioners. Within this defined

scope, the use case diagram modified from the use case diagram presented in Section 4.2.2.1 is presented below.

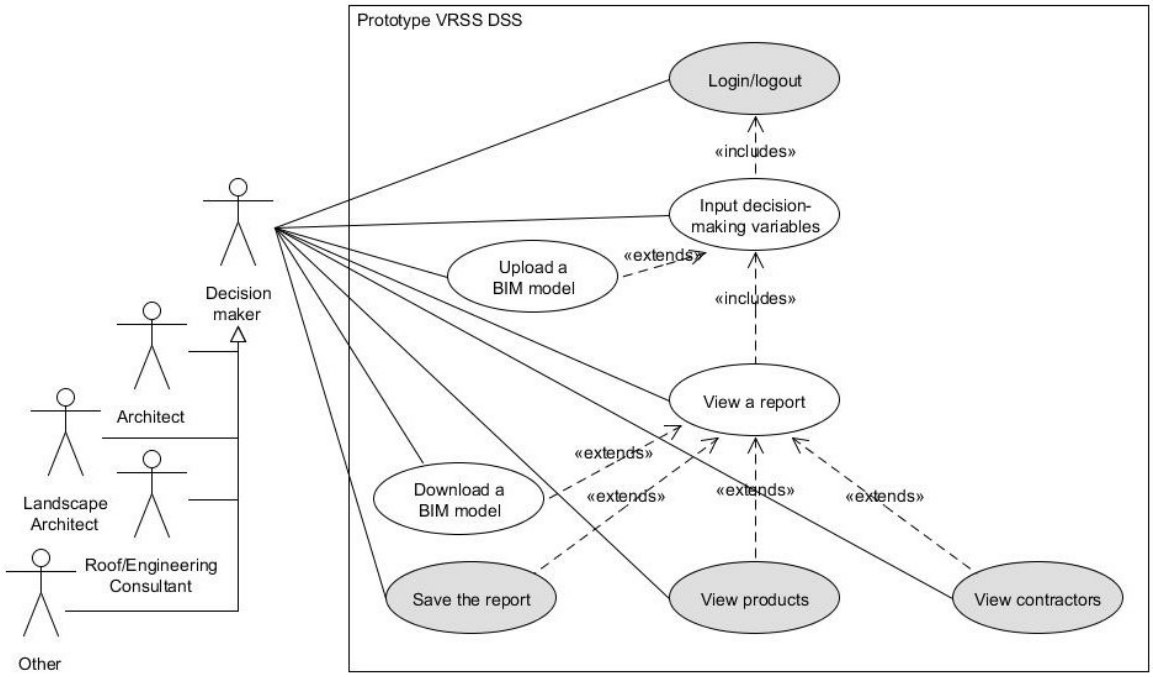


Figure 4-43: Scope of Prototyping Cycle 2

Prototyping Cycle 2 concentrated on the white colored use cases shown in Figure 4-43. Figure 4-43 demonstrates that a user can upload a BIM model to the web application server. After the model is uploaded, the prototype VRSS DSS is then responsible for extracting reference roof information from the model, automatically filling in the input forms for the users, and returning a web page that contains a list of reference roof properties to the user. The user can look through the listed properties to verify the information extracted from the model. The user may modify the model if necessary or modify the inputs directly through a series of input forms provided by the prototype DSS. After the user provides all the required inputs for the system, the system then generates the Report page and returns it to the user. Afterward, the user can download BIM models that represent different green roof systems provided on the Report page.

It is important to note that the use cases that represent the scope of Prototype Cycle 1 are also colored in white as shown in Figure 4-43, because a few software components developed during Prototyping Cycle 1 needed to be modified to provide the BIM upload and download functions. For example, additional web form controllers for uploading a BIM model and presenting a list of roof properties needed to be added to the Background partial view previously developed during Prototyping Cycle 1. These existing components should also be taken into consideration during the modeling and construction phases of Prototyping Cycle 2.

To make the prototype DSS compatible with BIM authoring tools, Prototyping Cycle 2 consists of developing software components that work to provide the BIM upload and download functions for end users. To add new components into the system, some of the existing components built during Prototyping Cycle 1 may need to be modified to accommodate the new components. The modifications needed to be made to the previously created components will be further discussed alongside the components developed during Prototyping Cycle 2.

4.3.2.2. Prototyping Cycle 2: Feasibility Study

For Prototyping Cycle 2, technology and resources were the dominant dimensions among the four dimensions of project feasibility: technology, finance, time, and resources. Technology played an important role in enabling the exchange of data between the prototype DSS and major BIM authoring tools. The resources, especially the BIM authoring tools used for investigating the compatibility issue, are quite expensive. To ensure that the prototype DSS is feasible to be developed for using in conjunction with BIM tools, it is essential to examine the technology and resources involved in the prototype development during Prototyping Cycle 2 in detail.

Technology

To implement DSS as a platform to transform “building information” to “useable knowledge” for design practitioners, it is essential to understand the interrelationship between BIM authoring tools and DSS. Based on several methods suggested by experts such as Eastman et al. (2008), Smith and Tardif (2009), and Onuma (2010), there are two methods that are relevant when exchanging building data between BIM authoring tools and DSS. The first method is to exchange data using physical data files. The second method is to upload data to BIM servers in which multiple BIM applications can use the centralized data to perform their tasks. Between these two methods, the author has adopted the physical data files method, as it is flexible and uncomplicated to implement. The significant benefit of this approach is that it allows the prototype DSS and BIM authoring tools to be loosely coupled. The independence between applications helps reduce risk and uncertainty caused by recurring changes and updates associated with each application. It is most reasonable at the prototyping level to take this approach, because changes can be considered the only known constant in the prototyping process.

Because the author has adopted the physical data files method for exchanging building data between the prototype DSS and BIM authoring tools, it is essential to choose appropriate data exchange formats and study how data can be exchanged using such formats. Of the many data exchange formats that are currently available and have been used in the building industry, the author has implemented the Industry Foundation Classes (IFC) as a basis for data exchange. IFC is currently the most widely-used public, non-proprietary, and well-developed product model in the AEC industries. At the current state of BIM technology, the IFC is more than likely to become the international standard for data exchange and integration within the building industries (Eastman et al. 2008).

The IFC is an open data exchange standard recently developed by the buildingSMART International (bSI) and registered as a standard by the International Organization of Standard (ISO) as ISO/PAS 16739. According to the bSI (2011), the IFC is in the process of becoming an official International Standard ISO/IS 16739. By the time of this study, there will reportedly be more than 150 software applications that support the IFC as listed on the bSI website at: http://www.buildingsmarttech.org/implementation/implementations. The current stable version of the IFC is the IFC2x Edition 3 or IFC2x3. The specification of IFC2x3 can be founded at: http://buildingsmart-tech.org/specifications/ifc-releases/ifc2x3-tc1-release/summary.

In addition to the IFC, the bSI also develops the ifcXML, the eXtensible Markup Language built up on the IFC. Although it is currently not a standard, the ifcXML seems to have a bright future, because XML is a semantic, portable language, which can be understood by both human and computers. In the Business Information System (BIS) field, XML is widely used to transfer data over the Internet and Web and is very natural to web-based DSS. However, in the AEC domain, there are only a few software applications that support ifcXML as compared to the applications that support the conventional IFC in all building related fields. In addition, the ifcXML files tend to be larger in size than the conventional IFC files. Therefore, the ifcXML is not implemented in this version of prototype DSS, but should be further examined in its future development.

According to the IFC2x3 Model Implementation Guide published by Liebish (2009), all implementations of the IFC are the implementations of IFC view, also later defined as Model View Definition (MVD). According to the bSI (2011), there are currently three IFC views that could be used to implement the IFC2x3: 1) coordination view, 2) basic FM handover view, and 3) structural analysis view.

For this prototyping project, the development of prototype DSS implements the IFC coordination view, which is normally used for exchanging building data between the architectural, structural, and mechanical BIM applications. This model view provides the definition for spatial structure, building, and building service elements with shape representations. The shape representations include parametric shapes for a limited range of standard elements and the ability to also include non-parametric shape for all other elements. The coordination view also provides the property sets, material definitions, and other alphanumeric information of the spatial structure, building, and building service elements. Furthermore, the current version of the coordination view is extended by the following three add-on views: 1) quantity take-off add-on view, 2) space boundary add-on view, and 3) 2D Annotation add-on view.

Among the different categories of data provided by the IFC coordination view, the data assigned to the property sets of building elements and the data defined by the quantity take-of add-on view tend to be the most applicable to the prototype DSS for vegetated roofing system selection. Table 4-6 summarizes the available data that can be retrieved from `IfcSlab` objects based on the IFC2x3 specification. In this case, it is assumed that the users defined roofs as roof slabs, which are objects of the `IfcSlab` class.

Table 4-6: VRSS Framework data that can be exchanged through the common IFC2x3 property sets

Framework input	IFC2x3 element	IFC2x3 Property sets	IFC2x3 Property name
Storm water retention			
Annual coefficient of discharge (Ψ_a)	-	-	-
Potential energy savings			
Project location			
Country	-	-	-
State	-	-	-
City	-	-	-
Proposed roof characteristics			
Net area	IfcSlab	Pset_QuantityTakeOff	NetArea
R-value of roof assembly	IfcSlab	Pset_SlabCommon	ThermalTransmittance
Solar reflectance	-	-	-
Infrared emittance	-	-	-
Energy costs and equipment efficiencies			
Summertime cost of electricity	-	-	-
Air conditioner efficiency (COP)	-	-	-
Energy source and cost for heating	-	-	-
Heating system efficiency	-	-	-
Approximate sound transmission class			
Sound Transmission Class (STC)	IfcSlab	Pset_SlabCommon	AcousticRating
Surplus death load of roof system			
Death load capacity of proposed roof project	-	-	-
Approximate weight of roof system	IfcSlab	Pset_QuantityTakeOff	GrossWeight
Potential contribute to LEED certification			
Sustainable Site (SS) Credit 2 is assigned	-	-	-
Comply w/ Sustainable Site (SS) Credit 5.1	-	-	-
Comply w/ Sustainable Site (SS) Credit 5.2	-	-	-
Comply w/ Sustainable Site (SS) Credit 6.1	-	-	-
Comply w/ Sustainable Site (SS) Credit 6.2	-	-	-
Comply w/ Sustainable Site (SS) Credit 7.2	-	-	-
Cost (special factor)			
Lifecycle costs	-	-	-

As shown in Table 4-6, there are certain data required by the VRSS Framework that are currently not available in the IFC data model. According to bSI (2011), the IFC2x3 provides the `IfcBuildingElementProxy` class, which is an extensible proxy that can be used by the users to exchange specific data. This technique may require user agreements to ensure that the users provide appropriate properties to the proxy element. In order to reduce user's time and effort, IFC support files should be developed with a specific property set which contains the roof system properties required by the framework as shown in Table 4-7. These files should be provided on the `Main` page of the prototype DSS. The IFC support files will be revisited in the next section of this document.

Table 4-7: VRSS Framework data that can be exchanged through the custom IFC2x3 property set

Reference roof properties	IFC2x3 element	IFC2x3 Property sets	IFC2x3 Property name
Annual coefficient of discharge (Ψ_a)	IfcSlab	Pset_VrssDss/Other	VrssDss_AnnualCoefficientOfDischarge
Approximate weight of roof system	IfcSlab	Pset_VrssDss/Other	VrssDss_ApproximateWeight
Infrared emittance	IfcSlab	Pset_VrssDss/Other	VrssDss_InfraredEmittance
Net area	IfcSlab	Pset_QuantityTakeOff	NetArea
R-value of roof assembly	IfcSlab	Pset_VrssDss/Other	VrssDss_RValue
Solar reflectance	IfcSlab	Pset_VrssDss/Other	VrssDss_SolarReflectance
Sound Transmission Class (STC)	IfcSlab	Pset_VrssDss/Other	VrssDss_SoundTransmissionClass

After the IFC data exchange format is selected to be used for exchanging building data between BIM authoring tools and the prototype DSS, it is important to examine how data can be exchanged using the

IFC physical data files. This use case scenario requires one to export an IFC file from an IFC compatible BIM authoring tool and upload the IFC file to the server that hosts the prototype DSS as shown in Figure 4-44.

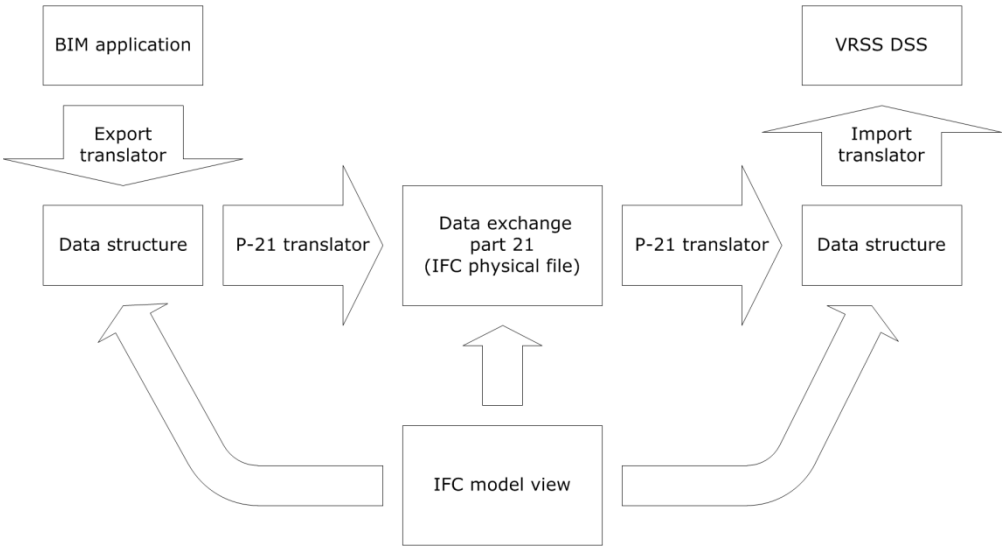


Figure 4-44: Data exchange using the IFC physical file adapted from Eastman et al. (2008)

As demonstrated in Figure 4-44, inside the BIM authoring tool, the export translator is responsible for assigning the data generated based on the tool native structure to appropriate IFC classes. Then the data in the form of IFC classes are translated to an IFC physical file used for transferring data to the prototype DSS. Fundamentally, an IFC data file is a clear text file, which follows the STEP physical file format defined in ISO10303-21:1994, also referred to as ISO10303 Part 21 (Liebich 2009). After the IFC file is generated, the file can be uploaded to the DSS server by the user. The translator embedded in the DSS is then responsible for translating data contained in the IFC file to IFC classes. Next, the data are translated by the import translator to data defined by the DSS data structure. The DSS then looks for the available data and displays the available inputs to the user through the user interface. Finally, the user can modify the inputs and process the decisions concerning vegetated roofing system selection. Similarly, this approach can be used to transfer data from the prototype DSS to BIM tools.

Within the process discussed above, it can be considered that the ISO10303 Part 21 (P-21) translators play an important role in converting the data contained in the IFC physical data files to data that can be used by the prototype DSS and vice versa. For data translation, the DSS employs the IFC Engine DLL library (research license) developed by researchers at RDF Ltd., based in Bulgaria. The library was originally developed to be used with the C++ programming language; however, the developers also developed some examples that show how the IFC Engine DLL can be used with other multipurpose programming languages such as Visual Basic .NET, C#, and Java. The IFC Engine DLL can be downloaded from the RDF Ltd. website at http://rdf.bg/index.html.

Resources

Among a vast numbers of BIM-based tools, the author decided to used GRAPHISOFT ArchiCAD 16 and AutoDesk Revit 2013 to examine the data exchanged between the prototype DSS and BIM authoring tools. ArchiCAD and Revit support IFC and are widely used building design practitioners. Both tools can also be installed on the computer for use in developing the prototype DSS. Furthermore, ArchiCAD and Revit can be used for educational purposes without additional expense, which is ideal for the prototyping project discussed in this chapter.

Between ArchiCAD and Revit, the author decided to use ArchiCad during the modeling and construction phases of Prototyping 2. The reason being, ArchiCAD is an IFC2x3 certified software application. It allows building objects to be managed and navigated using the IFC data structure. ArchiCAD also provides a variety of IFC import and export capabilities. Among such provided capabilities, Figure 4-45 demonstrates the ArchiCAD IFC Manager that can be used to manage and navigate through a building model using the IFC2x3 data structure.

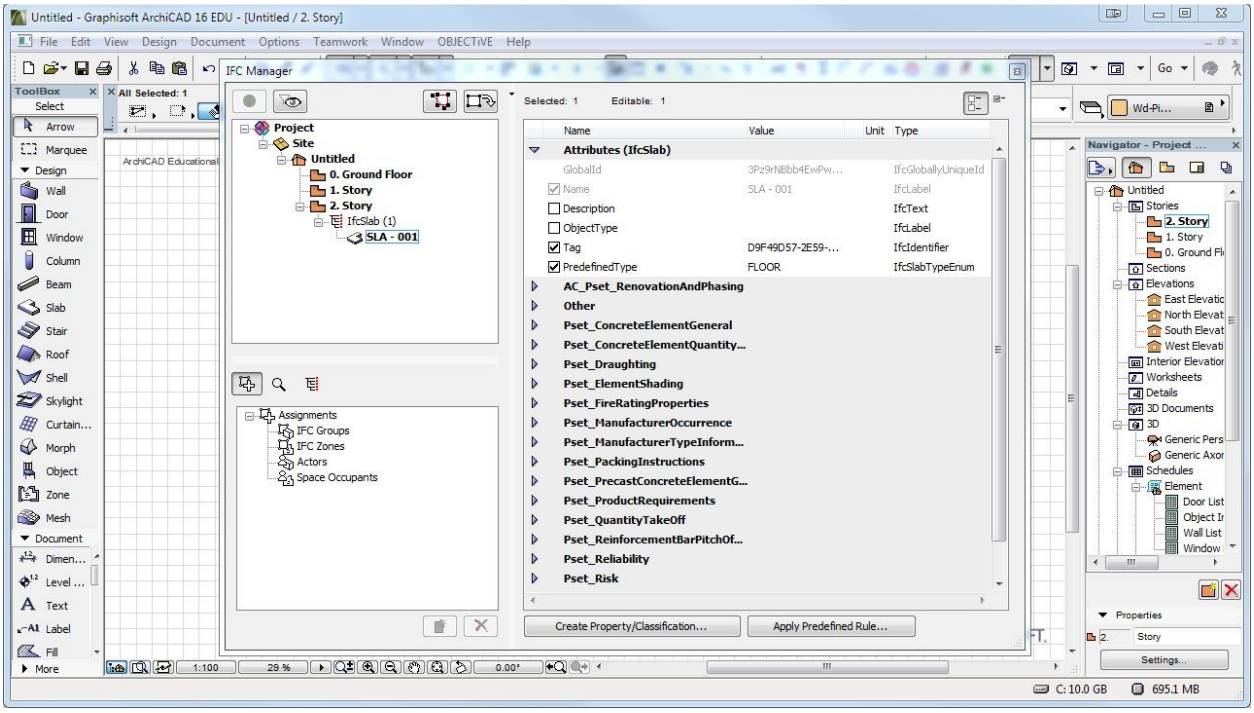


Figure 4-45: ArchiCAD IFC Manager

Compared to ArchiCAD, Revit has not yet been IFC2x3 certified by the time of this study and does not provide highly manageable IFC import and export functions. However, the communication phase indicates that Revit represents the BIM authoring tool that most of the research participants have used as part of their building projects. Therefore, it is important to examine the issue of data exchange between Revit and the prototype DSS after the data exchange between ArchiCAD and the prototype DSS is well examined.

Based on the available technology and resources, the prototype DSS seems to be feasible to be further developed with a goal to provide the end users with the BIM model upload and download functions. To achieve this goal, the author decided to implement IFC as a format for exchanging data between the prototype DSS and BIM authoring tools. Moreover, the author decided to use ArchiCAD and Revit for examining how data can be transferred from a BIM tool to the prototype DSS and vice versa using an IFC file.

4.3.3. Prototyping Cycle 2: Modeling in the Form of Quick Design

4.3.3.1. Prototype Cycle 2: Requirements Modeling

As mentioned in the previous sections, the goal of Prototyping Cycle 2 was to provide the IFC file upload and download capabilities for end users. To achieve this goal, it was essential to develop functional models to identify the user-operable functions and operations contained within the classes that serve such functions. This section discusses the functional model created during the modeling phase of Prototyping Cycle 2 in detail.

Figure 4-46 demonstrates the activity diagram that depicts the user-operable functions and system operations derived from the information provided in Section 4.3.1 and 4.3.2. The white colored actions represent the new functions and operations that were the main focuses of Prototyping Cycle 2. The light gray colored actions represent the existing functions and operations needed to be reexamined during Prototyping Cycle 2. The dark gray colored actions represent the existing functions and operations that were not affected by the new functions and operations.

In Figure 4-46, user-observable functions include the functions for uploading and downloading an IFC file to and from the prototype VRSS DSS. To provide these functions, the system should be able to extract data from an IFC file uploaded to the application server. The system then creates in-memory data and uses such data as initial inputs for the CBA Tabular method. From this point, the end user can continue to provide additional decision-making variables or modify the initial inputs. Next, the system is responsible for generating the report for the end user. After the end user comprehends the report, the user may exit the DSS or download an IFC file that represents the most appropriate green roof system for the user's project. If the end user chooses to download an IFC file, the system is then responsible for generating an IFC file and making it available for the user to download.

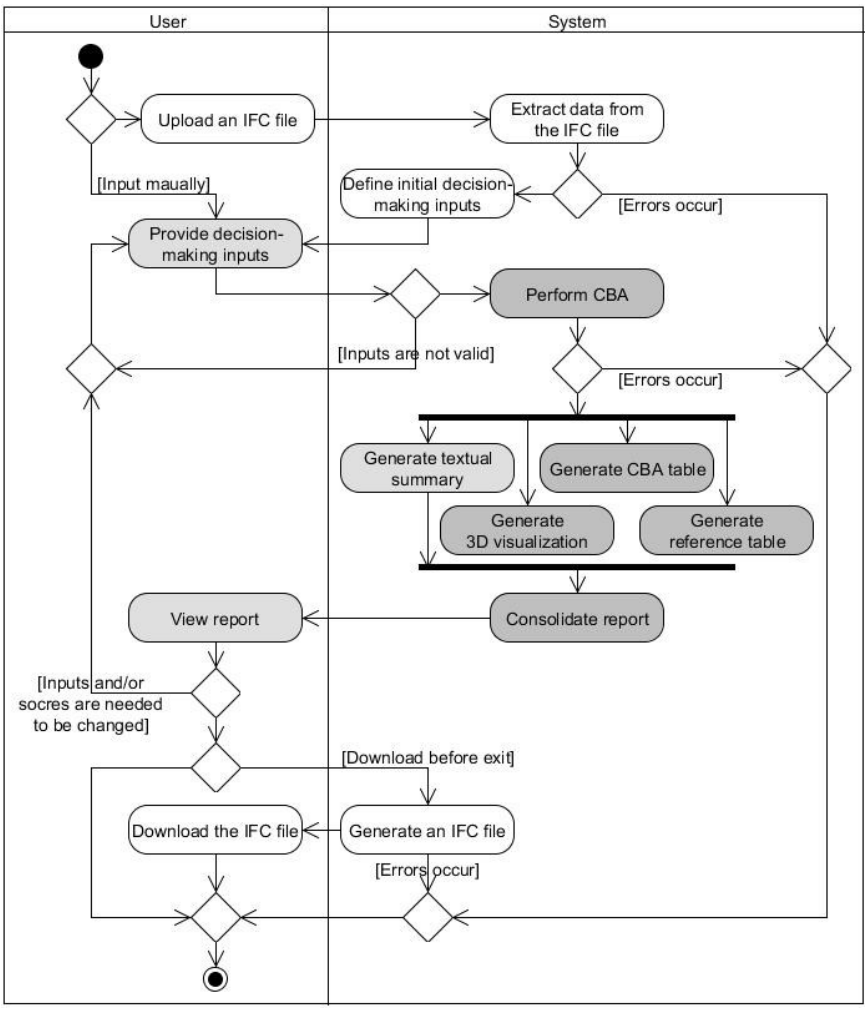


Figure 4-46: Functional requirements for Prototyping Cycle 2

Functional modeling performed during the modeling phase of Prototyping Cycle 2 gave useful information about how the prototype DSS should be developed to provide the IFC file upload and download capabilities for end users. Because the requirements for the new capabilities fell within the functional domain of project requirements, it is sufficient for the functional model to be developed and presented in this section. The understanding gained from the created functional model was used as a basis for designing and constructing the prototyping DSS, which will be discussed in the following sections.

4.3.3.2. Prototyping Cycle 2: Design Modeling

Fundamentally, making the prototype DSS IFC-compatible tended to have no impact on how the interface, aesthetic, content, navigation, and architecture of the DSS were designed. Therefore, the main focus of design modeling performed during Prototyping Cycle 2 was to design the prototype VRSS DSS at the component level. The component-level design was focused on how IFC support components should be added to the existing system is discussed below.

The component diagram shown in Figure 4-47 represents the component-level design for the prototype DSS developed during Prototyping Cycle 2. For this design scheme, a physical IFC file uploaded to the server is considered an external data source. The prototype DSS can access the uploaded IFC file through `IfcRead` that utilizes `IfcEngine`, the class that serves the operations provided by the IFC Engine DLL library discussed in Section 4.3.2.2. The prototype DSS can also generate an IFC file using the operations provided by `IfcWrite`. `IfcRead` and `IfcWrite` implement the interfaces allowing them to be loosely-coupled with `IfcData`, the class that serves all IFC support operations to the `VrssFramework`. `VrssFramework` can access `IfcData` through the `IIfcData` interface. As shown in Figure 4-47, all the components responsible for providing IFC related operations are contained in the `IfcBase` package.

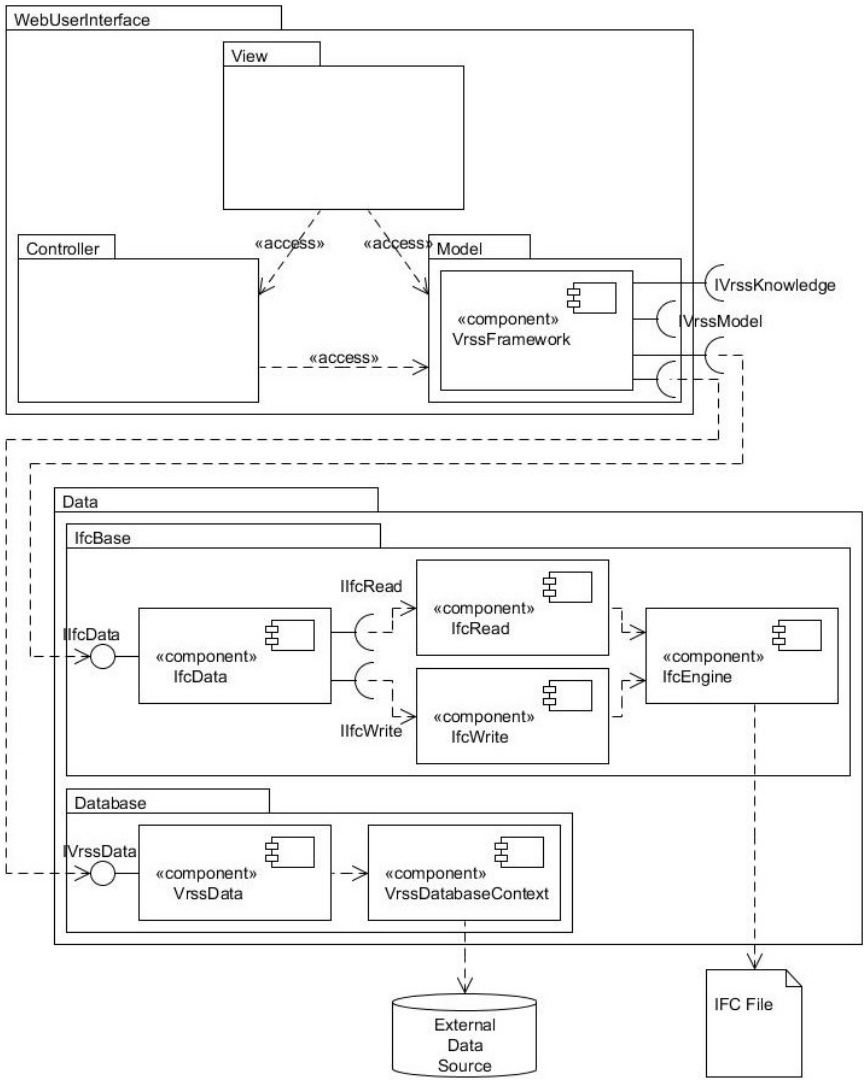


Figure 4-47: Component-level design for Prototyping Cycle 2

It is important to note that the way in which the prototype DSS was designed has made most of the system components independent from each other. In this case, the data management and user interface subsystems were the only two high-level components that needed to be modified. Therefore, the knowledge-based and model management subsystems are not presented in Figure 4-47. Furthermore, this design allows IFC support components to be added to the data management subsystem. The components can be later used by other DSS, which might be developed in the future, for reading and writing IFC files.

The design model discussed in this section provides useful information about how IFC support components should be built and added to the existing system. Because the previous design of the prototype DSS adopted the separation of concerns principle, it is quiet flexible for the DSS subsystems to be extended to accommodate the new components presented in this section. After the components that work together to provide the IFC support operations were well defined, Prototyping Cycle 2 can then move forward to the construction phase, which will be extensively discussed in the next section.

4.3.4. Prototyping Cycle 2: Prototype Construction

Based on the requirement and design models presented in Section 4.3.3.1 and 4.3.3.2, the construction phase of Prototyping Cycle 2 was comprised of three major tasks. The first task was to prepare IFC files for facilitating the constitution of the IFC support components. The second task was to build IFC support components and make them available through the data management subsystem. The third task was to add IFC support elements to the user interface of the prototype DSS. This section discusses how these tasks were accomplished during Prototyping Cycle 2 in detail.

IFC File Preparation

To build IFC support components, it is essential to create an IFC file or files for enabling the Test-Driven Development (TDD) extensively discussed in Section 4.2.4. Once the files became available, a test fixture and tests can then be developed to specify the behaviors of the IFC support classes. The procedure used for creating IFC files and the files created during the construction phase of Prototyping Cycle 2 is presented below.

As mentioned in Section 4.2.3.2, the author decided to use ArchiCAD as a primary tool for investigating how reference roof data can be exchanged between the prototype DSS and BIM authoring tools. To create IFC files, the author followed the instructions provided within the GRAPHISOFT's (2012) document, entitled IFC 2x3 Reference Guide for ArchiCAD 16. This document can be found at http://www.archicadwiki.com/IFC. In ArchiCAD, the custom roof properties can be added to the `IfcSlab` class/type, the super class of `IfcRoof`, using the IFC Scheme Setup tool as shown in Figure 4-48. IFC Scheme Setup allowed the author to create an IFC property set, which contains a series of custom properties, using the Create IFC Property tool. For example, the author can create an IFC property for the annual coefficient of discharge of a reference roof as presented in Figure 4-48. After all properties are

created, a reference roof, which is an instance of the `IfcRoof` type, can be created as shown in Figure 4-49.

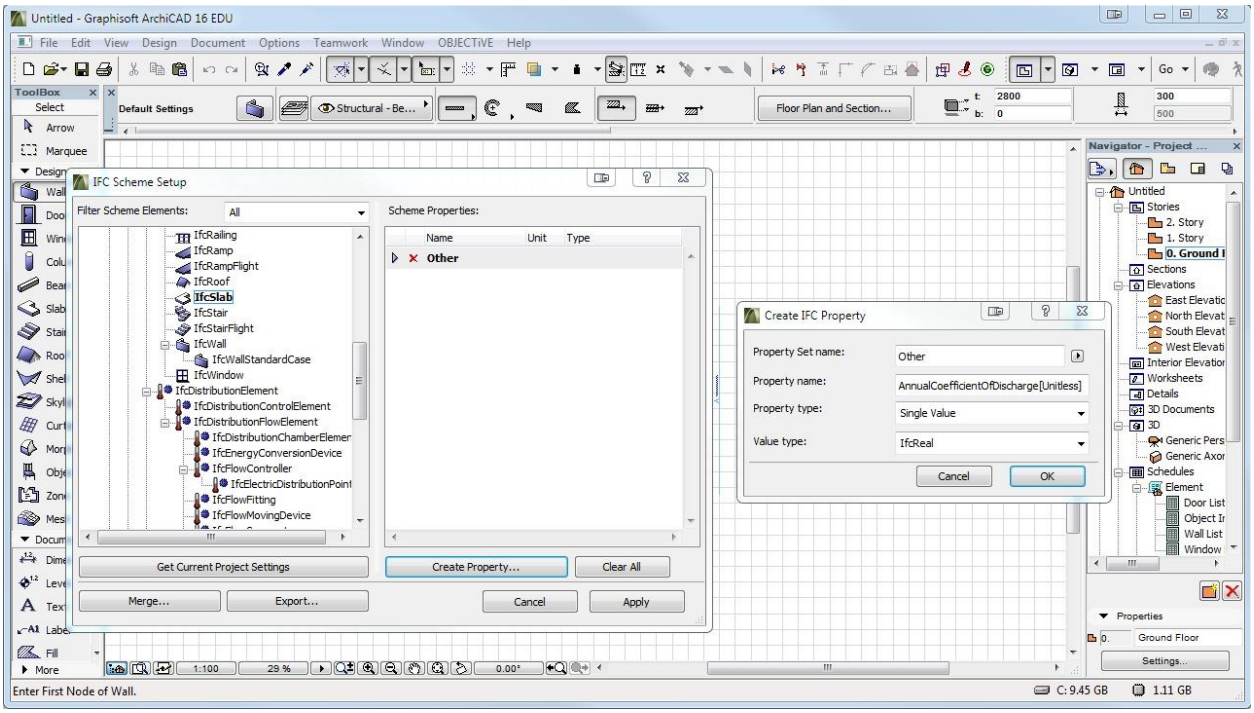


Figure 4-48: ArchiCAD IFC Scheme Setup

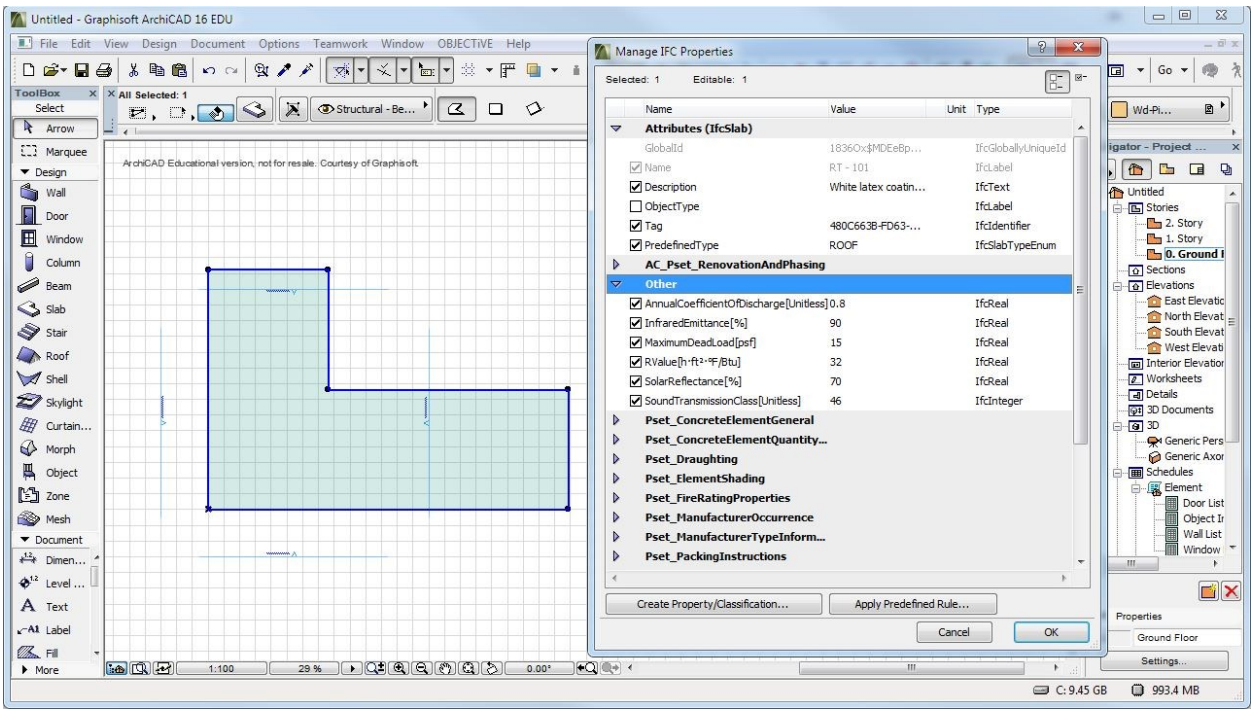


Figure 4-49: A reference roof

Also presented in Figure 4-48, IFC Scheme Setup has a capability for exporting the custom IFC scheme to be used in other projects. In other projects, the custom properties created above can be merged to an existing scheme using the Merge operation shown in Figure 4-48. In addition to IFC Scheme Set Up, ArchiCAD has the IFC Translation Setup tool that can be used for specifying how IFC files should be exported from or imported to ArchiCAD as demonstrated in Figure 4-50. These are some of the IFC support features provided by ArchiCAD, which allowed the author to share an IFC scheme that contains custom roof properties and an IFC translator with end users via the `Main` page of the prototype DSS.

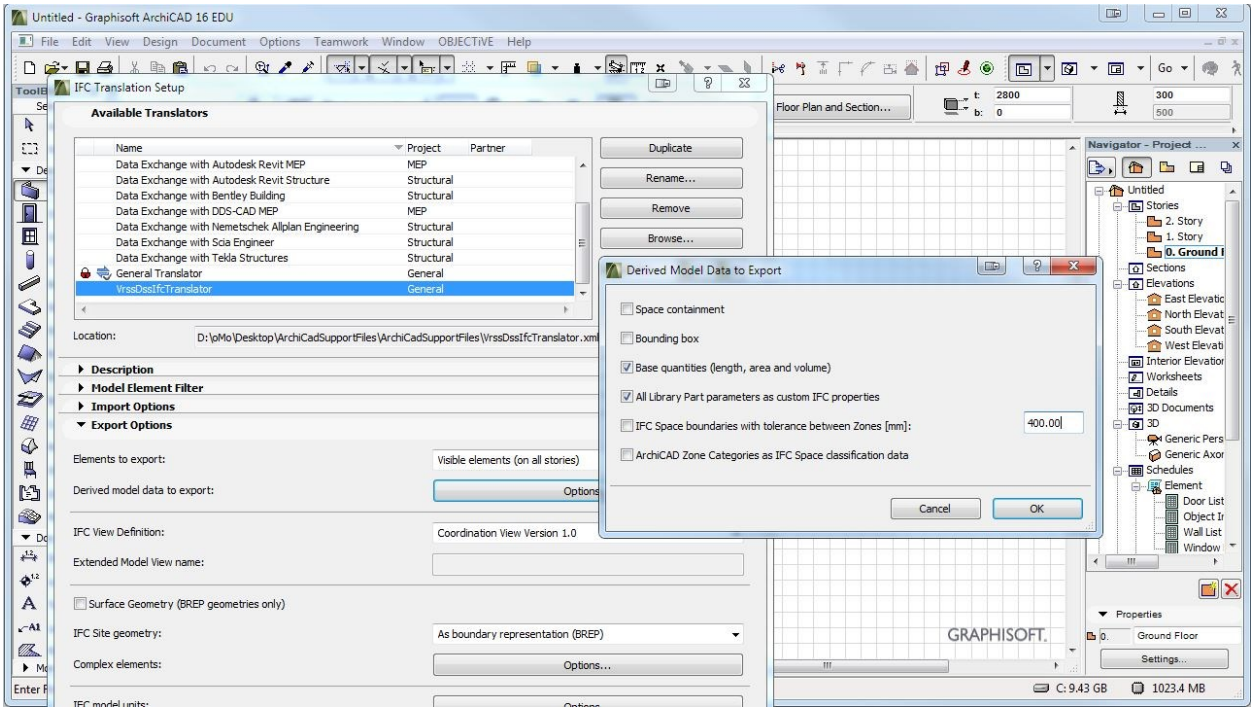


Figure 4-50: ArchiCAD IFC Translation Setup

By using the instructions discussed above, an IFC file that contains a reference roof was created and ready to be uploaded to the DSS application server. The information contained in the IFC file is partially presented in Box 4-6. Box 4-6 shows that the reference roof (#189) is an example of the `IfcSlab` class, which has a set of attributes such as name and description. This object has a set of properties (#229) previously defined in ArchiCAD, which contains the following items: #220, #224, #225, #226, #227, and #228. The property set is linked to the reference roof by the object of the `IfcRelDefinesByProperties` class (#231) as shown in Box 4-6.

Box 4-6: Information contained in the IFC file created using ArchiCAD

```
...
#189= IFCSLAB('2RvhfRp0D9butgUtho2Wam',#15,'RT - 001','White latex coating with highest
solar reflectance on 5" foam insulation',$,#136,#184,'9BE6BA5B-CC03-4997-8DEA-
7B7AF20A0930',.ROOF.);
#220= IFCPROPERTYSINGLEVALUE('AnnualCoefficientOfDischarge[Unitless]',$,IFCREAL(0.8),$);
#224= IFCPROPERTYSINGLEVALUE('InfraredEmittance[%]',$,IFCREAL(90.),$);
#225= IFCPROPERTYSINGLEVALUE('MaximumDeadLoad[psf]',$,IFCREAL(15.),$);
#226= IFCPROPERTYSINGLEVALUE('RValue[h\S\7ft\S\2\S\7\S\0F/Btu]',$,IFCREAL(32.),$);
#227= IFCPROPERTYSINGLEVALUE('SolarReflectance[%]',$,IFCREAL(70.),$);
#228= IFCPROPERTYSINGLEVALUE('SoundTransmissionClass[Unitless]',$,IFCINTEGER(46),$);
#229=
IFCPROPERTYSET('03BDrE$ZYA5ZBwsqhFOmE$',#15,'Other',$,(#220,#224,#225,#226,#227,#228));
#231= IFCRELDEFINESBYPROPERTIES('1Jv05PRMo0ztDUmzZJiSzU',#15,$,$,(#189),#229);
...
```

As mentioned earlier, the author also used Revit, which seemed to be the BIM-authoring tool most commonly used by research participants, to create IFC files. To create IFC files, the author followed the instructions provided by Autodesk, found at http://wikihelp.autodesk.com. In Revit, the custom properties can be created in the form of shared parameters using Edit Shared Parameters as shown in Figure 4-51. The shared parameters can be assigned to a particular project using Parameter Properties as presented in Figure 4-52. As shown in Figure 4-52, the shared parameters are assigned to the Roof type. Afterward, a reference roof, an instance of the Roof type that has all the custom properties, can be created as presented in Figure 4-53.

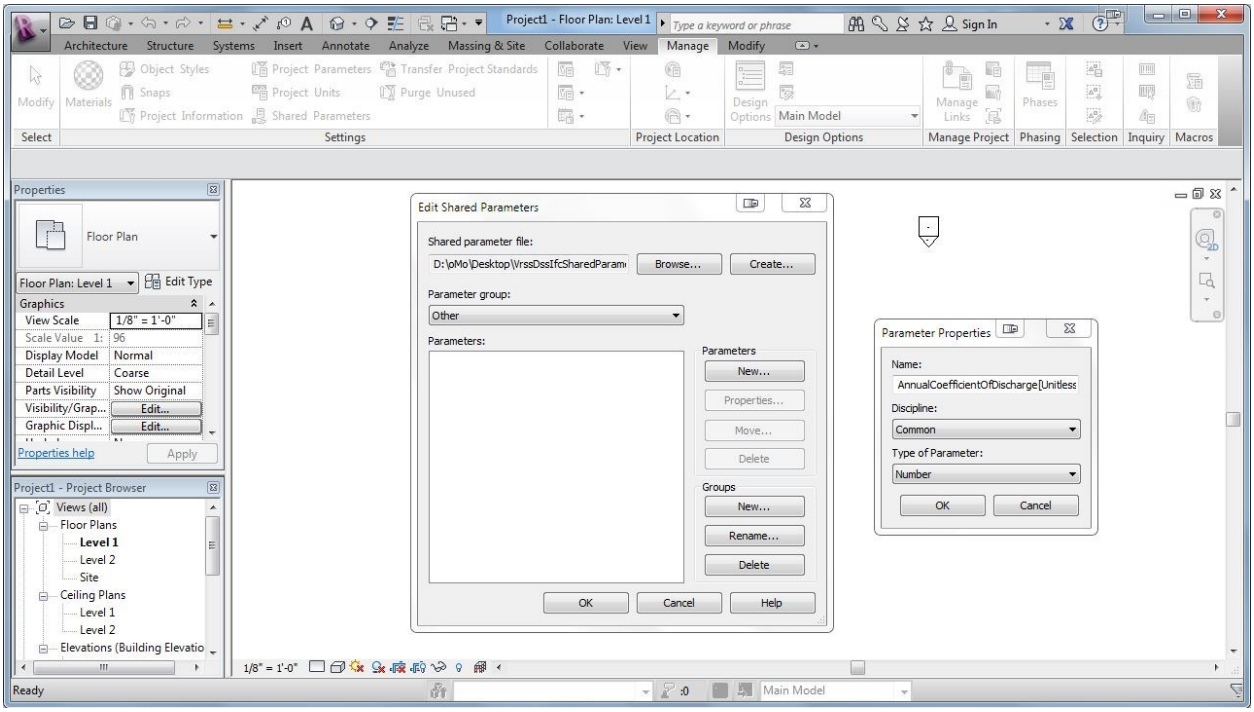


Figure 4-51: Revit Edit Shared Parameters

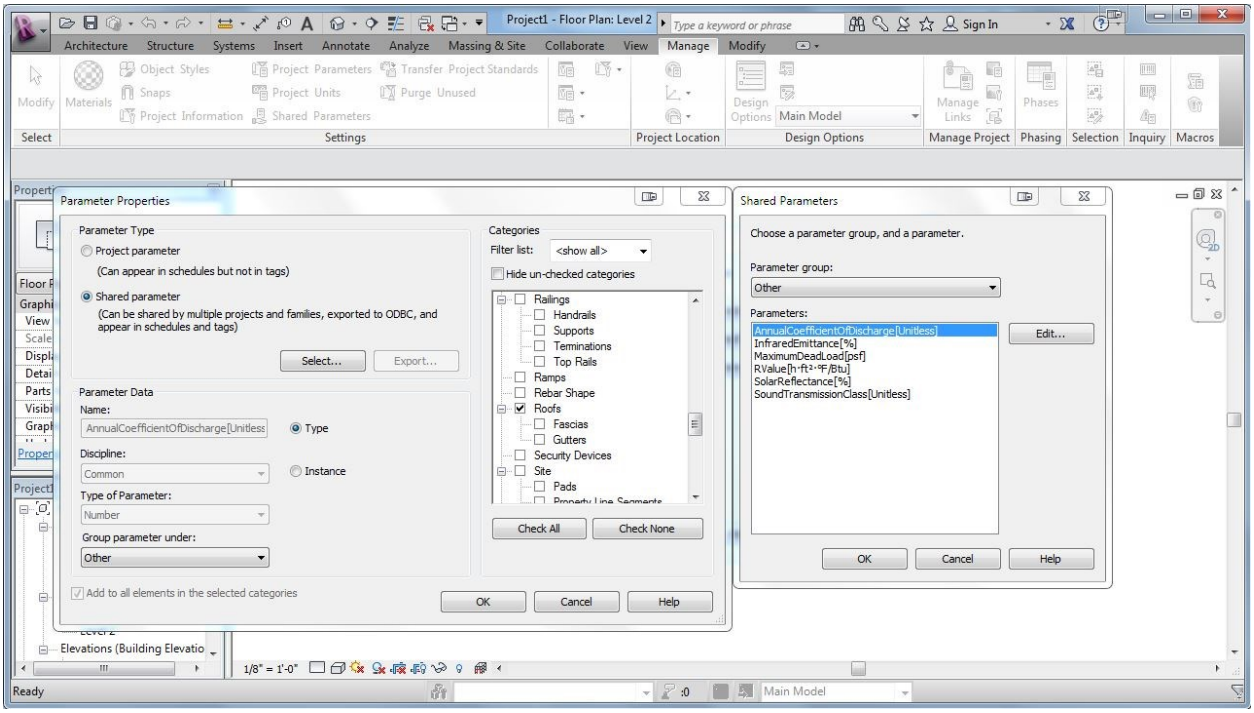


Figure 4-52: Revit Parameter Properties

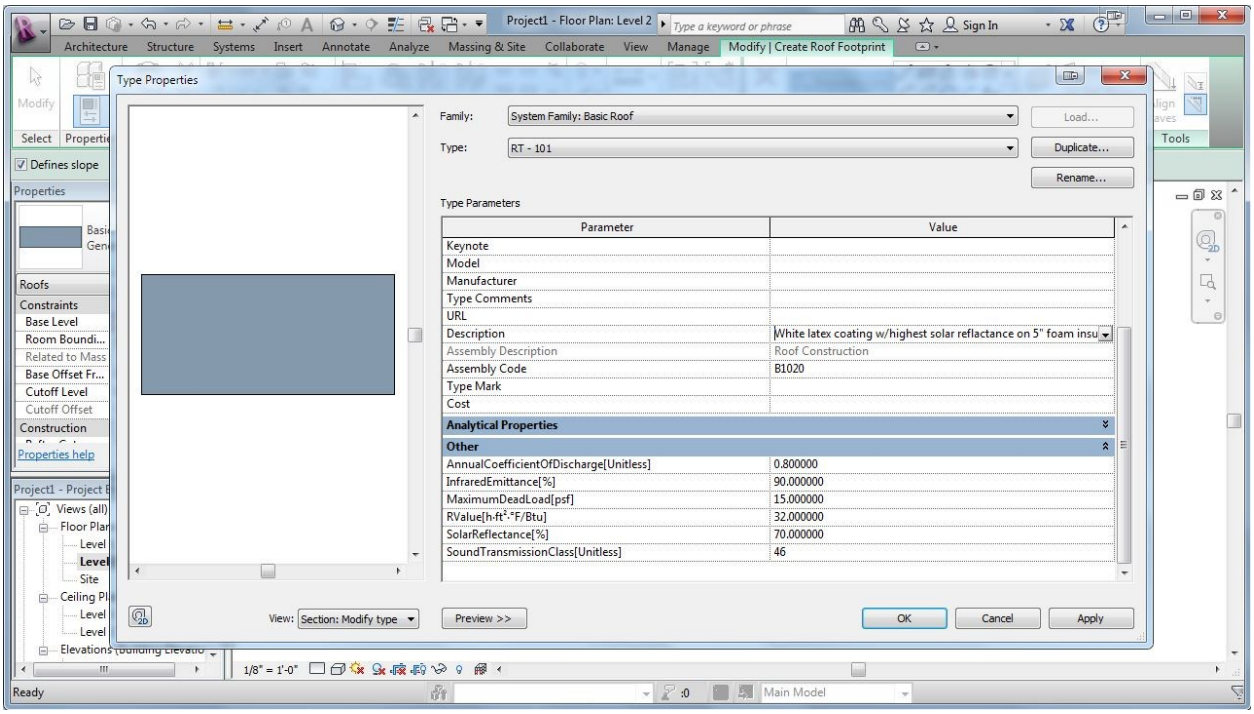


Figure 4-53: A reference roof created in Revit

To export an IFC file, it is essential to examine how the IFC translator is set up in Revit. Figure 4-54 demonstrates the IFC export layers implemented by Revit for exporting IFC files. As shown in Figure 4-54, the IFC Export Classes tool does not have an operation for modifying the presented IFC scheme,

which does not contain the custom roof properties. However, the IFC export layers can be saved onto the computer hard drive and edited individually using a text editor program. For this project, the author used Notepad++ as a text editor tool for assigning the custom properties to the IFC export layers as shown in Figure 4-55. Afterward, the edited IFC export layers that contain all the custom properties can be loaded back to Revit using IFC Export Classes as presented in Figure 4-56. Finally, the reference roof created in Revit can be exported as a physical IFC file as demonstrated in Figure 4-57. The reference roof information contained in the IFC file is partially presented in Box 4-7.

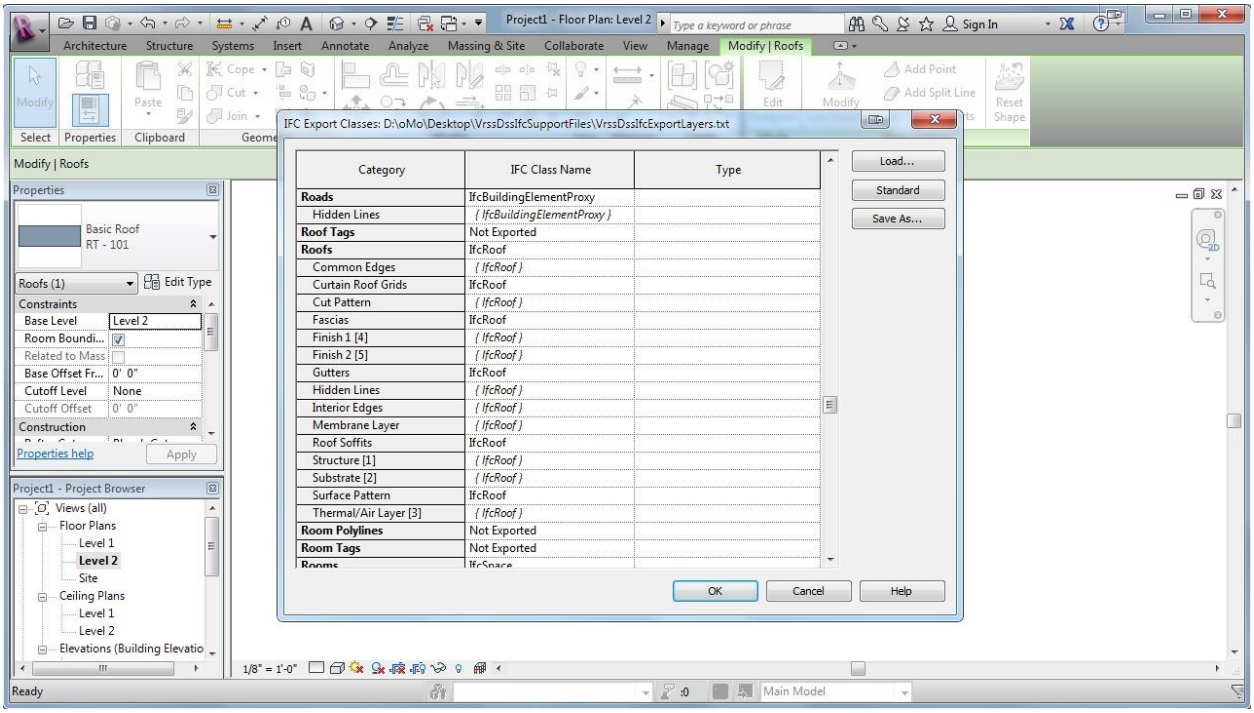


Figure 4-54: Revit initial IFC export classes

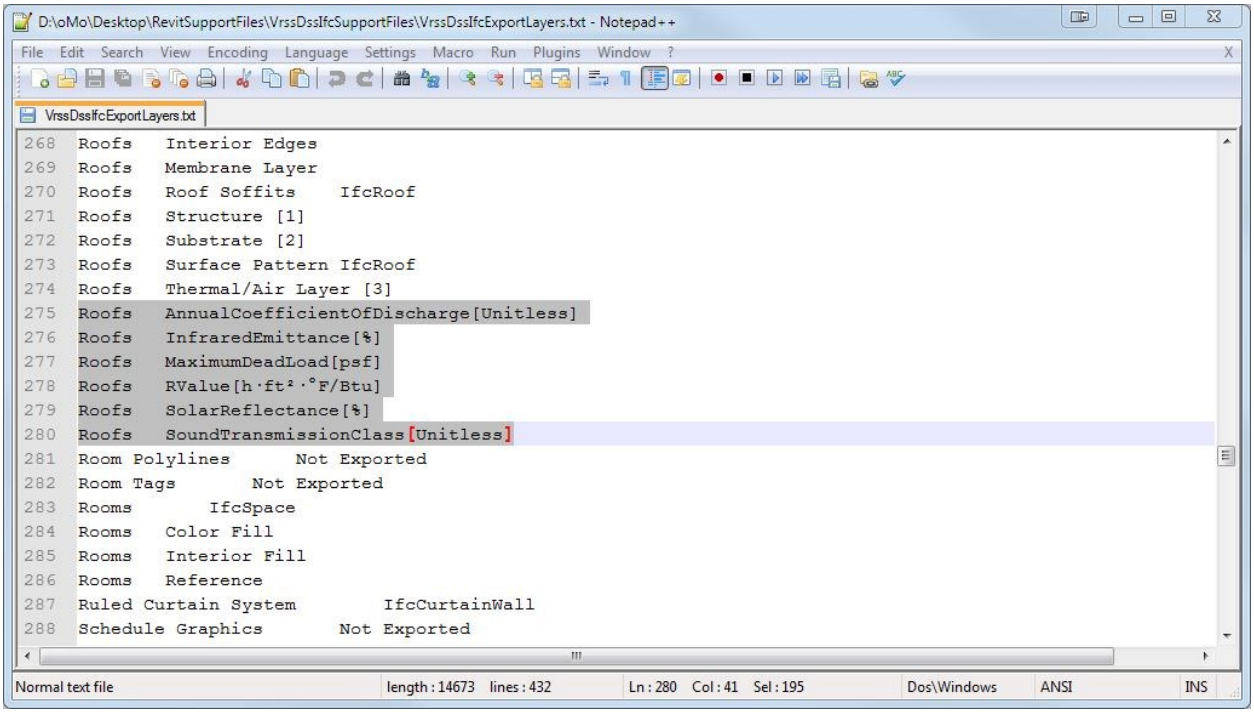


Figure 4-55: Additional roof properties

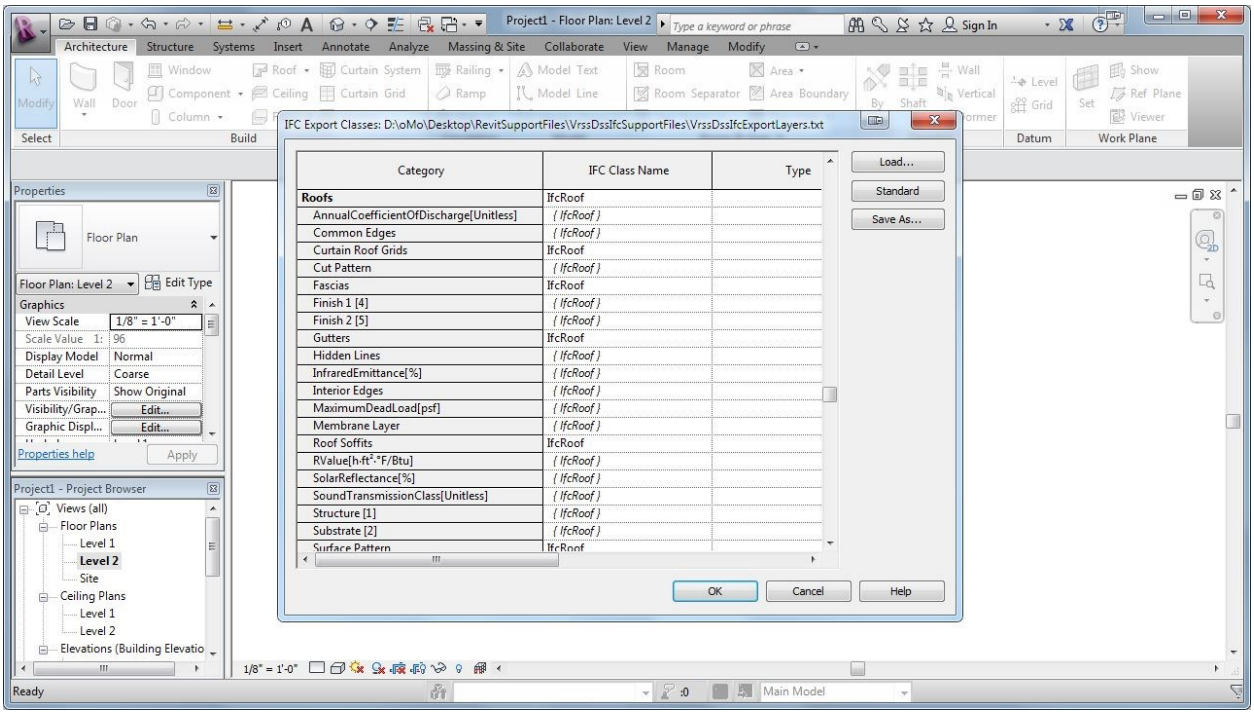


Figure 4-56: Revit IFC export with additional roof properties

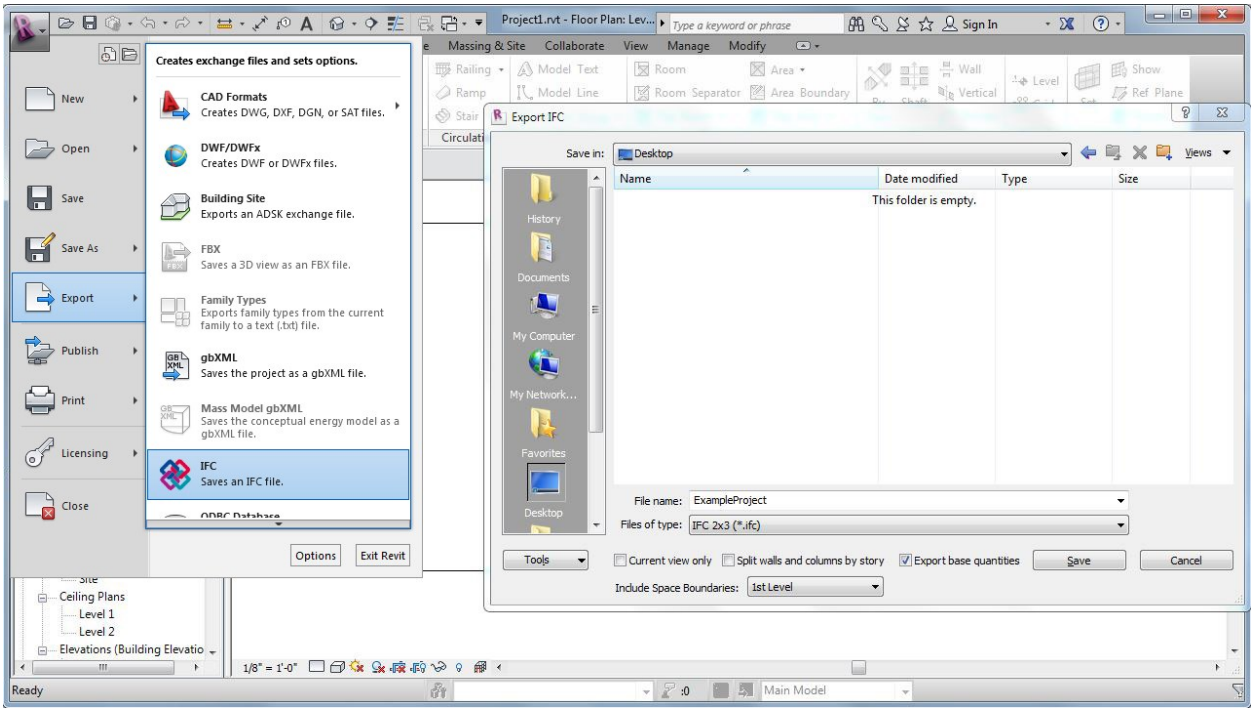


Figure 4-57: Revit IFC export

Box 4-7: Information contained in the IFC file created using Revit

```
...
#113= IFCSLAB('3d4uN7F0bEAgja$ixlxId_',#61,'Basic Roof:RT - 101:181752',$,'Basic Roof:RT
- 101',#81,#111,'181752',.ROOF.);
#128= IFCPROPERTYSINGLEVALUE('SolarReflectance[%]',$,IFCREAL(70.),$);
#129= IFCPROPERTYSINGLEVALUE('SoundTransmissionClass[Unitless]',$,IFCINTEGER(46),$);
#130= IFCPROPERTYSINGLEVALUE('RValue[h\X2\00B7\X0\ft\X2\00B200B700B0\X0\F/Btu]',$,
IFCREAL(32.),$);
#131= IFCPROPERTYSINGLEVALUE('AnnualCoefficientOfDischarge[Unitless]',$,IFCREAL(0.8),$);
#134= IFCPROPERTYSINGLEVALUE('MaximumDeadLoad[psf]',$,IFCREAL(15.),$);
#136= IFCPROPERTYSINGLEVALUE('InfraredEmittance[%]',$,IFCREAL(90.),$);
#163= IFCPROPERTYSET('3XiSdyYqb9WAsyQ9dL1DIS',#61,'Other',$,(#128,#129,#130,#131,#134,
#136));
#276= IFCRELDEFINESBYPROPERTIES('1mgis5xLn9cxLeWE9EbTd9',#61,$,$,(#79,#113),#163);
...
```

While the IFC file can be successfully created in and exported from Revit, it is important to exam how the process above can be simplified to encourage end users to use the prototype DSS in conjunction with Revit. It is quite fortunate that AutoDesk has provided a procedure for transferring custom IFC properties across Revit projects, which can be found at http://revit.autodesk.com/library/html/index.html. Based on this procedure, the author was able to create a Revit file used for transferring custom IFC properties as presented in Figure 4-58. This Revit file was made available for download on the `Main` page of the prototype DSS together with and the IFC export layers and shared parameters files.

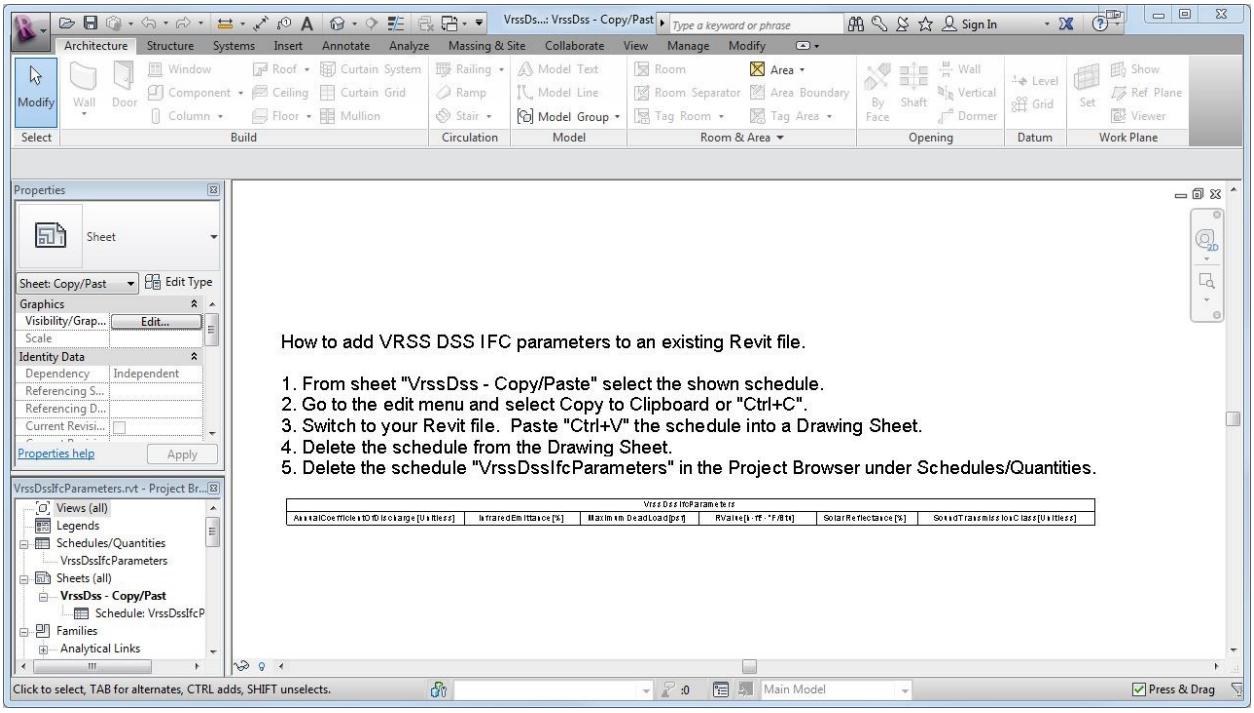


Figure 4-58: Revit file for custom transferring IFC properties

The information above shows that the custom roof properties required by the VRSS Framework discussed in Section 4.2.3.2 can be injected to an IFC file used for exchanging data between the prototype DSS and BIM-authoring tools. In addition to these required roof properties, it is important to examine how to retrieve the data about roof assemblies and use such data as a basis for generating green roof models for end users to download. To narrow down the scope of data, the author decided to retrieve only the data that describes roof geometry. This data can be used for creating a simple, conceptual green roof model and generating the 3D visualization element presented on the `Report` page of the prototype DSS.

By studying multiple IFC files exported from ArchiCAD and Revit, the author found that ArchiCAD and Revit implement the IFC standard for describing building elements differently. For example, when exporting an IFC file that describes a simple rectangle roof, ArchiCAD uses the `IfcArbitraryClosedProfileDef` class to describe the shape of the rectangle roof as presented in Box 4-8, item #128. In contrast, Revit uses the `IfcRectangleProfileDef` class to do the same task as shown in Box 4-9, item #83. Therefore, when writing the code for the class that provides the operations for reading data from IFC files, it is important to account for the incompatibility issues caused by how ArchiCAD and Revit implement the IFC standard.

Box 4-8: Information about roof geometry contained in the IFC file created using ArchiCAD

```
...
#128= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'',#126);
#138= IFCEXTRUDEDAREASOLID(#128,#135,#136,300.);
#148= IFCSHAPEREPRESENTATION(#116,'Body','SweptSolid',(#138));
#158= IFCPRODUCTDEFINITIONSHAPE($,$,(#148));
#163= IFCSLAB('0uPFiFGP50pe0CORJtg2q6',#15,'RT - 101',$,$,#114,#158,'3864FB0F-4191-40CE-
800C-61B4F7A82D06',.ROOF.);
...
```

Box 4-9: Information about roof geometry contained in the IFC file created using Revit

```
...
#83= IFCRECTANGLEPROFILEDEF(.AREA.,$,#82,74.,69.);
#87= IFCEXTRUDEDAREASOLID(#83,#86,#15,1.);
#96= IFCSHAPEREPRESENTATION(#53,'Body','SweptSolid',(#87));
#98= IFCPRODUCTDEFINITIONSHAPE($,$,(#96));
#100= IFCSLAB('2hXjc3cvr1Kw8WsnJvfWud',#61,'Basic Roof:RT - 101:182473',$,'Basic Roof:RT
- 101',#81,#98,'182473',.ROOF.);
...
```

After IFC files were well created, the test fixture and tests can be created to specify the behaviors of the classes that serve IFC support operations as presented in Figure 4-59. Based on such test fixture and tests, Box 4-10 demonstrates a piece of pseudo code for testing the `ReadIfcFile()` operation. This example pseudo code was written based on the Arrange/Act/Assert (A/A/A) pattern discussed in Section 4.2.4. The first step in this test is to define the paths to IFC files containing a reference roof and schema files. Next, the second step is to extract data from the IFC file. Finally, the last step is to validate that the IFC file contains only one roof entity. After the test fixture was developed based on the diagram presented in Figure 4-59, it can be used for enhancing the construction of classes and interfaces that work together to provide IFC support operations based on the defined behaviors.

Throughout the IFC files preparation process, the author was able to create a series of IFC files using both ArchiCAD and Revit. These files were used for enhancing the coding and testing tasks performed during the construction phase of Prototyping Cycle 2. The author also developed test fixture and tests for specifying the behaviors of IFC support components. The construction of IFC support components will be discussed further in this section.

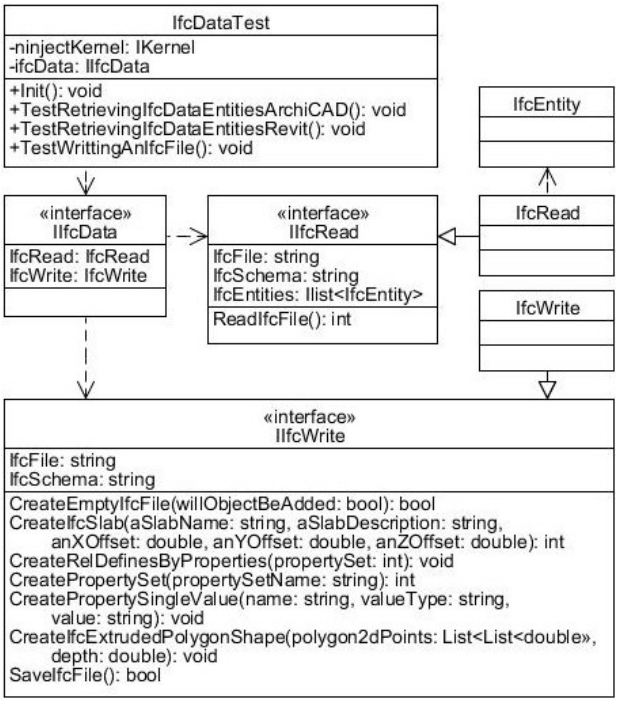


Figure 4-59: As built class diagram for `IfcDataTest`

Box 4-10: Pseudo code for testing the `ReadIfcFile()` operation

```
//Arrange
Define an IFC file path
Define a schema path

//Act
Read the IFC file

//Assert
Is the number of entity equal to 1
```

Data Management Subsystem

As mentioned earlier, the author saw an IFC file as an external data source similar to a database. Therefore, the construction phase of Prototyping Cycle 2 involved writing the code for IFC support components, testing the code to identify possible errors, and making the components available through the data management subsystem. The information about how these tasks were performed is provided below.

Figure 4-60 presents a class diagram developed based on the component-level design presented in Section 4.3.3.2. In Figure 4-60, `VrssFramework` accesses IFC support functions through `IIfcData` that utilizes `IIfcRead` and `IIfcWrite`. `IIfcRead` is an interface implemented by `IfcRead`, the concrete class that serves operations for retrieving data from IFC files. `IIfcWrite` is an interface implemented by `IfcWrite`, the concrete class that provides operations for creating IFC files.

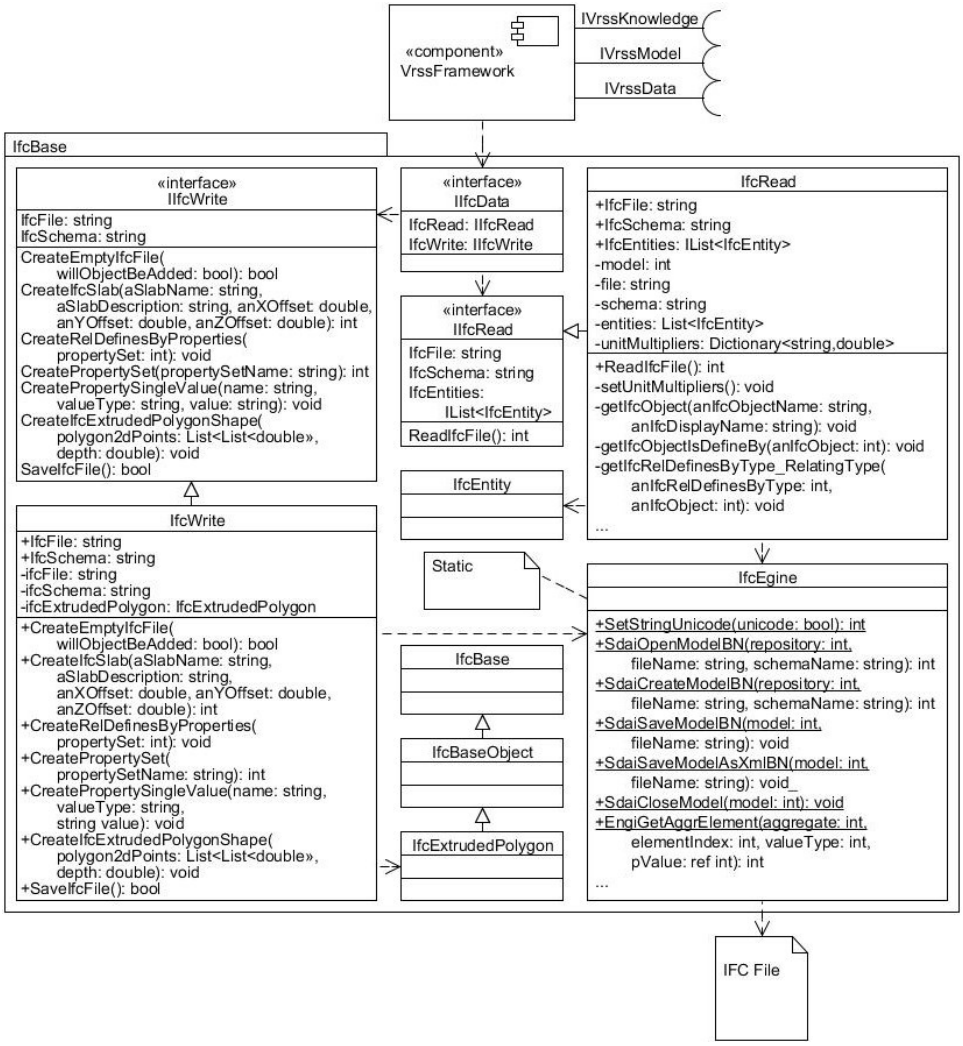


Figure 4-60: As built class diagram for IFC support components

As shown in Figure 4-60, `IfcRead` uses `IfcEngine`, a static C# class that serves the functions provided by the IFC Engine DLL library discussed in Section 4.3.2.2. `IfcEngine` was built based on an example C# class, `Example.cs`, came with the IFC Engine DLL package. This example file demonstrated how to implement IFC Engine DLL, which is a C++ library, in C# programming environment. The programming principle used in the example file is presented in Box 4-11.

According to MSDN (2013), the Common Language Runtime (CLR) provides the Platform Invocation Services (PInvoke). PInvoke allows managed code to call C-style functions in native Dynamic-Linked Libraries (DLLs). As presented in Box 4-10, the functions provided by IFC Engine DLL can be called using the `DllImport` attribute from the code written in C#. After the necessary IFC Engine DLL functions become available through `IfcEngine`, `IfcRead` can then access those functions using the operations served by `IfcEngine` for retrieving data from an IFC file as shown in Box 4-10. The data extracted from an IFC file then becomes memory-resident data represented by `IfcEntity`. `IfcEntity` was built in the

form of POCO class that contains multiple inter POCO classes. When writing the code for `IfcEntity`, the author strictly implemented the data modeling concept defined by IFC2x3 TC1 specification, found at http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm, to enhance data interoperability. Once the in-memory data becomes available, `VrssFramework` can then use the data to define the inputs required by the VRSS Framework.

Box 4-11: PInvoke and `DllImport` attribute

```
static class IfcEngine {
    [DllImport("IFCEngine.dll", EntryPoint = "setStringUnicode")]
    public static extern int SetStringUnicode(bool unicode);

    [DllImport("IFCEngine.dll", EntryPoint = "setOpenModelBN")]
    public static extern int SdaiOpenModelBN(int repository, string filename,
        string schemaName);

    [DllImport("IFCEngine.dll", EntryPoint = "setCreateModelBN")]
    public static extern int SdaiCreateModelBN(int repository, string filename,
        string schemaName);

    [DllImport("IFCEngine.dll", EntryPoint = "setSaveModelBN")]
    public static extern void SdaiSaveModelBN(int model, string filename);

    [DllImport("IFCEngine.dll", EntryPoint = "setSaveModelAsXmlBN")]
    public static extern void SdaiSaveModelAsXmlBN(int model, string filename);

    [DllImport("IFCEngine.dll", EntryPoint = "setCloseModel")]
    public static extern void SdaiCloseModel(int model);

    [DllImport("IFCEngine.dll", EntryPoint = "engiGetAggrElement")]
    public static extern int EngGetAggrElement(int aggregate, int elementIndex,
        );

    ...
}
```

In Figure 4-60, `IfcWrite` also uses `IfcEngine` for creating IFC files. To write an IFC file, the author followed an example IFC writing program written in C++ that came with the IFC Engine DLL package. Based on this example program, the author was able to generate an IFC file that complies with IFC2x3 TC1 specification using the functions provided by IFC Engine DLL.

As shown in Figure 4-60, `IfcWrite` also uses `IfcEngine` to access the IFC Engine DLL functions. Fundamentally, an IFC file can be generated by adding a series of IFC elements to an empty IFC file, which is actually a text file that can be created manually using text editor programs. According to the example program, the IFC elements can be added hierarchically using operations served by multiple classes, which were connected using a hierarchical structure, as presented in Figure 4-60. For example, `IfcBasic` is responsible for creating an empty IFC file and adding basic information about the IFC file and building project. Next, `IfcBaseOject` is responsible for adding building elements to the IFC file. Lastly, `IfcExtrudedPolygon` is responsible for adding the geometry of the building elements to the IFC file. Often, in these classes, the author had to use unsafe code and pointers as part of the code for facilitating direct memory manipulation as shown in Box 4-12.

Box 4-12: Unsafe code and pointers

```
public class IfcExtrudedPolygon : IfcBaseObject {
    ...
    public unsafe int CreateIfcSlab(string aSlabName, string aSlabDescription,
        double anXOffset, double anYOffset, double anZOffset) {
        TransformationMatrix matrix;
        int ifcSlabInstance = 0;
        double xOffset = anXOffset, yOffset = anYOffset, zOffset = anZOffset;
        string slabName = aSlabName, slabDescription = aSlabDescription;

        IdentityMatrix(&matrix);
        matrix._41 = xOffset;
        matrix._42 = yOffset;
        matrix._43 = zOffset;

        //Build Slab and add it to the BuildingStorey
        ifcBuildingStoreyInstancePlacement =
            IfcBuildingStoreyInstancePlacement;
        aggrRelatedElements = AggrRelatedElements;
        if (ifcBuildingStoreyInstancePlacement != 0) {
            fixed (int* pIfcSlabInstancePlacement = &ifcSlabInstancePlacement) {
                ifcSlabInstance = buildSlabInstance(&matrix,
                    ifcBuildingStoreyInstancePlacement,
                    pIfcSlabInstancePlacement, slabName, slabDescription);
                IfcEngine.SdaiAppend(aggrRelatedElements, 6, ifcSlabInstance);
                IfcEngine.SdaiAppend(aggrRelatedObjects, 6, ifcSlabInstance);
            }
        }
        return ifcSlabInstance;
    }
    ...
}
```

Based on the software functions provided by the IFC Engine DLL, the author was able to build the IFC support components discussed above and provide them via the data management subsystem. The IFC Engine DLL functions can be accessible using multiple coding principles including PInvoke, `DllImport` attribute, unsafe code, and pointers. The IFC support components constructed as part of Prototyping Cycle 2 enabled the prototype VRSS DSS to extract data from physical IFC files and to create IFC files that describe green roof systems for end users to download.

User Interface (Dialog) Subsystem

After the IFC support components were built, the last task of the construction phase of Prototyping Cycle 2 was to modify the prototype DSS user interface developed during Prototyping Cycle 1. Previously in Prototyping Cycle 1, the user interface was developed as an ASP.NET MVC application, which comprises three major components: model, view, and controller. The changes made to the user interface components are extensively discussed here.

According to the requirements discussed in Section 4.3.3.1, the user interface should provide end users with web form controls for uploading an IFC file, selecting a reference roof system from roof systems described in the IFC file, and presenting the properties of the chosen reference roof to end users. The author decided to add IFC support controls and control logic for switching the controls onto the original `Background` page of the DSS presented in Figure 4-61. In Figure 4-61, end users can request IFC support controls by clicking on the provided action link, "Use Building Information Modeling (BIM) data".

Afterward, the `Background` page with the IFC support controls will be presented to the end users as shown in Figure 4-62. The control logic mentioned earlier is presented in the form of simplified code in Box 4-13.

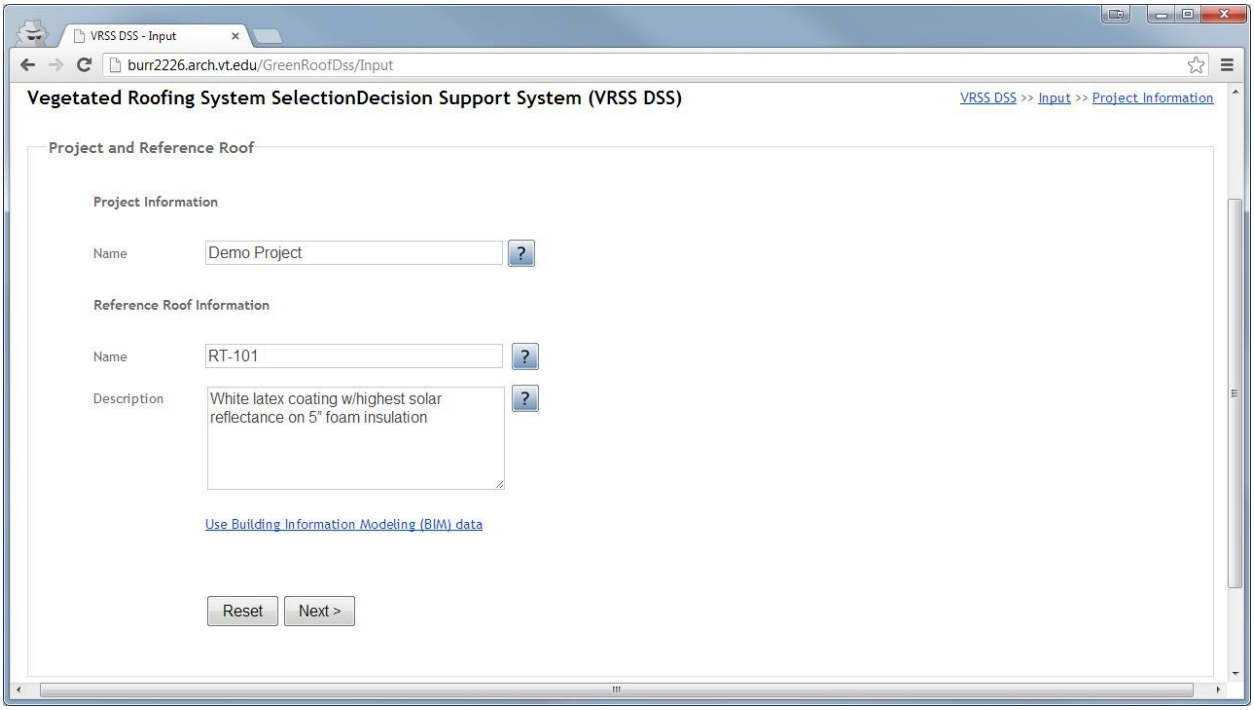


Figure 4-61: Original `Background` page

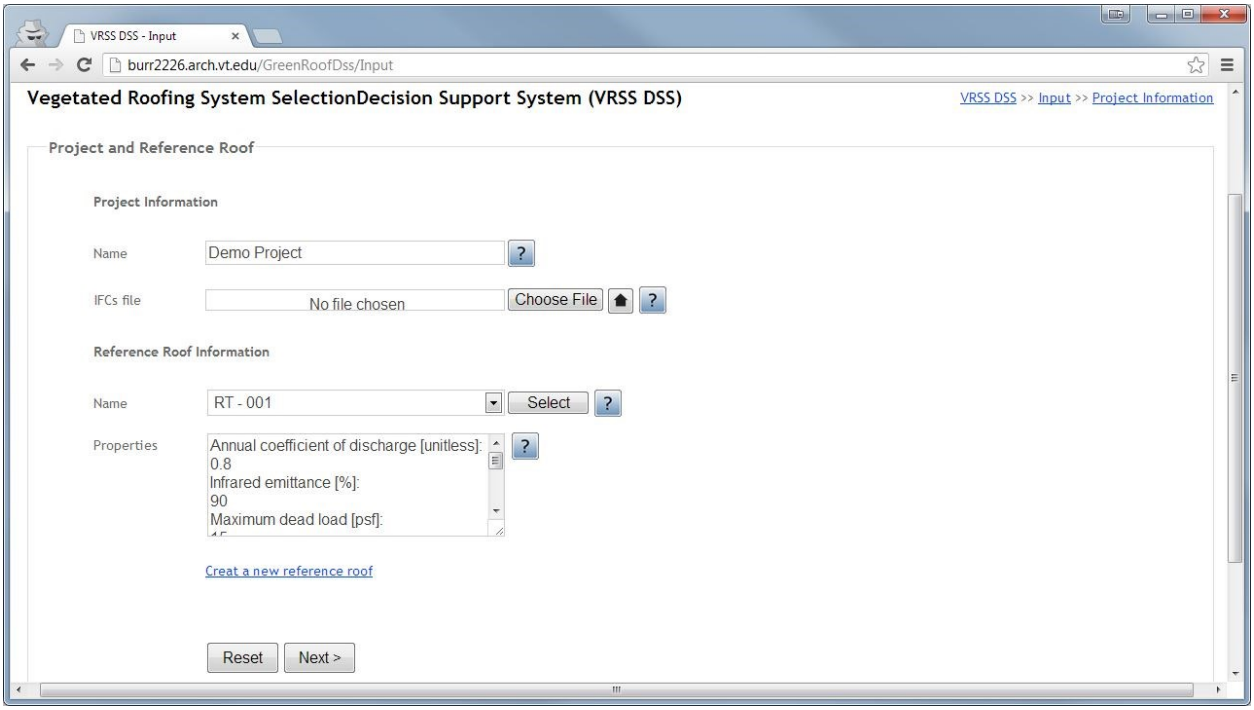


Figure 4-62: Modified `Background` page

Box 4-13: Manual and IFC support web controls

```
...
@if (!Model.Background.IsIfcUploaded) {
    //present web controls for manual inputs
} else {
    //present IFC support controls
}
...
```

To present the reference roof information extracted from an IFC file on the `Background` page, it is essential to modify the view model responsible for providing information for the IFC support controls discussed above. For example, the code in Box 4-13 shows that the `Background` property of the view model (`Model`) requires the `IsIfcUploaded` property, which helps identify which set of controls that should be presented to end users. The additional POCO classes built to support the view model is presented in Figure 4-63 whereby `VrssInput` is referred to as `Model` in Box 4-13. In Figure 4-63, the light gray colored classes are the existing classes built during the construction phase of Prototyping Cycle 1. The white colored classes in Figure 4-63 are the classes needed to be modified or built during the construction phase of Prototyping Cycle 2.

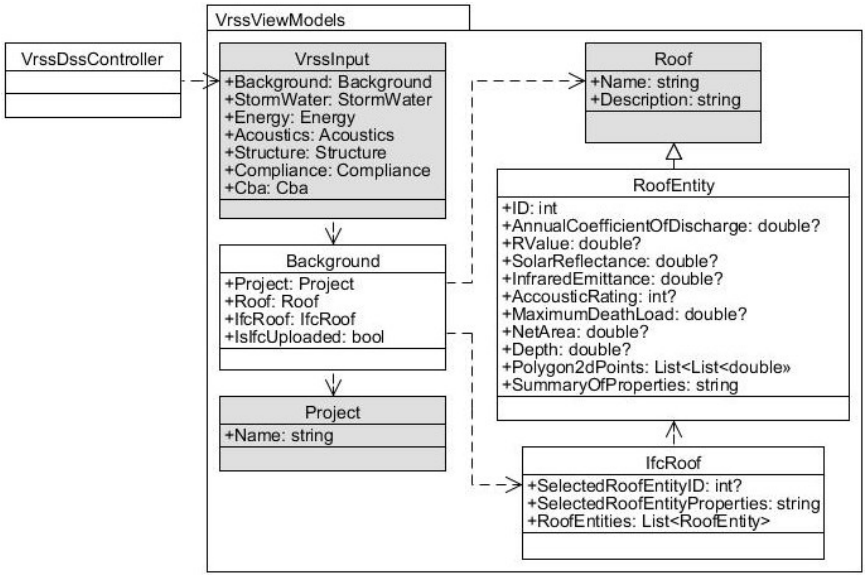


Figure 4-63: Additional view model classes

After the view and model components were modified, the last step was to modify the controller method that interacts with the view model to gather necessary information and passes the information onto the view. For instance, Box 4-14 shows a partial code for the `Background` method of `VrssDssController`. The code shows that one of the method parameters is the identification number of the reference roof. This number is used for retrieving roof properties from the uploaded IFC file. After the information about the reference roof is gathered, `VrssDssController` then passes the information

back to the `Background` page, which is responsible for presenting reference roof information as shown in Figure 4-62.

Box 4-14: Partial code for the `Background` method of `VrssDssController`

```
...
public virtual ActionResult Background(
    [Bind(Prefix="Main.Project.Name")] string aProjectName,
    [Bind(Prefix="Main.Roof.Name")] string aRoofName,
    [Bind(Prefix="Main.Roof.Description")] string aRoofDescription,
    HttpPostedFileBase anIfcFile,
    [Bind(Prefix="Main.IfcRoof.SelectedRoofEntityID")] int? theSelectedRoofEntityID) {
    ...
    if (selectedRoofEntityID.HasValue) {
        vrssInput.Backgorund.IfcRoof.SelectedRoofEntityID = selectedRoofEntityID;
        vrssInput.Backgorund.IfcRoof.SelectedRoofEntityProperties = vrssInput
            .Main.IfcRoof.RoofEntities
            .Where(r => r.ID == selectedRoofEntityID)
            .Select(r => r.SummaryOfProperties).FirstOrDefault();
        vrssInput = setUpReferenceRoof(vrssInput);
    }
    this .Session["VrssInput" ] = vrssInput;
    this .TempData["CurrentView" ] = "Backgorund";
    return RedirectToAction("Input" );
    ...
}
...
```

During the construction phase of Prototyping Cycle 2, the author was able to build IFC support components for enabling the exchanging of data between the prototype VRSS DSS and BIM-authoring tools. To facilitate the coding and testing tasks, the author used ArchiCAD and Revit for creating IFC files that diversely described a variety of reference roofs. These IFC were used as a basis for constructing the software components that interact with IFC files. In addition to IFC support components, the author modified the existing components built during Prototyping Cycle 1 to provide end users with IFC file upload and download functions. The final outcome of the prototype construction conducted during Prototyping Cycle 2 was a working prototype DSS that has IFC support capabilities.

4.3.5. Prototyping Cycle 2: Delivery, Deployment, and Feedback

After the prototype DSS was constructed to be compatible with BIM-authoring tools, the author was able to arrange a couple of meetings with the CHPE faculty researchers to deliver the updated version of the prototype DSS and receive feedback on the prototype. The meetings allowed the prototype DSS to be evaluated by the CHPE researchers who played the client role in this prototyping project. This section provides the information about the meetings in detail.

To deliver the prototype DSS, the author deployed the DSS on the author's laptop computer using IIS7 Express and gave a quick demonstration of the IFC upload and download functions. For the demonstration, the author began with how IFC-based BIM models can be created in and exported as IFC files from ArchiCAD and Revit. The author then showed how to upload an exported IFC file using the prototype DSS web interface. Next, the author demonstrated how the information contained in the uploaded IFC can be used to initially determine the inputs required by the VRSS Framework. Afterward,

the author showed how IFC files that describe different green roof systems can be downloaded and saved onto a local computer. During the meeting, the CHPE researchers were free to explore the IFC support functions and were provided with technical assistance from the author.

In addition to the prototype demonstration, the author had opportunities to discuss some limitations and known issues concerning the IFC file reading and writing functions with the CHPE researchers. As mentioned in the previous section, the construction of IFC support components concentrated on exploring how the data about reference roof geometry and properties required by the VRSS Framework can be exchanged between the prototype DSS and BIM tools. In consequence, the code for IFC support components was not extendsively written for extracting data from and adding data into IFC files beyond the scope of reference roof geometry and properties. The limitations and known issues can be summarized below:

- The reference roof object described in the IFC file uploaded to the prototype DSS must be an instance of `IfcSlab` class defined by IFC2x3 TC1 specification.
- The geometry of the reference roof must be defined based on the `IfcArbitraryClosed-ProfileDef` or `IfcRectangleProfileDef` class defined by IFC2x3 TC1 specification.
- A particular IFC file provided by the prototype DSS contains only a low detailed green roof model that describes the roof properties used in the VRSS Framework and roof geometry extracted from the originally uploaded IFC file.

During the meetings, the CHPE researchers were able to provide valuable feedback on the IFC support capabilities of the prototype DSS. In general, the prototype DSS, as seen by the CHPE researchers, has fulfilled the high-level requirement for the prototype DSS to be compatible with major BIM-authoring tools. Although the prototype DSS has some limitations and known issues, its IFC support capabilities are sufficient enough for demonstrating how DSS can be used in conjunction with BIM-authoring tools. Furthermore, the limitations and known issues discussed above have a considerable potential to be resolved by writing additional code for retrieving more data from IFC files and for adding more detailed information to the IFC files automatically generated for end users to download.

In conclusion, Prototyping Cycle 2 focused on the BIM compatibility feature of the prototype DSS. To promote data interoperability, the author decided to use IFC physical files that comply with IFC2x3 TC1 specification for exchanging reference roof information between the prototype DSS and major BIM-authoring tools. At this point, the updated version of the prototype DSS is capable of working individually and working in conjunction with BIM tools to provide decision support information concerning vegetated roofing system selection for decision makers. Based on the information gathered from the meetings presented above, the prototype DSS can fulfill the requirement for Prototyping Cycle 2 and can continue to be developed further in Prototyping Cycle 3, which will be discussed shortly in the next sections.

4.4. Prototyping Cycle 3

4.4.1. Prototyping Cycle 3: Communication

The last prototyping cycle of the prototyping process focused on how the prototype DSS can be used for facilitating the collaboration between building design practitioners on vegetated roofing system selection. To examine the possibilities of the prototype DSS to be used as part of collaborative decision making, key pieces of information were obtained from the communications between the author and the research participants established in early 2012 and during the deployment phase of Prototyping Cycle 1. Said information is presented below.

Based on the research participants' opinions, the prototype DSS, as a web application, has a potential to be used for presenting decisions concerning the selection of vegetated roofing systems. The prototype DSS provides an online, interactive report that contains a CBA table, which presents information that leads to the final decision on which generic vegetated roofing system is the most appropriate for a particular project. The CBA table can be seen as the most valuable part of the report that can be used as a means for collaboration.

As seen by research participants, the prototype DSS could be more appealing to be used for communicating green roof selection decisions if it can generate reports for end users that can be shared with other project team members. The reports may be created in the form of Portable Document Format (PDF) documents, which are easy to view on different computer devices, to share with other practitioners, and to print out. In addition to PDF documents, one of the research participants recommended that it would be value-added if the prototype DSS could provide ready-to-use data representation elements such as bar charts that could be reformatted and inserted into users' developed reports or presentations. For convenient use, these elements could be contained in spreadsheets, slide presentations, or text documents for users to download.

The information above was very useful for the author to identify how the prototype DSS should be developed to facilitate the collaboration between design practitioners. The author decided to provide end users with the capability to download PDF reports from the prototype DSS. The PDF reports can be seen as a means to document users' decisions concerning vegetated roofing system selection and could, as well be used to communicate decision-making results between project team members. Nevertheless, the author decided to leave the capability to provide the data representation elements for a future iteration of the prototype DSS so as to narrow down the scope of this prototyping cycle.

The communications between the author and the research participants established in early 2012 and during the deployment phase of Prototyping Cycle 1 helped the author gain an in-depth understanding of the collaborative decision-making support feature of the prototype DSS. The information gathered from the communications lead to the main concentration of Prototyping Cycle 3 in which the prototype DSS should be able to provide PDF reports for end users to download. This information was used as a basis for prototype development throughout Prototyping Cycle 3.

4.4.2. Prototyping Cycle 3: Quick Planning

4.4.2.1. Prototyping Cycle 3: Scope

Based on the information obtained from the communications between the author and the research participants discussed in the previous section, Prototyping Cycle 3 focused on providing end users with a capability to download PDF reports generated by the prototype DSS. To fulfill this purpose, the use case diagram presented in Section 4.2.2.1 can be modified to depict the scope of Prototyping Cycle 3 as shown in Figure 4-64 below.

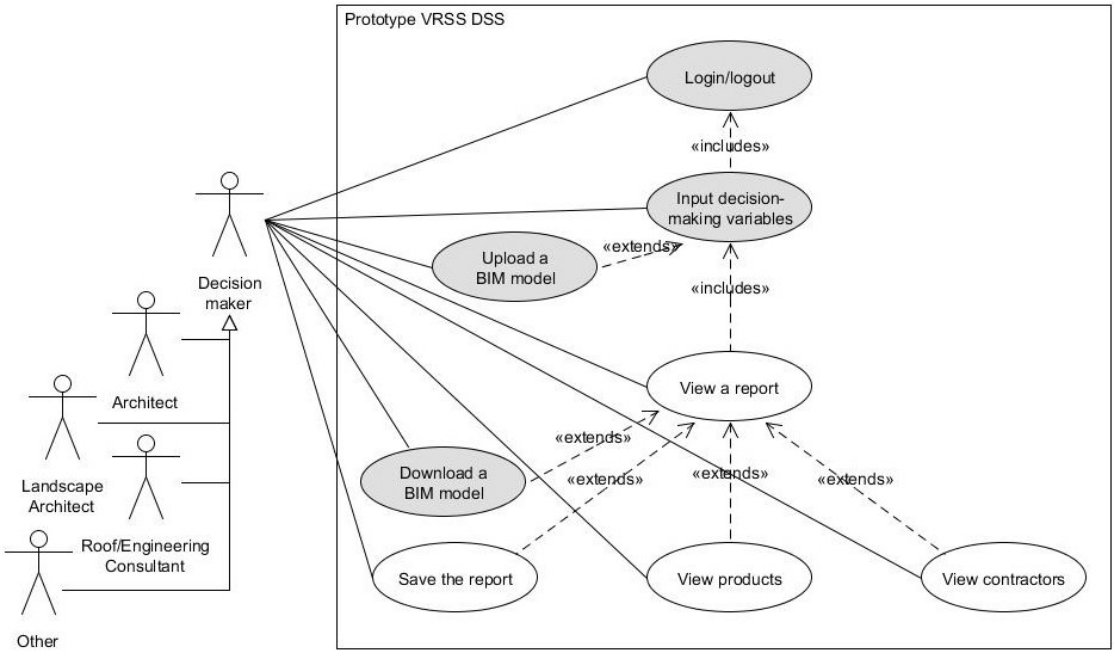


Figure 4-64: Scope of Prototyping Cycle 3

In Figure 4-64, the white colored use cases represent the scope of Prototyping Cycle 3. Figure 4-64 shows that building designers should be able to download a PDF report that contains decision support information presented on the `Report` page of the prototype DSS after it is generated. In this case, the `Report` page should be modified to provide a web control for downloading the PDF file.

In addition to the PDF file download use case, the author decided to include the use cases in which decision makers may request product and contractor information specific for a chosen green roof type into this prototyping cycle. It is important to note that the use cases seen by product providers and contractors were not taken into consideration in this study as discussed in Section 4.2.2.1. Therefore, these two use cases were included only to show some possibilities of additional services for decision makers.

The understanding gained from the information above helped the author narrow down the scope of Prototyping Cycle 3. Within the defined scope, Prototyping Cycle 3 primarily involved providing a capability to download PDF reports for end users and subsequently included making minor changes to

the prototype DSS user interface. The defined scope was used as a basis for prototype DSS development throughout this prototyping cycle.

4.4.2.2. Prototyping Cycle 3: Feasibility Study

Within the scope of Prototyping Cycle 3 discussed in the previous section, it is essential to examine the technology that allows the prototype DSS to create PDF files for end users to download. Therefore, as part of the planning phase, the author decided to conduct a brief study on the technology dimension of project feasibility, which is presented below.

To identify the technology used for creating PDF files, the author searched for information from both online and in NuGet, a package manager tool provided within the IDE. While the search results lead to a considerable number of PDF support libraries, the author decided to use iTextSharp as a C# programming library for creating PDF files in this prototype project. iTextSharp is a PDF library ported from the original Java library for PDF generation and manipulation, known as iText. iText is one of the most widely used free/open source PDF libraries. For iText, the author found a considerable amount of useful resources that seemed to facilitate the implementation of iTextSharp in a timely manner, especially the information provided on the iText website at http://itextpdf.com and in Lowagie's (2011) book, entitled iText in Action.

Based on a brief study on the technology dimension of project feasibility, the author decided to implement iTextSharp for generating PDF files, because of its capabilities and available supporting resources. The implementation of this free/open source library made the prototype DSS feasible to be developed further as a means to provide end users with PDF file download capability during Prototyping Cycle 3.

4.4.3. Prototyping Cycle 3: Modeling in the Form of Quick Design

As mentioned in Section 4.4.2.1, Prototyping Cycle 3 concentrated on providing building practitioners with the PDF download capability in order to facilitate practitioners' collaboration on vegetated roofing system selection. Fundamentally, the requirement for the PDF download capability can be seen as a functional requirement. Therefore, among the interaction, functional, content, navigation, and configuration models, the author chose to develop a functional model for examining the user-observable functions and the system operations related to the PDF download capability. The understanding gained from the developed model is presented in this section.

Figure 4-65 demonstrates an activity diagram that depicts the user-observable functions and system operations for the PDF file download capability. As shown in Figure 4-65, end users can access the PDF file download capability after they view the online, interactive report generated by the DSS. If the end users decide to download a PDF report, the DSS is responsible for generating a PDF report that contains the latest decision support information shown in the interactive report and then return the PDF report to the end users. The end users then can save the served file on their local computer.

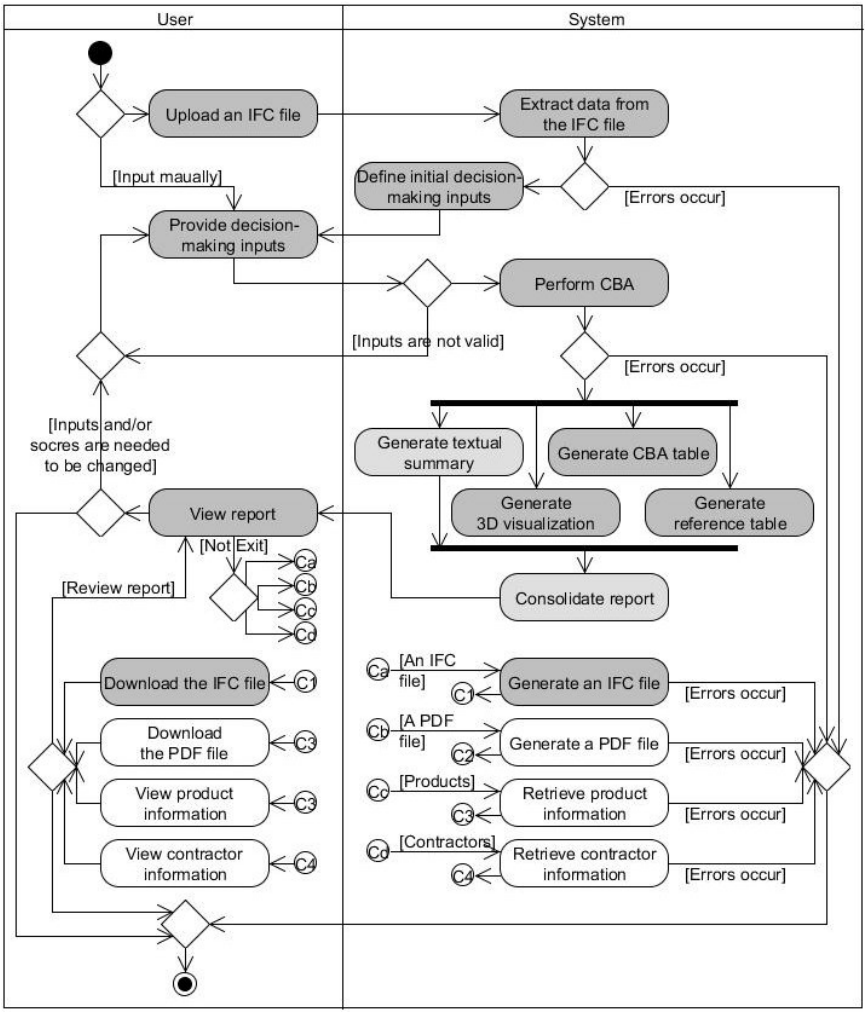


Figure 4-65: Functional model for Prototyping Cycle 3

In addition to the PDF download function, end users can subsequently choose to view product and contractor information as presented in Figure 4-65. For these functions, the system is responsible for returning web pages, which are placeholders for the content that may be developed in the future, as previously discussed in Section 4.4.2.1. From the web pages that provide product and contractor information, the end users can return to the interactive report to exit the application.

In the activity diagram shown in Figure 4-65, the white colored activities represent the user-observable functions and system operations mentioned above. To support these functions and operations, potential modifications will need to be made to the existing system. Thus, the light-gray colored activities represent the existing operations that should be taken into consideration during the construction phase of this prototyping cycle. In Figure 4-65, the dark-gray colored activities represent the functions and operations that are not affected by the PDF support functions and operations.

The understanding gained from requirements modeling above helped determine how the prototype DSS should be constructed in order to provide the PDF files downloading capability for end users. Because the

requirements for PDF support did not seem to affect the initial design of the prototype DSS, design modeling was not performed during Prototyping Cycle 3. It was assumed that the functional model presented in this section provides sufficient information for the construction phase of Prototyping Cycle 3, which will be extensively discussed in the next section.

4.4.4. Prototyping Cycle 3: Prototype Construction

To fulfill the requirements for Prototyping Cycle 3 discussed in the previous section, the extended construction of the prototype DSS involved three activities. The first activity was to write and test the code for the class or classes responsible for generating PDF reports. The second activity was to modify the DSS user interface to provide end users with a web control for downloading the PDF reports. The last activity was to develop example web pages that provide end users with product and contractor information. This section discusses how these three activities were performed in detail.

Because PDF reports seem to be specific to the prototype VRSS DSS, the author decided to create the classes that provide PDF support operations in the form of non-reusable C# classes. These classes were made available through the model directory of the user interface subsystem as shown in Figure 4-66. In Figure 4-66, `VrssPdfWriter` is the only class that was constructed to provide PDF support operations for creating PDF files, which can be accessed directly by `VrssDssController`. `VrssPdfWriter` creates a PDF file by adding document elements (e.g., title, paragraph and table) into a PDF document using operations served by iTextSharp, the PDF library discussed in Section 4.4.2.2. A partial code that represents this concept is provided in Box 4-15.

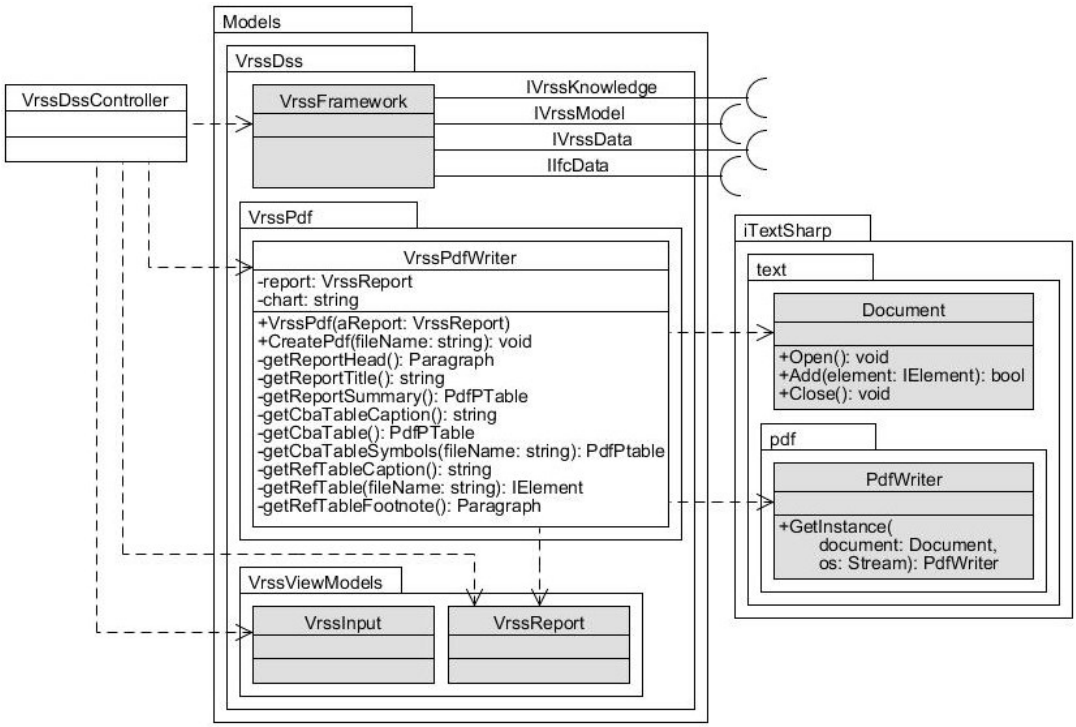


Figure 4-66: As built class diagram for PDF support classes

Box 4-15: Partial code for creating a PDF document

```
public class PdfReportWriter {
    ...
    public void CreatePdfFile(string fileName) {
        Document document = new Document(PageSize.LETTER, 36, 36, 54, 36);
        PdfWriter writer = PdfWriter.GetInstance(document, new
            FileStream(fileName, FileMode.Create));

        document.Open();
        document.Add(getReportHead());
        document.AddTitle(getReportTitle());
        document.Add(getReportSummary());
        ...
    }
    ...
}
```

Once the PDF support operations became available, it was essential to add a controller method for invoking the `VrssPdfWriter` functions. Box 4-16 presents a partial code written for the `DownloadPdf()` method of `VrssDssController`. The presented code shows that a PDF file is created in the form of a memory stream for a user to download using the action link provided on the `Report` page as presented in Figure 4-67. Afterward, when the generated PDF file is saved onto the user's computer, the file will be instantly removed from the memory. By implementing this technique, the generated PDF file will not be stored on the server computer. As well, every time the user makes changes on the `Report` page, a new PDF will be generated to provide the most recent decision support information for the user. An example of the PDF report can be found in Appendix D.

In addition to downloading a PDF file from the `Report` page, an end user may choose to access web pages that provide product and contractor information for a specific green roof type using additional action links shown in Figure 4.68. For instance, if the user decides to view production information, the user will be provided with the `Product` page developed as a placeholder for product information as shown in Figure 4-69.

Box 4-16: Partial code for `DownloadPdf()`

```
public class GreenRoofDssController : Controller {
    ...
    public virtual ActionResult DownloadPdf() {
        ...
        fileName = createPdfReport(report);
        pdfFile = Path.Combine(Server.MapPath("~/Files/Downloads"),
            fileName);

        var stream = new MemoryStream(System.IO.File.ReadAllBytes(pdfFile));
        System.IO.File.Delete(pdfFile);
        stream.Position = 0;
        return File(stream, contentType, downloadName);
        ...
    }
    ...
}
```

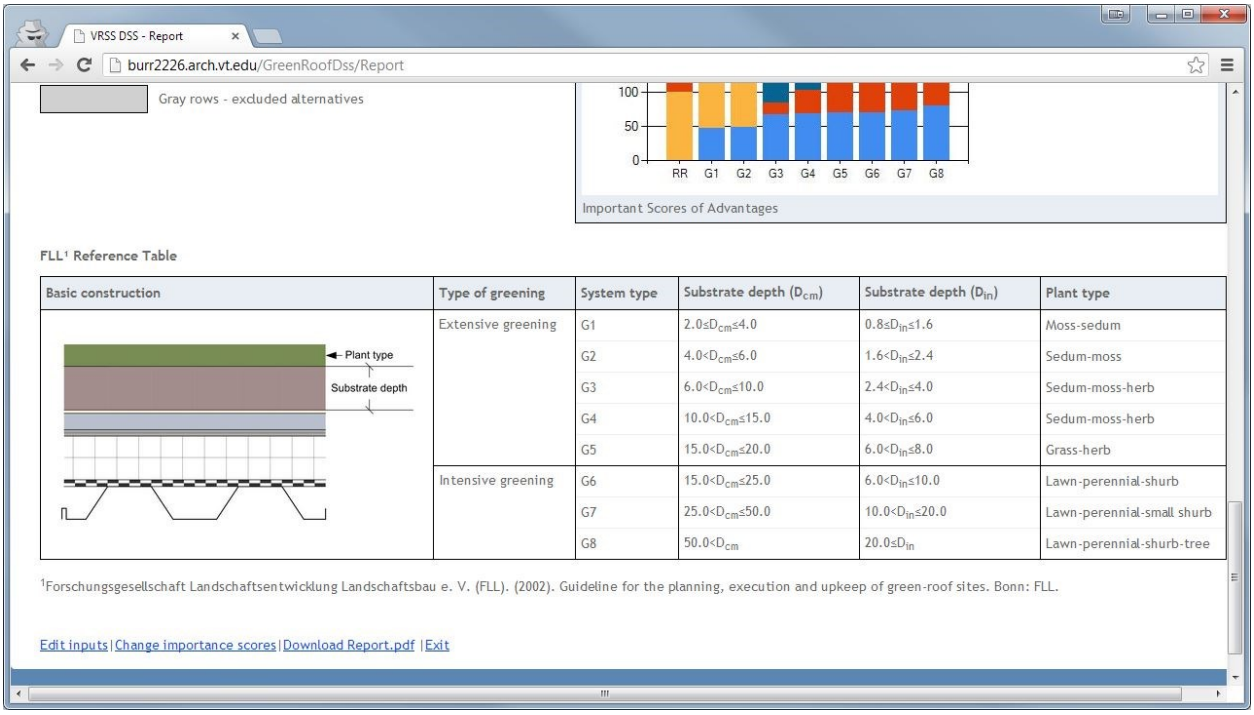


Figure 4-67: Action link for downloading a PDF file

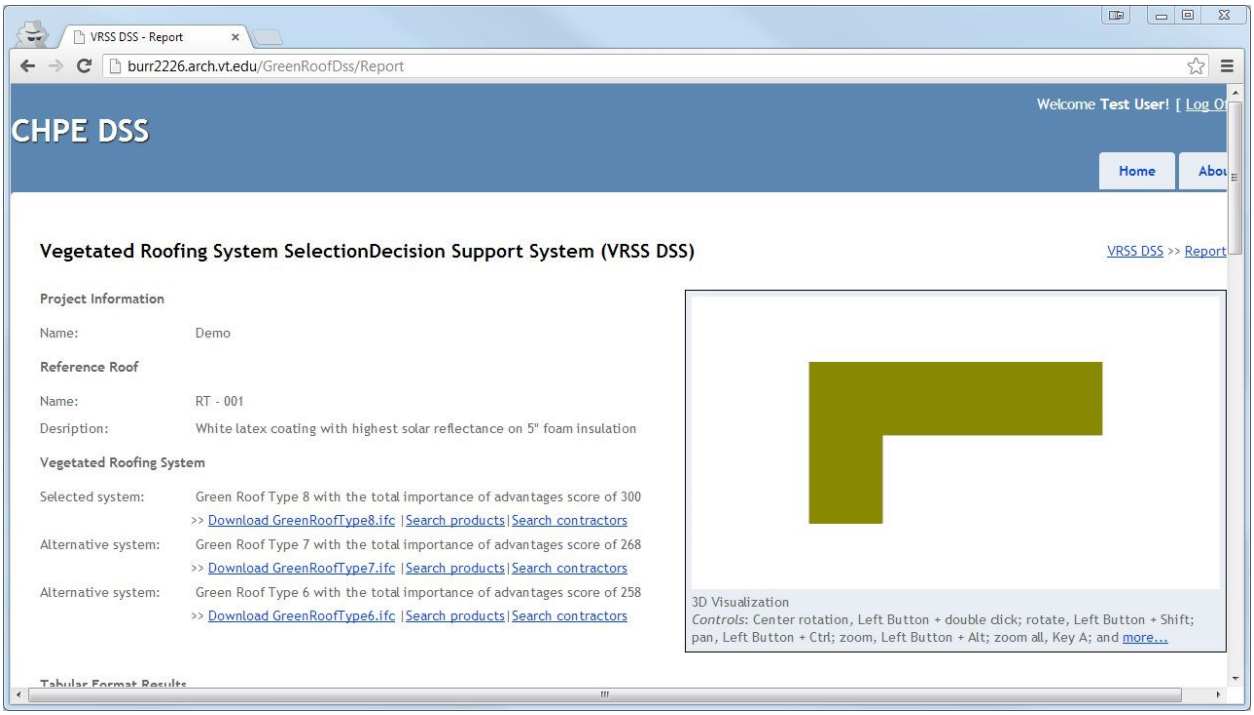


Figure 4-68: Action link for requesting the `Product` page

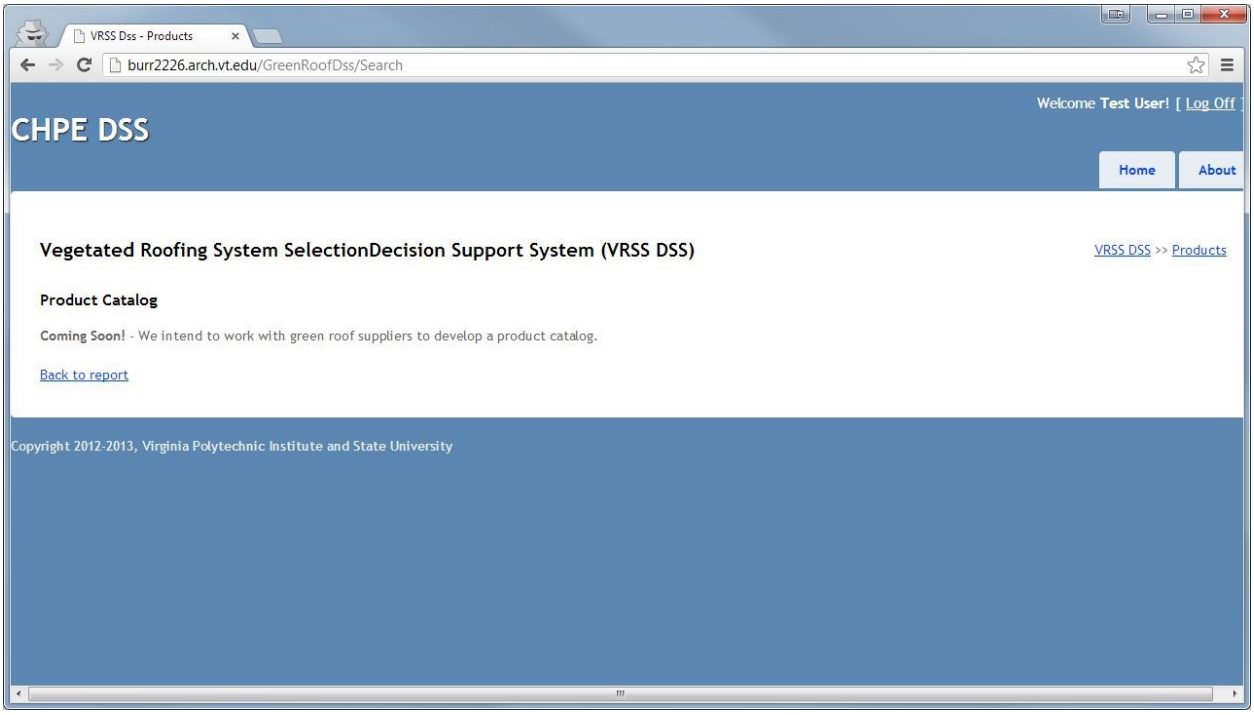


Figure 4-69: `Product` page

Throughout the construction phase of Prototyping Cycle 3, the author was able to create a class that is responsible for generating PDF files, add a web control that allows end users to download PDF files, and create example web pages for serving product and contractor information. The construction resulted in a working prototype DSS that can be used for facilitating design practitioners' collaboration on vegetated roofing system selection by providing PDF reports for viewing offline, sharing, and printing. This updated version of the prototype DSS also provides end users with web pages, which are the place holders for product and contractor information that may be available in the future.

4.4.5. Prototyping Cycle 3: Delivery, Deployment, and Feedback

Once the prototype DSS was developed to fulfill the three high-level requirements discussed in Section 4.2.1, it was essential for the system to be evaluated before being deployed. During the deployment phase of Prototyping Cycle 3, the author had opportunities to have the CHPE faculty researchers review the prototype DSS and give feedback on the system features and functions. Then, the prototype development was finalized, and the prototype DSS was deployed on the server computer. The information about the reviews and deployment is presented in this section.

To carry out the reviews, the author arranged meetings with the CHPE researches to present the updated version of the prototype DSS deployed on the author's computer using IIS7 Express. The demonstration not only concentrated on the PDF file download function, but also on the overall operation of the prototype DSS. The researchers were also encouraged to participate in a hands-on experience of the system and were provided with the author's technical assistance.

The feedback received during the meetings mentioned above indicates that the PDF reports provided by the prototype DSS could potentially be used for facilitating the collaboration between building project team members on vegetated roofing system selection. The reports are capable of documenting vegetated roofing system selection decisions and the reasoning behind the decisions. Such decision support information is presented in one page, which makes the reports easy to comprehend. By providing PDF file download capability, end users can save the PDF reports onto their local computer for offline viewing. As well, because PDF is a portable document format, the saved reports can be easily printed out and shared via emails or cloud storages.

Above all, the developed prototype DSS can appropriately fulfill the three high-level requirements discussed in Section 4.2.1, which makes the prototype DSS become an accurate representation of BIM-interoperable, web-based decision support systems. It is agreeable that the prototype DSS is reasonably valid for deployment on the server computer.

As part of the deployment phase of Prototyping Cycle 3, the author finalized the prototype development and deployed the prototype DSS on the server computer, which has been equipped with IIS7 and SQL Server 2008 Express R2 as discussed in in Section 4.2.2.2. The author deployed the prototype DSS by generating a deployment package and uploaded the package onto the server computer. On the server computer, the package was turned into a live web application using the Server Manager tool that came with the server operating system, Windows Server 2008 R2.

In conclusion, the deployment phase of Prototyping Cycle 3 provided opportunities for the prototype DSS to be evaluated by the CHPE faculty researchers. After it passed the evaluations, the prototype development was finalized, and the prototype DSS was deployed on the server computer. This version of the prototype DSS can be considered a valid research tool for examining the usefulness of BIM-interoperable, web-based, and knowledge-driven DSS, which will be extensively discussed in Chapter 5.

4.5. Prototype Development Results

The prototype development discussed throughout this chapter resulted in a prototype BIM-interoperable, web-based DSS for vegetated roofing system selection or VRSS DSS. The prototype DSS is capable of providing building project stakeholders with decision support information concerning vegetated roofing system selection. The prototype DSS is IFC2x3 TC1 compatible, which enables it to be used in conjunction with major BIM authoring tools such as ArchiCAD and Revit. The prototype DSS can also be used for facilitating collaborative decision making by providing PDF reports in which end users can save on their computer devices for offline viewing, print out, and share with the project team members. The prototype DSS was distributed through a demonstration website developed for serving multiple DSS for building practitioners, found at http://burr2226.arch.vt.edu. Because the website may be available online for a limited time, the screenshots of the prototype DSS and the demonstration website are provided in Appendix D. The presence of VRSS DSS suggests that a BIM-interoperable, web-based DSS can be

developed based on the current DSS and BIM theories and technologies to support building design practitioners' decisions. Based on its features and functions, VRSS DSS can be seen as a valid tool to be used as part of the qualitative study on the usefulness of BIM-interoperable, web-based DSS, which will be discussed further in the next chapter.

Chapter 5

5. Post-Use Interview Study

5.1. Interview Study Procedure

The second phase of this study aimed to examine the usefulness of the prototype DSS developed during the first phase of research as perceived by building design practitioners. To achieve this goal, a post-use interview study was conducted based on the qualitative research process mentioned in Chapter 3.1, which comprises four interrelated steps or phases: 1) data collection, 2) data reduction, 3) data display, and 4) conclusion drawing and/or verifying. These steps are fundamentally composed of a series of research activities as was extensively discussed in Chapter 3.3. In this section, the activities that were performed during the second phase of research are discussed in detail.

As mentioned in Chapter 3.3.1, the post-use interview study directly involved human subjects. Therefore, it was important to obtain approval for research protocol and supporting documents from the Virginia Tech Institutional Review Board (IRB) before the qualitative research process took place. To receive IRB approval, researchers must receive training concerning the protection of human subjects from the IRB. After completing the training, the approval request can then be submitted via the IRB Protocol Management system provided by the IRB. For this study, a set of proposed documents including research protocol, consent form, recruitment letter, and white paper were submitted via the IRB Protocol Management system in early 2012. After the submission, there was a response from an IRB officer asking for a few changes in the proposed documents. The changes included splitting the proposed consent form into two forms for use during the first and second phases of research, altering some wordings, and rewording in the consent form for the first phase of research that prototyping project meetings will be recorded. Based on the IRB officer's suggestions, the documents were edited and resubmitted to the IRB Protocol Management system. The edited documents were approved by the Virginia Tech IRB Administrator, Carmen T. Green, and were active until early 2013. In addition to the initial set of documents, an amendment was made to include Phase 2 interview questions and introductory questionnaire in early 2013. As well, because the post-use interview study had been conducted until 2013, a continuing review request for the research protocol and supporting documents were made in early 2013, and were approved by the Virginia Tech IRB Chair, Dr. David M. Moore. The IRB approved documents are provided in Appendix C.

During the time that the IRB approved documents were valid, the qualitative research process began with collecting data about the potential benefits of the prototype DSS from building practitioners, who were randomly recruited to represent the larger population of potential DSS users. To invite building practitioners to participate in the post-use interview study, the recruitment letter and white paper, found in Appendix C, were randomly sent out to almost one hundred professionals in the fields of Architecture,

Engineering, and Construction (AEC) in the form of in-person communication, tri-fold brochure, and e-mail. Among the significant number of invitation recipients, there were twelve AEC practitioners who volunteered to participate in the study. Information about the research participants is provided in Section 5.2.

As mentioned in Chapter 3.3.1, this study used in-depth interviews as a means to collect data about the usefulness of the prototype DSS from the research participants. The interviews were conducted in the form of face-to-face meetings at the locations convenient to the participants and in the form of online audio/video conferences. Prior to the interviews, the research participants were provided with the IRB approved consent form, the address of the demonstration website that hosts the prototype DSS, and the address of the online introductory questionnaire that summarizes the questions that would be asked during the interviews. Based on the research protocol, the participants were encouraged to review, sign, and return the consent form before the interviews. The participants were also encouraged to have a hands-on experience of the prototype DSS and to respond to the introductory questionnaire prior to the interviews. During the interviews, the participants were encouraged to answer the interview questions, which can be found in Appendix C. The participants were also provided with a software demonstration upon request and with brief answers to any questions they may have had concerning the study and the prototype DSS before the questions were asked. As well, the interviews were recorded with the participants' permission for the data retrieval purposes.

It is important to note that the interview questions provided in Appendix C were asked to help collect comprehensive data concerning the research participants and the usefulness of the prototype DSS. The questions related to the participants concentrated on the participants' background in vegetated roofing system selection and building information modeling. This set of questions was formulated to help the researcher understand the nature of the larger population of potential DSS users as discussed in Chapter 3.3.1.2. The questions related to the prototype DSS focused on the participants' perspectives on the values or benefits of the prototype DSS, in particular, and the fully developed version of the system that will serve multiple DSS in general. This set of questions were developed based on the operational benefits and potential cost threshold of the prototype development shown in Table 5-1. The benefits and cost threshold were established to help determine the effectiveness of the prototype DSS as suggested by Keen's (1980) value analysis approach discussed in Chapter 3.1.

After the interviews were conducted, the next phase of the qualitative research process encompassed transcribing interview records and reducing the data contained in the interview transcriptions into smaller, more manageable pieces. During this phase of the qualitative research process, the data was reduced into three categories. The first category represents the data about the research participants' background. The second category represents the data about the operational benefits of the prototype DSS. And the third category represents the data about the potential benefits and adoption of the full DSS application version that will provide end users with multiple DSS. In each category, the data was also reduced into

schemes or themes. For example, the schemes in the participants' background category included, but were not limited to, the participants' background in vegetated roofing system selection and BIM processes. In this chapter, the reduced data is provided in the form of comprehendible information in Section 5.2, 5.3, and 5.4.

Table 5-1: Matrix for evaluating the effectiveness of the prototype DSS

Benefits/cost threshold	Description
Decision support	The DSS provides useful decision support information for vegetated roofing system selection.
Compatibility	The DSS can use building data generated from BIM authoring tools used by the users.
Accessibility	The DSS can be accessed using different web browsers and/or internet accessible devices.
Time Savings	The DSS helps reduce time normally used to reach similar decisions
Decision documentation	The DSS helps document decision makers' decisions.
Communication	The DSS helps facilitate communications between project stakeholders.
Litigation	The DSS helps assert the legality of documenting design decisions.
Adoption	The maximum cost that the users would pay to gain the operational benefits is acceptable.

Once the large piece of data was reduced into smaller, more manageable pieces, the remaining steps of the qualitative research process involved displaying the data and drawing research conclusions. As mentioned in Chapter 3.3.3, the primary objective of displaying data is to facilitate a cross-case analysis to help understand the usefulness of the prototype DSS potentially perceived by the larger population of end users. The outcomes of the analysis were used as a basis for conclusion drawing. The more detailed information about the cross-case analysis and conclusion drawn from the collected data is provided in Section 5.5.

In conclusion, after the prototype DSS was developed and deployed on the server computer, a post-use interview study founded upon the qualitative research process was conducted to understand the usefulness of the DSS. The qualitative research adopted in the study fundamentally comprised a series of activities as presented in this section. Throughout this chapter, the outcomes of the qualitative research activities performed during the second phase of research will be further discussed in detail.

5.2. Interviewees' Background

As mentioned in Section 5.1, there were twelve building design practitioners who volunteered to participate in the post-use interview study. This group of participants or interviewees was assumed to be representative of a larger end user population. To understand the relationship between the interviewees and the end user population, it is essential to understand the interviewees' background in vegetated roofing system design and building information modeling. Therefore, the information about each interviewee background is provided below.

Interviewee 1

Interviewee 1 is a roofing consultant who has a background in engineering. The interviewee is extremely familiar with vegetated roofing system or green roof design. The interviewee's experiences comprise conducting research in Austin, Texas, on vegetated roofing systems and providing consultation on roofing systems including vegetated roofing systems. The interviewee completed five vegetated roofing projects.

The interviewee's responsibility in the projects was to provide consultation to design teams lead by architects for new building projects. The interviewee has not used any specific tools that help provide decision support information for vegetated roofing system design. The tools that the interviewee has employed for supporting vegetated roofing system design decision include two dimensional or 2D construction drawings and industry information.

In addition to vegetated roofing system design, Interviewee 1 is moderately familiar with Building Information Modeling (BIM). The interviewee has purchased a BIM authoring tool for five years and has occasionally attended seminars on BIM topics. The interviewee has been involved in twenty five building projects that employed BIM processes. The interviewee has regularly used 2D drawings generated from BIM models, but has not used BIM tools directly. Based on the interviewee's experience of BIM, the interviewee has seen that there is a growing demand on BIM deliverables imposed by building owners. The interviewee addressed during the interview that the building owners, both in private and public sectors, have progressively used BIM tools for Facilities Management (FM), especially for building maintenance and long term performance management. To collaborate with building owners and design teams, the interviewee has been using Revit as a primary tool for navigating through BIM models and printing out 2D drawings.

Interviewee 2

Interviewee 2 is a roofing consultant who has a background in engineering. The interviewee is extremely familiar with green roof design. The interviewee has been involved in a number of green roof projects, especially in the Washington, District of Columbia (DC), area. The interviewee's experiences include designing and specifying green roofs for existing buildings. The interviewee has designed and specified green roofs for more than twelve existing buildings, six of them located in DC. The interviewee's responsibilities include, but are not limited to, evaluating existing buildings, making recommendations for roofing systems, preparing the scope of work for bidding by competent contractors, managing the bidding process, assisting in the selection of the bidder, aiding in the awarding of the contract, and providing support for construction quality assurance. The interviewee typically uses the experiences of team members and industry information (e.g. information from roofing professional trade shows and manufacturers) for designing and specifying green roofs.

Interviewee 2 is not at all familiar with BIM technology. The interviewee has heard of BIM, but has never used BIM tools. According to the interviewee, BIM tools are not appealing to the interviewee, because the interviewee normally works on existing buildings in which BIM models are not available and the nature of the interviewee's work has been relying more on performing calculations rather than looking up data. Nevertheless, the interviewee believes that BIM could be more useful for retrofitting projects in the future due to the growing adoption rate of the technology in new building projects.

Interviewee 3

Interviewee 3 is a roofing system manufacturer representative who is extremely familiar with green roof design. The interviewee has been involved in designing green roof systems and providing instructional seminars on green roof systems for the design community. The interviewee has participated in more than five green roof projects whereby the interviewee's responsibility was to evaluate the design of green roof systems. The interviewee typically applies the experiences of design team members to the design of green roof systems. According to the interviewee, the members of the team may include, but are not limited to, architects, landscape architects, and structural engineers.

Interviewee 3 is slightly familiar with building information modeling. The interviewee has had basic knowledge of BIM processes, but has never used any BIM tool.

Interviewee 4

Interviewee 4 is a junior-level architect who also serves as a LEED AP professional for the interviewee's design team. The interviewee is very familiar with vegetated roofing system design. The interviewee's experiences include designing green roofs and proposing several green roofs on a variety of building projects. As an architect, the interviewee had an opportunity to work on one large-scale green roof project. The interviewee's responsibilities normally include laying down the waterproofing and runoff conditions and working on the construction details of green roof systems. As a LEED AP professional, the interviewee's responsibility usually involves proposing the implementation of green roof systems in order to obtain LEED credits. In general the interviewee uses the verbal collaboration between multidisciplinary project team members as a means to help select green roof systems. To choose a green roof, the interviewee typically collects decision-making variables from the team members, and then performs cost-benefits analysis.

As an architect, Interviewee 4 is extremely familiar with BIM-based software. The interviewee has been using BIM-based tools for almost ten years and has been involved in more than 130 BIM projects. The interviewee's responsibilities include, but are not limited to, compiling construction documents, facilitating interdisciplinary coordination, and performing design analyses (e.g. daylighting and energy analyses). The BIM tools that the interviewee has been using in the projects include Revit, MicroStation, Rhino, Green Building Studio, Vasari, and Ecotect. By the time of this study, the interviewee was about to start using Integrated Environmental Solutions or IES as a tool for analyzing building performances.

Interviewee 5

Interviewee 5 is a junior-level architect who is moderately familiar with green roof design. The interviewee's experience includes proposing green roofs for two building projects. In those two projects, the interviewee was responsible for preparing the construction documents that involved green roof system construction details. The interviewee normally uses information provided by green roof providers as a basis for green roof design.

Interviewee 5 is extremely familiar with BIM technology and has been using BIM-based tools for six years. The interviewee has involved in more than five BIM projects. The interviewee responsibilities include developing mass models, performing design analyses, and compiling construction documents. The BIM-based tools that the interviewee has been using include Revit, Green Building Studio, and Ecotect.

Interviewee 6

Interviewee 6 is an architect who serves as a project manager. The interviewee is very familiar with vegetated roofing system design and has been involved in at least three building projects that implement vegetated roofing systems. In such projects, the interviewee was responsible for conducting research, facilitating structural system coordination, working out construction details, and specifying vegetated roofing systems. The interviewee typically requests information about vegetated roofing systems from manufacturer representatives after deciding that they will be used on the building. To receive appropriate details and specifications, the interviewee usually provides manufacturer representatives with project conditions such as employed structural systems and expected LEED goals. Most of the time, the interviewee has received proprietary product details and specifications from manufacturer representatives. Therefore, the interviewee has to generalize the received product details and specifications before including them in construction documents. Based on the interviewee's experiences, the vegetated roofing systems that the interviewee has specified are quite similar across building projects in terms of plant type, growing medium, and construction technique.

Interviewee 6 is moderately familiar with BIM processes and has been using BIM-based tools for about two years. During the time of this study, the interviewee has worked on a BIM project in which the interviewee used a BIM authoring tool on a regular basis for the first time. The interviewee became familiar with BIM processes, because this project was very large and complex. In the project, the interviewee was responsible for managing the project team, hand sketching construction details for team members, and modeling some construction details using a BIM authoring tool. The interviewee has used Revit as a primary BIM tool since the interviewee began to adopt BIM processes.

Interviewee 7

Interviewee 7 is a roofing researcher who has background in architectural technology. The interviewee is moderately familiar with green roofs and understands the fundamental principles of green roof design. The interviewee's experiences comprise conducting research and publishing research papers related to green roofs and designing green roof systems. The interviewee has been involved in the design of green roofs for more than ten projects. In those projects, the interviewee was responsible for conducting research on green roofs and providing consultation for the project design team. For research, the interviewee has a special interest in hybrid or integrated green roofs, which represent the integration between green roofs and ballasted roofs. For green roof design, the decision support tools that the interviewee normally uses include the Cool Roof Calculator and EnergyPlus.

Interviewee 7 is not at all familiar with BIM technology and has never used any BIM-based tool.

Interviewee 8

Interviewee 8 is a director of sustainability who has a background in architectural/structural engineering. The interviewee is extremely familiar with vegetated roofing system design. The interviewee's experiences include exploring the options of vegetated roofing systems and working on the installations and the details of vegetated roofing systems. The interviewee has been involved in four vegetated roofing system projects including three completed projects and one under construction. The interviewee's responsibilities in the projects include conducting research, assisting with detailing the installation, performing cost analysis, and collaborating with vegetated roofing system installers. The interviewee normally uses information from installers and project team members' experiences for supporting vegetated roofing system design decisions.

Interviewee 8 is slightly familiar with BIM-based tools and has never used any BIM tool in particular. While the interviewee has never worked on a BIM model, the interviewee works with colleagues who have heavily used BIM tools in their projects.

Interviewee 9

Interviewee 9 is a senior-level architect and a former member of the developer team that developed Revit. The interviewee is moderately familiar with vegetated roofing system design. The interviewee's experiences include proposing green roofs for LEED building projects. The interviewee has been involved in two building projects that have vegetated roofing systems in Washington, DC, area. While aesthetic and environmental benefits of vegetated roofing systems are highly recognizable, the interviewee sees that LEED has become the most important driver of vegetated roofing system adoption for buildings in the DC area. In general, the interviewee's responsibilities include creating reusable BIM components that represent generic construction details of vegetated roofing systems. The interviewee uses information provided by product vendors for supporting vegetated roofing system design decisions. Furthermore, the interviewee mentioned during the interview that decisions related to green roof design are unlikely to be made by one person. The decisions would probably be made by an intermediate- or a senior-level architect together with a landscape architect and other consultants such as structural engineer, mechanical engineer, and sustainability specialist.

As a former Revit developer, the interviewee is extremely familiar with BIM processes and BIM-based tools. The interviewee has been involved in more than one hundred BIM projects. The interviewee's responsibilities comprise participating in the projects as a project designer, doing technical detailing work, automating modeling processes for design teams, developing sophisticated spreadsheets for LEED analyses, and making measurement tools within Revit that use schedules to retrieve data for building compliance. The interviewee most of the time uses Revit and occasionally uses Navisworks as part of BIM processes.

Interviewee 10

Interviewee 10 is a director of sustainability who has a background in architecture. The interviewee is very familiar with green roof design. The interviewee's experiences include designing green roofs and doing technical detailing work. The interviewee has been deeply involved in two building projects that have green roofs and has proposed green roofs for more than eighty building projects. In the two projects that the interviewee was deeply involved with, the interviewee was responsible for laying out the green roofs, working on the construction details of the green roofs, and following through the construction and submittal processes. In other projects, the interviewee has been involved in a more advisory role. The tools that the interviewee uses for designing green roofs include different calculators such as wind and structural calculators and the experiences of the interviewee's colleagues.

In addition to the interviewee's experiences in green roof design, Interviewee 10 is very familiar with building information modeling. In the time that the interviewee served as a project architect, the interviewee's experiences included creating reusable BIM components and building BIM models. After the interviewee became a director of sustainability, the interviewee began using BIM tools for navigating through and extracting data from BIM models rather than building BIM models. The interviewee has been involved in more than twenty BIM projects. The interviewee primarily used Revit as a BIM-authoring tool.

Interviewee 11

Interviewee 11 is a mechanical engineer who is slightly familiar with green roof design. The nature of the interviewee's experience comprises participating in the conceptual design of green roofs. The interviewee has not been involved in any constructed green roof project. As a mechanical engineer, the interviewee' responsibilities include evaluating the thermal impact of green roofs on the overall building envelope and designing the insulation layer of green roofs. The interviewee has never used any tool for supporting green roof design decisions.

Interviewee 11 is moderately familiar with BIM technology. The interviewee's experiences include navigating through BIM models, creating section drawings, and extracting data from BIM models. The interviewee has been involved in more than ten BIM projects as a mechanical engineer of record. The interviewee primarily uses Trane TRACE as a mechanical load calculation tool and uses Revit as a BIM authoring tool. Trane TRACE is compatible with Revit as it supports Green Building XML (gbXML) files that can be exported from Revit.

Interviewee 12

Interviewee 12 is an architect who serves as a project manager. The interviewee is moderately familiar with vegetated roofing system design. The interviewee's experiences include attending educational seminars and having conversations with vegetated roofing system professionals. The interviewee has not been involved in any vegetated roofing system projects. However, if the interviewee has to design vegetated roofing systems, the interviewee would probably request information from vegetated roofing

system suppliers and conduct research on the products emphasizing the implication to the employed structural systems and the costs of vegetated roofing system implementation. The interviewee also stated that the interviewee would be the person who is responsible for choosing green roof systems for building projects.

Interviewee 12 has been trained in using a BIM authoring tool and is moderately familiar with BIM. The interviewee has been involved in at least three BIM projects. The interviewee's responsibilities include managing the projects, overseeing BIM processes, using a BIM authoring tool for viewing and editing BIM models created by other team members, exporting 2D CAD drawings from BIM models, and printing out 2D and 3D views. The interviewee mainly uses Revit as a BIM authoring tool.

The Variation of Interviewees' Background

As mentioned in Chapter 3.3.1.2, this study adopts the maximum variation sampling technique to help select research participants or interviewees, as this technique provides the widest possibility of selecting a sample group that represents a larger user population. The maximum variation sampling technique involves defining the maximum range of interviewee's settings or contexts, and then selecting research participants from the widest range of possible interviewees in the defined settings or contexts. Guided by the initial range of interviewee's contexts discussed in Chapter 3.3.1.2, the variation of the interviewees' background shown by the interviewees' familiarities with vegetated roofing system design and building information modeling is presented in Figure 5-1 below.

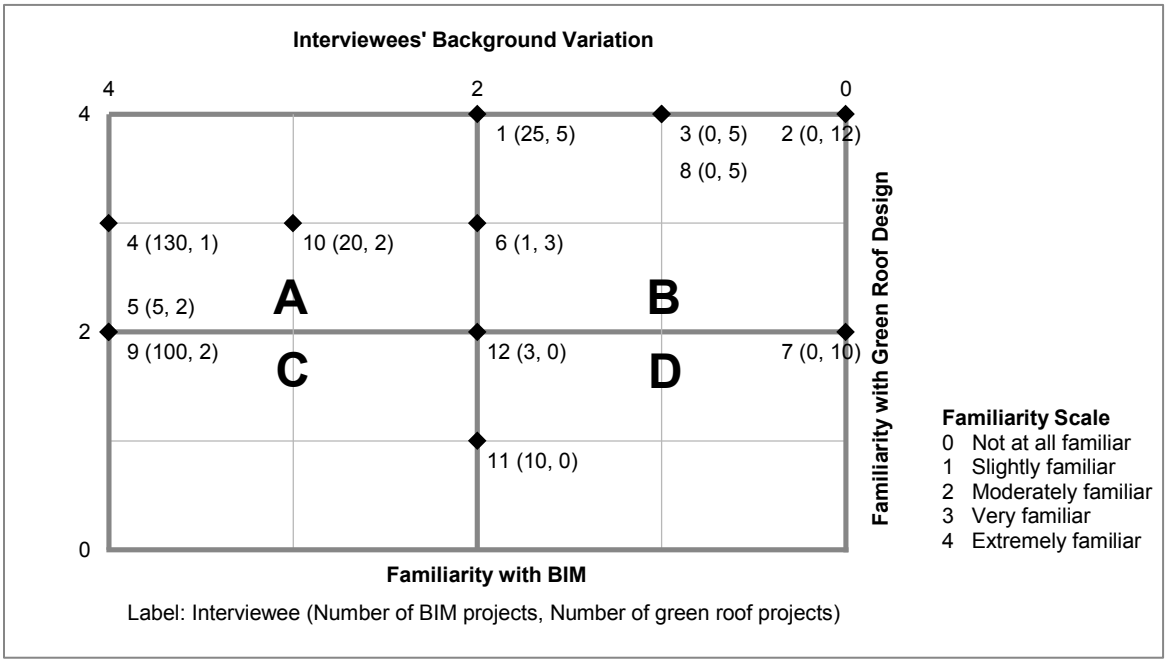


Figure 5-1: Interviewees' background variation

Figure 5-1 demonstrates four different contexts of potential end users of the fully developed version of the DSS. In Figure 5-1, Region A represents the end users whose familiarities with green roof design and

BIM range from "moderately familiar" to "extremely familiar". Region B represents the end users whose familiarity with green roof design ranges from "moderately familiar" to "extremely familiar" and whose familiarity with BIM ranges from "not familiar at all" to "less than moderately familiar". Region C represents the end users whose familiarity with green roof design ranges from "not familiar at all" to "less than moderately familiar" and whose familiarity with BIM ranges from "moderately familiar" to "extremely familiar". Region D represents the end users whose familiarities with green roof design and BIM range from "not familiar at all" to "less than moderately familiar".

As shown in Figure 5-1, from twelve interviewees, Region A comprises seven interviewees including five architects, one roof consultant, and one sustainability specialist. Region B encompasses four interviewees including two roofing consultants, one roofing researcher, and one sustainability specialist. Region C includes one mechanical engineer. Figure 5-1 also shows that the interviewees who participated in this study represent the population of potential DSS end users whose background in green roof design and building information modeling falls within Region A, B, and C. Although the invitations were randomly sent to almost one hundred AEC practitioners regardless of their background, the practitioners who are not familiar with green roof design and BIM were not interested in participating in this stage of research. Therefore, it is important to note that the conclusion drawn based on the data collected from the interviewees may not be applicable to the population of end users whose background in green roof design and BIM fall within Region D shown in Figure 5-1.

In summary, this section provides comprehensive information derived from the data about the interviewees' background in vegetated roofing design and building information modeling collected through a series of in-depth interviews. The information about the interviewees' background indicates that the interviewees involved in this study are diverse enough to represent a large portion of potential end user population. For this study, the information about interviewees' background will be used as a basis for interpreting the data about the usefulness of the prototype DSS and the fully developed system in the future, which will be extensively discussed in the following sections.

5.3. Interviewees' Feedback on the Prototype DSS

5.3.1. Decision Support Tool for Vegetated Roofing System Selection

In general, almost all of the interviewees agree that the prototype VRSS DSS is a useful decision support tool and could potentially be a good educational tool, especially for the architects or other designers who are responsible for making high-level decisions involved in the implementation of vegetated roofing systems. Frequently, the majority of the interviewees often have to deal with unstructured decisions when deciding whether or not a green roof should be used and/or trying to choose an appropriate green roof for a particular building project. For this aspect, the prototype DSS becomes useful because it provides a framework, or a more structured approach, which helps simplify the decision-making processes currently employed by the interviewees. The prototype DSS also provides basic knowledge about green roofs including, but not limited to, the information about eight generic green roof types and the important

decision-making variables and factors involved in green roof selection. With the understanding of basic knowledge, the necessary decision-making variables can be easier and more effective for identifying and collecting from a variety of sources such as experienced project team members and product vendors. Furthermore, the decision-making factors are well-grounded on intangible benefits potentially provided by green roofs, which make it easier for the interviewees to convince other project team members to consider using green roofs on buildings.

Beyond the generalized information above, it is beneficial to learn about additional benefits of the prototype DSS as seen by some individual interviewees. As a roofing consultant, Interviewee 1 mentions that the prototype DSS may be useful for facilitating communications between project team members. Interviewee 1 recognizes that the prototype DSS would provide opportunities for the interviewee to participate in the early phases of the project to assist architects or building owners with the initial inputs for the prototype DSS rather than only being involved in the later phases of the project as usual. In addition to Interviewee 1, Interviewee 9 thinks that the prototype may be useful for knowledge distribution, which could help design practitioners increase their capabilities. The interviewee's statement is provided below:

> Anytime that you can sort of take quite a lot of knowledge that is hard to get into one place and you make it easy for somebody who doesn't actually have that knowledge to actually employ, you know, it's a fantastic thing, because, what you are really doing, you are making our employees sort of more capable and useful by giving them this kind of tools.

While the prototype DSS seems to be a useful decision support tool for green roof selection, there are some concerns raised by some interviewees who are not quite sure about the benefits of the prototype DSS. One concern expressed by Interviewee 6 is that the prototype DSS might not be useful for the interviewee because the interviewee might be involved in building projects that have green roofs only once every one or two years. For such projects, green roofs are usually included as part of the project requirements at the beginning of the project often to receive LEED credits. In this kind of situation, the interviewee typically looks for highly detailed information about green roofs rather than generic information provided by the prototype DSS. Detailed information can be easily obtained from product suppliers. In this case, Interviewee 6 intends to use the prototype DSS between the late design development and early construction documentation phases of building design services discussed in Chapter 1.1.1, whereas other interviewees tend to use the DSS between the early schematic design and early design development phases. Based on the suggestion provided by Interviewee 6, the prototype DSS would be useful when it can serve decision makers' specific needs at particular phases of building design services. This understanding can be supported by the response to the question about the usefulness of the prototype DSS as a tool for vegetated roofing system selection provided by Interviewee 3, who is a manufacturer representative, below:

I am not sure. You are probably asking the wrong person. I would see the market for this system would be the design professionals.

Based on the provided information, the prototype DSS tends to be a useful decision support tool for vegetated roofing system selection as seen by almost all interviewees, and at various degrees. The prototype DSS seems to be useful for architects and other designers who moderately make and are directly responsible for making decisions about the implementation of vegetated roofing systems. Furthermore, the prototype DSS may be more useful during the early stages of building design services than in the later stages due to the high-level decision support information it provides.

5.3.2. Decision Support Information for Vegetated Roofing System Selection

Most of the interviewees share a similar perspective that the prototype DSS provides useful introductory decision support information for vegetated roofing system selection. In general, there are two major pieces of information that are useful for the interviewees: the basic information about decision-making inputs and the decision support information provided on the report. For the decision-making inputs, the prototype DSS provides useful information about decision-making factors, which helps the interviewees understand the potential tangible benefits of green roofs, one example being the benefit for storm water flow reduction. The interviewees also find the information provided in a downloadable user manual useful for understanding how the decision-making inputs are used for determining the attributes of the proposed conventional roof and green roofs. For the report, the interviewees find that the prototype DSS provides sufficient information, which seems to help them make sound decisions on green roof selection based on eight generic green roof types. The interviewees see that the information on the report is clearly provided in a variety of forms, including a brief textual description, a decision-making table, a comparison chart, a 3D visualization, and a 2D section drawing, which makes the information easy to comprehend. Furthermore, the information on the report seems to be useful for setting requirements for the detailed-level design.

According to some interviewees, there are some additional pieces of information that could potentially be provided through the prototype DSS. As stated by Interviewees 3, 7, and 8, the prototype DSS should also provide some horticultural information such as the information about engineered soil and vegetation, which could potentially be useful for the reconsideration phase of the decision-making process. The horticultural information, typically provided by horticulturists or product vendors, often has to be gathered from various sources; it would be helpful if this kind of information could be looked up in one place. In addition, Interviewees 1 and 8 suggested that there should be some kind of support information that helps end users provide appropriate inputs to the system. For example, the current version of the prototype DSS only provides one possible input value for the coefficient of discharge of the reference roof, which is based on the water retention capability of the non-vegetated gravel area of green roofs. For this input, the interviewees suggest that more possible coefficient of discharge values for different kind of conventional

roofs should be identified and provided to help end users relate the input value to the actual reference roofs in their projects. One of the thoughts shared by Interviewee 8 is provided below:

> I think when I was using the tool I felt like I had to read the user manual very heavily, and I was still kind of left with some questions about it. So if I just have a regular flat black top roof, what would that be? Some kind of, like there needs to be some of the information for comparison, because we don't necessarily all use the same sort of coefficient, or you know what, those measurements are not necessarily the same in our industry.

In general, most of the interviewees are satisfied with the fundamental-level decision support information provided by the prototype DSS. The understanding gained from the collected data suggests that the interviewees expect not only the decision support information from the prototype DSS, but also the support information for assisting them with the decision-making inputs and during the reconsideration phase of the decision-making process. Based on the interviewees' suggestions, further studies on the support information should be taken into consideration in order to improve the User Experience (UX) of the system.

5.3.3. Building Information Modeling (BIM) Compatibility

Almost all interviewees who directly and indirectly use BIM tools see that the BIM compatibility feature of the prototype DSS is beneficial to their work. All interviewees who are BIM users agree that the BIM compatibility feature is one of the most useful features of the tool. In addition to the BIM user group, interviewees who have been involved in the more supporting roles of building projects such as engineering, roofing, or sustainability consultants, consider this feature useful even though they may not use BIM tools directly. This group of interviewees sees that the BIM compatibility feature is useful for collaborating with their team members who regularly use BIM. The interviewees who have never used BIM tools nor participated in any project that utilizes BIM are not sure or do not think that the BIM compatibility feature is useful for their work.

Most of the interviewees share a similar perspective that, by having the BIM-compatible capability, the prototype DSS provides effective ways to utilize building data in BIM models for decision support. Fundamentally, the prototype DSS is useful because it can automatically take data already in existence in BIM models and use the taken data to support decision making, which helps reduce end users' efforts to input the data into the system manually. The prototype DSS can also provide end users with BIM models that can be imported back to their projects. Furthermore, the prototype DSS can be seen as a platform that allows decision makers who are not savvy BIM software users to gain benefits from the data contained in BIM models in support of decision making. For this particular benefit, even the interviewees who are extremely familiar with BIM software also see that working with the data separately seems to be an appropriate approach. For example, a statement given by Interviewee 4 is provided below:

> What I like about it is that it's obviously information-based. So you export a schedule of values where most interaction is exporting between the different modeling programs. Typically, it's a lot of cleaning up, when you just go off the visual side of it, it has never reproduced exactly how you want. So this way it's really nice, because it's just a database and there is very much a seamless integration between Revit, it seems like, and DSS, which is just perfect, the less cleanup the better.

Among the essential benefits of the BIM compatibility feature discussed above, many interviewees think that the ability to provide end users with BIM models for downloading and importing back to their projects is very important and should be leveraged in the future versions of the DSS. As mentioned by the interviewees, the system should be able to provide end users with generic detailed-level green roof models that can be further modified inside BIM authoring tools. For instance, a suggestion given by Interviewee 9 is provided below:

> In the long run, what I would want is to be able to have a new Revit roof type, kind of appearing on the model, that I can then go to change all the roofs that I already created by that stage and convert them to the type so they will have the right layers in them, right materials, and probably key details that you want to accompany that particular roof type. How does it meet that kind of parapet? How do you put paving in it? You know how it interacts with irrigation system something like that.

In addition to the improvement in the quality of BIM models provided by the prototype DSS, the effectiveness of the methods for establishing the communication between the prototype DSS and BIM authoring tools should be reexamined in the future versions of the DSS. When Interviewee 6 tried to add custom IFC parameters into an employed BIM authoring tool, the interviewee found that the recommended setup method seemed to take quite a few steps and could be error prone. Therefore, Interviewee 6 suggests that the prototype DSS and other DSS that may be developed in the future should be provided in the form of a plug-in suite, which can be integrated into BIM authoring tools. The interviewee statement is provided below:

> I think, if this type of tool is part of a suite of other tools; for example, for the decision support tools, for a couple of dozen more and more components of typical buildings, that allow the input of known criteria and the determining of the unknown criteria to import families and something like that, I can see it being useful if it is integrated within Revit.

Based on the information provided above, the BIM support feature of the prototype DSS seems to be useful for the interviewees. Although further improvements are required, the interviewees' shared experiences indicate that this feature helps reduce the time for data input and provides end users with pre-built BIM models for green roofs that can be imported into their BIM projects. The interviewees' experiences also suggest that, while the DSS should be provided as an independent platform, plug-ins should be developed for facilitating the exchange of data between the DSS and BIM authoring tools.

5.3.4. System Accessibility

As a web-based application, almost all interviewees agree that the prototype DSS is highly accessible and easy to maintain. For accessibility, the prototype DSS can be accessible from most widely-used web browsers such as Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, and Apple Safari. As a result, the prototype DSS can be used on multiple computer operating systems (e.g., Microsoft Windows, Apple Macintosh, and Linux) and Internet-accessible mobile devices (e.g., smartphones, tablet computers, and laptop computers). For maintenance, the prototype DSS does not have to be installed on end users' computers, which helps reduce the efforts in and costs of software installation and maintenance. By using the prototype DSS, end users will always be provided with the latest version of the application.

In addition to the essential benefits above, there are some interviewees' suggestions related to how they might access the prototype DSS that should be taken into consideration. As suggested by the interviewees, the prototype DSS tends to be used on desktop and laptop computers rather than on smartphones or tablet computers. In their work environments, the interviewees usually use desktop and laptop computers for assisting in their decision making activities. The interviewees also prefer to use the prototype DSS on desktop and laptop computers basically because they can easily use a variety of functions provided by the DSS, particularly for printing reports. Furthermore, desktop and laptop computers are typically the machines on which BIM applications are installed. To support this finding, the suggestion provided by Interviewee 7 is presented below:

> Yes, it's really useful. I think that first of all, having the tool available; a web-based tool is important. I would say that by having a basic web platform that is accessible through a web browser is important. There are other opportunities with other systems like smartphones or tablet Apps and things like that, but I would say that, because of the value of the printed report, I would say that the most important method for accessing the tool will be through a conventional PC, utilizing a web browser.

Although the interviewees tend to use the prototype DSS on desktop and laptop computers, it does not mean that the mobility of the prototype DSS is not useful. Most of the interviewees suggest that it is important for the prototype DSS to be accessible from web browsers running on different operating systems. The interviewees mention that there are more than one operating systems being used regularly in their work environments. The interviewees may also access the prototype DSS from different places such as from offices and building sites. As mentioned by some of the interviewees, mobile devices, especially large-screen tablet computers, seem to have a considerable potential to be used for viewing decision-making results and presenting the results to other project team members. To support this finding, the thought shared by Interviewee 4 is provided below:

> Yes, obviously the workflow is being able to have a flexibility to show a client in the field, or pull it up on my phone or iPad, or at the office where everybody has different web browsers. I guess if it

> doesn’t work on one thing that makes it very frustrating and then you are hesitant to use it again, almost, if it feels kind of limited. But if you have unlimited access and it’s easy to use, people will use it a lot more.

The prototype DSS provided as a web-based application can provide a high degree of flexibility for end users. The prototype DSS tends to be accessed by end users primarily from their desktop and laptop computers. Moreover, end users may occasionally view and present decision-making results generated by the prototype DSS on mobile internet-accessible devices such as tablet computers.

5.3.5. Digital Reports

All the interviewees share the same perspective that it is very beneficial that the prototype DSS can produce printable, digital reports. The interviewees agree that the reports provided in the PDF format are easily kept on their computers as backup records or paper trails. PDF reports are also useful for viewing on different computers and handheld devices that have PDF reader applications. Furthermore, PDF reports are easy to print out and share with other team members. It is important to note that PDF reports can be used as a means for documenting how the interviewees' decisions are made as well as the results of the interviewees' decisions. To provide an example of the interviewees' shared perspective, the statement given by Interviewee 2 is provided below:

> Yes, if you are in the situation where you are doing this type of thing, you want to share that with anybody or just even have it for a file for back up. Whether you store a copy electronically or print out a hard copy, or you collaborate with others on it, you have a paper trail and a tracking device shows that you did use something; you used something of it and handed someone a piece of paper. So I think that it’s helpful for documentation. (Is it useful for your decision to be documented?) Yes absolutely, the procedure and the methodology are value-added. Often somebody will go back and question why did you selected that. So if you go through that, especially as they are working at the budget for the project and they come back and say; do you think you really need this. They can come back and say, well you know, tell me what you do think to run the model right away and say, here is the difference. And so, I think it is important from both aspects. First, you got the documentation of the methodology. And second, you have an opportunity to go back and compare another result, but using the same tool say, here this is what you don’t get if you do this or, here’s what you do get if you do this. Yes, and it will be very fast, for that would be a huge time saver because a lot of time people have to go back to the drawing board, because they don't want to pay for an extra alternative for insulation, what does that do.

To make the PDF reports generated by the prototype DSS more useful, one of the interviewees, Interviewee 4, suggests that the current content and format of the reports are more appropriate for building designers, who already have technical knowledge, than for clients. If the prototype DSS can generate PDF reports that are easy to comprehend by nontechnical-orientated readers, it could be value-added to the system. In addition to the comment provided by Interviewee 4, Interviewee 10 suggests that

the PDF reports should provide the derivations of attribute values used in the CBA process to help better understand the accuracy of decision support information. The suggestion provided by Interviewee 10 is provided below:

> Yes, that's important because they are not looking at the screen with you. So you want to have something that you can share with them, but I was wondering if the report like; for instance, I enter the structural information and I wonder if the report can show the calculation behind the scene, because I don't know that the structural engineer will trust it unless he can see the calculation the same as the civil. The engineers will not trust the program unless they can see or follow the results, or else they just redo all the same work, and you spent a lot of time doing the work.

PDF report generation tends to be one of the most useful features for end users, because PDF reports are easy for offline viewing, printing, and sharing with project team members. PDF reports can be potentially used as a means for facilitating the collaboration between project team members. With some additional improvements discussed above, PDF reports have a considerable potential to be used by a larger group of end users.

5.3.6. Litigation

All of the interviewees are not quite sure about the benefit of the prototype DSS with regard to litigation. However, the interviewees are able to provide useful information about the legal issues that might be associated with green roofs and the decision support information provided by the prototype DSS. In general, the legal issues that are potentially related to green roofs may include waterproofing membrane, drainage system, and structural system damages. Because the prototype DSS does not provide detailed-level information about green roofs, the designer who specifies the green roofs would probably be the person who is responsible for the failures that might occur. To support this finding, a statement given by Interviewee 8 is provided below:

> Okay, that makes sense. So, I think what I would say is that this tool itself would not provide enough information to do with design; so there would be a lot more due diligence on the part of the designer to continue to develop the design of green roofs. So, I can see that it has been part of that, yes, but I think that there is a lot more that is going to go into the actual design if there is any sort of litigation.

Although the decision support information may not be involved in serious legal issues related to construction failures, the information could potentially be involved in the agreement on project requirements for green roofs between project stakeholders, especially between the building owner and the design team. One of the comments that represent this interviewees' perspective, stated by Interviewee 1, is provided below:

> What you got in the report, it shows why someone did something. It shows that someone did their due diligence related to picking the correct roof based upon the owner inputs. The downside of this is that the data would be, if the data is not correct, they should have picked G4 versus G3 because of data errors in software; I am not saying that there is, but if they base their decision upon inaccurate data then they would be beaten to death for many of the things. But by and large that they did everything in here and pick the right roof, and they show that to the juries that I did my due diligence and this is the report that shows that I did, they will give them more immunization when it comes to defending themselves in the court room.

For the situation above, the prototype DSS may potentially be involved in some legal issues related to the agreement on project requirements if the DSS provides inaccurate decision support information. Although the designer of record would normally be the person who is responsible for the false selection of green roofs as suggested by many interviewees, a legal disclaimer for acknowledging end users about the conditions and limitations of the tool needs to be provided as part of the prototype DSS delivery. For another issue related to the accuracy or the trustworthiness of the tool, Interviewee 6 mentions that the interviewee's company has a policy on the use of third-party Revit plug-ins that might legally affect the adoption of the prototype DSS. The statement given by Interviewee 6 is provided below:

> The other thing that would be a concern would be the software. If the software has security issues or bugs that might cause project central models to become corrupted or to become unusable or crash, it would be frustrating for the team as you would imagine. Typically, we have a BIM manager that works with the IT manager and they preapprove most type of either plugins or internet-based applications that will be brought into our models. So, either preapproved manufacturers that we would get that information from and that is trustworthy or it has to be something that gets clear. Not all firms allow that. Some might be more concerned about using a third-party tool like this. That is why I think it would be useful as a Revit plugin so it would have to end up to the point where it would be plugged into the Revit environment for them to approve of its use.

The information gathered from the interviewees suggests that the prototype DSS or the decision support information provided by the DSS may not be beneficial for litigation. However, if it contains errors or produces inaccurate decision support information, the prototype DSS could cause some problems to end users. Therefore, it is essential to combine a disclaimer with all forms of the prototype DSS delivery packages to ensure that end users understand the conditions and limitations of the delivered system.

5.3.7. Time Savings

Almost all of the interviewees share a similar perspective that the prototype DSS is beneficial for time savings. The prototype DSS can help reduce the time normally used for collecting data necessary for making decisions on vegetated roofing system selection. The interviewees find that the prototype DSS provides a framework that helps facilitate data collection and communication between project team

members. Once the data is collected and provided to the system, the prototype DSS can then provide feedback in the form of an interactive report that displays a comparison between a reference roof and eight generic types of green roofs together with other decision support information in a timely manner. To provide an example of the interviewees' shared perspective, a statement given by Interviewee 4 is provided below:

> Yes most definitely, the way it basically gives you a framework to work around. It makes it a lot quicker to think of the variables you need to cross compare, and it already has a template for you to input them on and give you feedback on. So, instead of me trying to reinvent the wheel every time compiling a list of information I need together, doing it, and then try to synthesize it myself between all the various options. It would be extremely tedious.

In addition to the main benefits above, Interviewee 8 suggests that the information provided by the prototype DSS can help narrow the range of green roof products. With a narrower range of products, the interviewee can potentially use less time to search for and collaborate with product vendors. For this matter, the prototype DSS seems to help reduce a significant amount of time when the interviewee has to work with clients in the public sector who typically has a demand for more generic specifications, which often have to be drawn up from information collected from multiple product vendors. Interviewee 8 also suggests that the software capability to provide end users with a list of products or vendors discussed in Chapter 4.4 seems to be beneficial for time savings. This capability could help save time if the information focused on local products or vendors. When dealing with green roofs, the interviewee sees that the local vendors are more familiar with the plant types for a specific micro climate. In addition, when dealing with LEED, it is important to limit the distance between the building sites and the material sources to receive LEED credits. One of the suggestions provided by Interviewee 8 is presented below:

> I know that one of the things that you have there is potential for contact with an installer and such that I think that would be very helpful. I mean you have to be very careful about embedding that because there will really need to be people who really actually provide. It would really have to be suppliers or the providers because every contractor is going to say that they can install it, but it would just make sure that I know the right people in the right locations.

Among all interviewees, Interviewee 10 is not quite sure about the time savings benefit provided by the prototype DSS. However, the interviewee sees that the prototype DSS may help save time when using it in new building projects located on unfamiliar locations. According to the interviewee, the prototype DSS may be rarely used in projects located in a similar area because there is a tendency that a similar type of green roof will be used in such projects. The statement given by Interviewee 10 is provided below:

> I am not sure because we work a lot in the same climate and buildings, in DC are very standard in height and structural capacity. I don't know that once we did it once, it would be the same answer throughout unless we switched climate or switched cities. It doesn't add the same exact value every time, and then sometime when we're working in the climate that we're familiar with,

like if I do work in Thailand or something like that, I end up doing a lot more research on the climate, which it is not the climate I live in. In those cases it becomes extremely valuable again.

The information gathered from the interviewees indicates that the prototype DSS potentially helps save the end users time normally used for collecting data from different sources and transforming the collected data into decision support information. The information also helps inform the future development of the prototype DSS on how the products or vendors list should be provided for end users. Based on the gathered information, the products or vendors list seems to be useful for end users, but not the contractors list. As well, the list should concentrate on the local product or vendor information as much as possible.

5.4. Interviewees' Feedback on the Fully Developed System

5.4.1. Potential Impact of Decision Support Systems (DSS) on Interviewees' Decision-Making and Design Processes

Almost all interviewees agree that having a collection or library of Decision Support Systems (DSS), similar to the prototype DSS, would positively impact their decision making and help improve their design process. In general, the decision-making methods used by the interviewees tend to rely on one's experiences and intuitions. As opposed to the formerly employed methods, the interviewees see that DSS seem to provide knowledge-driven decision-making methods for helping them make decisions. The interviewees also recognize that the DSS can help document both the decisions and the reasoning behind such decisions. By using DSS, the interviewees believe that their design decisions can be accurately made in a timely manner. The interviewees also see that the DSS can help them communicate their design decisions more effectively. To support this finding, the statement given by Interviewee 4 is presented below:

> I would strongly agree because I feel like, without the tool like this, the conversation between disciplines and your client is negative. If you don't have the information to back you up, all you really have to go off of it is intuition, and a lot of people, without the data, everyone thinks that you are trying to pull the wool over their eyes. Sort of get, I don't know. It's a very interesting relationship.

In addition to the potential benefits above, the interviewees see that DSS can help reduce the time that they normally use for making and communicating their decisions. For this benefit, Interviewee 9 elaborates that DSS have considerable potential to shift some of senior-level employees' decision-making responsibilities to more junior-level employees. This change in decision-making responsibilities will provide senior-level employees with more time to work on the more important decision-making tasks. The statement given by Interviewee 9 is provided below:

> Strongly agree with it, having a lot of these tools about the right things would be great largely because it saves time. I think another reason is it particularly changes who do that kind of work,

> and you can give that kind of work to a more junior person, which is good. So, it doesn't tie up as much the senior time like in the past. It's a reduction in time and also a reduction in experience required to make the decision, to make a good decision.

Among the interviewees, Interviewee 2 is not sure about how DSS would impact the interviewee's decision making and help improve the interviewee's design process. The interviewee argues that if the library provides similar tools that lead to the same decisions or provides tools for solving too simple problems, the library might not be useful. The interviewee's argument is provided below:

> I am going to say that I neither agree nor disagree. I somewhat agree and somewhat disagree. I guess the middle question makes sense, and the reason why is, when you have too many systems, it just adds more time and confusion to the process. That is the disagree side of it. On the agree side is, when you have multiple sources to achieve the same information, it provides some validations, and it's a good thing, but if you want a design tool that save you time you don't want to have to go to six of those. To save time, you just use the different one.

Based on the information above, the library of DSS tends to have positive impacts on end users' decision making and help improve end users' design process. By providing them with a library of DSS, the end users would be able to make decisions more accurately and quickly based on structured decision-making methods. The library of DSS can also potentially help increase the efficiency of end users' design process through the distribution of knowledge.

5.4.2. Interviewee's Willingness to Use Decision Support Systems for Design Assistance

All interviewees, who have been involved in the role of a decision maker in building design projects, are interested to use the full version of the DSS library that will be developed in the future, because they recognize that having the DSS library will be value added to their work. When it comes to values, from the study the interviewees see that the DSS have a considerable potential to help them improve their decision-making and design processes in certain ways. First, the DSS seem to provide more structured and knowledge-driven methods for decision making compared to conventional methods, which tend to rely on one's intuitions and experiences. Second, the DSS provides decision support information which potentially helps the interviewees make decisions quickly and accurately. Third, the DSS can be used as means for documenting how decisions are made. Fourth, the DSS can be beneficial for time savings and for improving the effectiveness of design process through knowledge distribution. Fifth, the DSS can be used as a platform for utilizing data contained in BIM models in support for decision making. Lastly, the DSS can help facilitate communications between project team members.

In addition to the values above, the information gathered from the interviewees reveals some criteria for adopting decision support tools like the DSS. First, the interviewees would consider using the DSS if they provide appropriate decision support information for the problems that they often encounter. For example,

some interviewees suggest that the VRSS DSS may not be appealing enough to be adopted, because the interviewees have rarely been involved in green roof projects. Second, the interviewees would use the DSS if they are easy to use. The interviewees suggest that decision support tools might not be used on a daily basis; therefore, they have to be user-friendly to reduce their learning time and effort. Third, some interviewees would use the DSS if they are highly customizable. The interviewees suggest that the DSS should be flexible enough for them to include their organizational knowledge into the system. Lastly, the interviewees who are regular BIM users would consider using the DSS if they are seamlessly integrated with BIM authoring tools. The interviewees see that the capability in which the VRSS DSS can generate BIM models on the fly for them to download could be potentially useful; however, it is important for the interviewee to be able edit the models inside their BIM authoring tools. To provide an example of the revealed criteria, the statement given by Interviewee 8 is provided below:

> Yes, you know, I think it would be interesting to see what the other modules would be. I think that the green roof one is interesting, but we obviously don't do a tremendous amount of them...
> …it is also obvious that you are going to have a firm knowledge base. So, how do you include that knowledge in the DSS, as well? It's almost like customizable. I mean, are you able to do that or are you able to input additional information? It would be, I guess, part of the consideration, as well.

The information gathered from the interviewees suggests that end users tend to use the DSS if they can serve needs and be value-added to their work. The information indicates that the interviewees are interested to use the DSS because they see the potential benefits of the DSS. In addition to providing end users with substantial benefits, it is important for the DSS to serve end users' needs. It can be considered that the information provided above helps clearly identify the end users' needs, which are very useful for the future development of the full version of the DSS library.

5.4.3. Expectations for the Fully Developed Decision Support Systems

Based on their hands-on experience with the prototype DSS, the interviewees are able to express their expectations for the future DSS development which can be divided into three categories. The first category encompasses the interviewees' expectations for the improvements on the current version of the prototype DSS. The second category involves the interviewees' expectations for additional features and functions, which could potentially be part of the existing DSS. The last category includes the interviewees' expectations for new DSS that could possibly be included in the DSS library. The interviewees' expectations related to these categories are provided below.

For the current version of the prototype DSS, the interviewees recommend that interactions between the DSS web interface and end users should be somewhat smoother than what they have experienced. To fulfill this, it is essential to examine how the Asynchronous JavaScript and XML (AJAX) technology, widely used for developing Rich Internet Applications (RIA), should be implemented to help improve the DSS web interface. Some interviewees also suggest that some of the Memory Aids (M) of the ROMC elements

should be improved to provide additional information that helps end users make sense of the inputs. For example, a set of coefficient of discharge values for a variety of conventional roof types should be provided to help end users easily pick a value that well represents the characteristic of their proposed roof. For this concern, additional studies should be conducted to provide end users with relevant baseline input values. Furthermore, some interviewees suggest that the user interface for collecting the importance score of advantages should provide support for helping them determine appropriate advantage scores. This includes rewording the existing descriptions of advantages to make them easier to comprehend and providing support information to help them get familiar with the CBA scoring technique. For example, a recommendation given by Interviewee 12, which represents the interviewees' improvement recommendations, is provided below.

> I think the biggest improvement would be working on the user interface where you put the factors in. The language of the factors and advantages are unclear. That portion would be the weakest part right now.

Beyond the system improvements above, some interviewees also recommend that the DSS should provide support for the International Standard Units (SI) and other measurement units. Interviewee 11, who is a mechanical engineer, elaborates on this issue that the units, currently employed as part of the user interface for collecting potential energy savings variables, are not typically used by mechanical engineers. Some interviews also request for new modes of input and output. For instance, Interviewee 5 recommends that the DSS should provide different modes of inputs such as inputs for existing and new buildings. In addition, Interview 4 recommends that the DSS should provide a new output mode for client reports, which would allow the user to send easy-to-understand reports directly to clients without recompiling and extensive communications on the documents. Also recommended by Interviewee 4, the DSS could potentially become a source for design creativity by providing case studies that focus on the creative use of environmental building systems like green roofs. Furthermore, as suggested by Interviewee 10, the DSS should provide end users with a file or project management feature with the Create, Retrieve, Update, and Delete (CRUD) functions. The suggestion given by Interviewee 10 is provided below:

> I would like to see previous versions to be able to gain access to different projects or different runs within different projects. Delete and keep, you know. You can keep the successful ones where you can keep as a library of what you are working on, right. Let's see USGBC LEED. In here, I have a list of all of the projects. So it would be interesting to have, and in some way similar to, this. So, you can see the projects that you run and may be the date that you run them on and the location that you can usually, if you are in a pinch, say I have figured it out for that location.

In addition to the DSS for vegetated roofing system selection, interviewees are also interested to have several other DSS in their DSS collection, primarily for environmental building systems, renewable energy systems, and compliance/accreditation assistance. According to the interviewee, one of the DSS for

environmental building systems that the interviewees are really interested to have is the DSS for integrated lighting design and building envelope design. The interviewees are also interested to have DSS for assisting them with renewable energy systems such as solar thermal and geothermal systems. Furthermore, the interviewees are interested to have the DSS for assisting them with rating systems like LEED or maybe helping them decide whether or not they should go for LEED. To provide an example of the interviewees' requests for new DSS, the statement given by Interview 8 is provided below:

> I think there are a lot of opportunities there whether it will be building envelope or PV, solar thermal or geothermal, kind of that alternative energy. The opportunities on how to optimize that kind of systems, I think that would be helpful, as well.

The interviewees' expectations for the future versions of the prototype DSS and the DSS library not only provide adequate information that could potentially guide the future development of the DSS, but also provide genuinely useful information could possibly lead to future research in this domain. For the VRSS DSS, the information about system improvements and additional features and functions should be taken into consideration when the prototype evolves into the full software development process. For the new DSS, the information suggests that the end users seem to be interested in adopting the DSS that are able to provide decision support for the sophisticated, unstructured design problems that they frequently encounter in their design practices.

5.4.4. Acceptable Cost

As mentioned in Chapter 3.1, DSS prototyping is not only used as a means for examining how end users perceive the values of having a DSS library, but also for understanding the maximum cost that end users might anticipate paying to obtain the benefits that the collection of DSS provide. While the values or benefits of the DSS are extensively discussed in the previous sections, this section aims to examine the cost of the whole DSS library suggested by the interviewees. The information about the possible cost of the DSS can be seen as one of the essential factors behind the investment of the full DSS development in the future.

Fundamentally, the numbers in U.S. dollars per user account per year given by the interviewee are varied depending on their interpretation on the benefits provided by the DSS library and how frequently they might use the DSS. The information gathered from the interviewees suggests that the interviewees tend to compare the DSS with other software applications that they have been using, which provide a similar degree of benefits. The interviewees then use the costs of the employed software applications for justifying the cost of the DSS largely based on the frequency of use. Based on this principle, eleven of the twelve interviewees are able to give the numbers in U.S. dollars that represent the maximum cost of the DSS as follows:

> 30.00 100.00 200.00 300.00 400.00 500.00 500.00 1000.00 2000.00 2000.00 2500.00

Based on the numbers above, a statistical method suggested by Ott and Longnecker (2001) can be used to help make inferences about the possible DSS cost. Among the numbers listed above, the number of $30.00 per yearly subscription is given by an interviewee who considers purchasing the DSS in the form of a plug-in suite for BIM authoring tools, which could possibly be integrated into BIM authoring tools installed on a significant number of employees' computers. In contrast, the remaining numbers are given by the interviewees who consider using the DSS as a standalone platform in support for decision making. Therefore, it is important to note that the number of $30.00 per yearly subscription is excluded from the statistics analysis presented below.

In Figure 5-2, the descriptive statistics information indicates that the mean, median, and mode of the data set are significantly diverse. In general, if the mean is greater than the trimmed mean, median, and mode, the data distribution tends to be nonsymmetrical and skewed to the right. The boxplot and normal probability of the cost data presented in Figure 5-2 also reveal the extreme right skewness of the data. As a result, the mean is not an appropriate presentation of the DSS cost.

Descriptive Statistics Information	
Mean	950
Standard Error	279.3842436
Median	500
Mode	500
Trimmed mean (10%)	862.5
Standard Deviation	883.490552
Sample Variance	780555.5556
Kurtosis	-0.97432331
Skewness	0.882136689
Range	2400
Minimum	100
Maximum	2500
Sum	9500
Count	10
Largest(1)	2500
Smallest(1)	100
Confidence Level (95.0%)	632.0110678

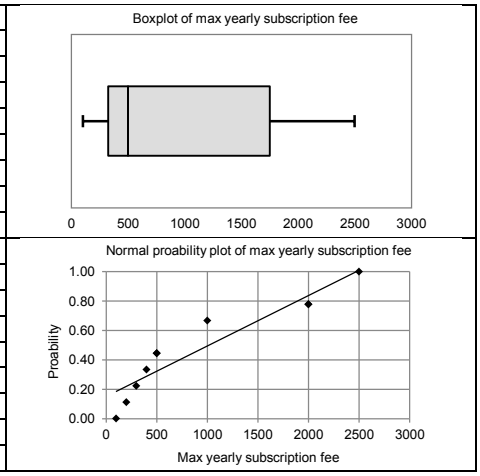


Figure 5-2: Descriptive statistics information

In this case, the median of $\widehat{M} = (y_{(5)} + y_{(5+1)})/2 = 500$ seems to be a more appropriate representation of the DSS cost than the mean. As shown in the list above, there are four numbers that evenly fall below and above $500.00 whereas there are six numbers below and four numbers above the mean of $950.00. To confirm this information, Ott and Longnecker suggested that a confidence interval for the population median M should be constructed using the binomial distribution with $\pi = 0.5$. In their book, Ott and Longnecker have provided a table that contains values for $C_{\alpha(2),n}$, which are percentiles from a binomial distribution with $\pi = 0.5$. From the table, in order to construct a 95% confidence interval for the population median, the $C_{.05,10}$ of the cost data can be defined as 1. Based on this $C_{.05,10}$ value, $L_{.025} = C_{.05,10} + 1 = 1 + 1 = 2$ and $U_{.025} = n - C_{.05,10} = 10 - 1 = 9$ represent the lower and upper limits of the interval. Thus, the 95% confidence interval for the population median can be given by $(M_L, M_U) = \left(Y_{(2)}, Y_{(9)}\right) = (200.00, 2000.00)$. For a 95% confidence interval, the binomial distribution principle suggests the

confidence level has to be greater than 0.95 or 95%. When it comes to the cost data, the exact level of coverage can be given by $1 - 2Pr[Bin(10, .5) \leq 1] = 0.995$, which is greater than 0.95. As a result, it is at least 95% confident that the median cost of the DSS is between $200.00 and $2,000.00.

The information above suggests that the cost of $500.00 seems to be appropriate for representing the maximum cost that end users might anticipate paying to obtain the benefits provided by the DSS collection. To make sense of this number, it is essential to look further into the information gathered from interviewees gain an in-depth understanding the DSS cost. Among two interviewees who suggest that the cost of $500.00 would be appropriate for the DSS, Interviewee 10 mentions that the DSS would probably be obtainable through a general budget if the yearly subscription fee is lower than $500.00. The interviewee's comment is provided below:

> This is a hard question because I am not in control of the money. So, I am going to say less than $500 dollars total. And, the reason is that I can pay for something like this. It comes out of a general budget, right.

It is essential to note that the possible cost of the DSS collection discussed in this section is drawn from the cost data estimated by the interviewees, based on their hands-on experience of the prototype DSS and their interpretation on what the fully functional DSS library might be. Although the cost value might not be highly accurate due to the limited size of data, it may be potentially used as a starting point for justifying the investment cost of full software development effort in the future.

5.5. Interview Study Conclusion

The post-use interview study discussed in this chapter began with collecting data from research participants through a series of in-depth interviews focusing on their experiences of the prototype DSS developed as part of this research effort. Once the data was collected in the form of interview records, the interview records were transcribed into text documents. Afterward, a series of data reduction and coding activities was performed to break a large piece of textual data into manageable pieces. The reduction and coding activities help reveal the information that was useful for making interpretation and developing an in-depth understanding about the usefulness of the prototype DSS. While the in-depth understanding of the usefulness of the prototype DSS was discussed extensively in the previous sections in the form of textual descriptions, this section aims to present the generalized information, which is useful for conclusion drawing. The understanding about the usefulness of BIM-compatible, web-based, knowledge-driven DSS can be generalized and displayed in Table 5-2.

Table 5-2: Generalized information about the usefulness of DSS

Category/ theme	Region A	Region B	Region C
Profession	Architecture (5), sustainability consultant (1), and roofing consultant (1)	Sustainability consultant (1), roofing consultant (1), roofing researcher (1), and roofing product manufacturer representative (1)	Mechanical engineer (1)
BIM background	Moderately to extremely familiar	Not at all to moderately familiar	Moderately to extremely familiar
Green roof background	Moderately to extremely familiar	Moderately to extremely familiar	Not at all to moderately familiar
Decision support tool	Useful (6/7) Support decision making, provide framework, facilitate data gathering, facilitate collaboration, facilitate knowledge distribution, provide presentation content	Useful (3/4) Facilitate data gathering, facilitate collaboration, provide education, simplify decision-making processes	Useful Provide framework, and help save time
Decision support information	Useful (6/7) High-level decision support information, comparative information, decision-making variables, decision-making factors, baseline information, information presentations	Useful (3/4) Decision-making variables, performance criteria	Useful High-level decision support information, comparative information
BIM compatibility	Useful (7/7) Data extraction, interoperability, automated model generation, downloadable models, potential plugins	Useful (2/4) Support open standard, automated model generation, downloadable models	Useful Data extraction, no need for extensive BIM knowledge or skills
Accessibility	Useful (6/7) Cross operating system platforms, mobility, flexibility	Useful (3/4) Cross operating system platforms, dynamic updates	Useful Cross operating system platforms
Digital reports for viewing, printing, and sharing	Useful (7/7) Offline viewing, printing, sharing, backups, paper trials, decision documentation	Useful (4/4) Offline viewing, printing, sharing, paper trials, backups, decision documentation	Useful Offline viewing, printing, sharing
Litigation	Useful (2/7) Due diligence	Useful (1/4) Due diligence	N/A
Time savings	Useful (5/7) Reduce decision-making time, reduce research time, reduce data gathering time, reduce BIM modeling time, reduce time used for communications, reduce time through knowledge distribution	Useful (4/4) Reduce time used for communications, reduce data gathering time	Useful Reduce research time, reduce data gathering time
Positive impact on decision making and design processes	Agree (7/7) Improve design process by reducing time for proposing ideas, Improve design process by reducing data gathering time, Improve decision-making process by providing structured methods, improve decision-making process by providing knowledge	Agree (3/4) Improve decision-making process by reducing data gathering time, Improve decision-making process by providing knowledge	Agree Improve design process by reducing time for proposing ideas
Willingness to use the full version DSS	Willing to use (7/7) Enhance decision-making and design processes, compatible with BIM tools, useful for multifactor decision making,	Willing to use (3/4) May not use it directly, use as a collaboration tool, will recommend to designers, may use for firm knowledge management	Willing to use
Expectations for future development	Multiple modes of input and output, environmental building systems, renewable energy, compliances, BIM tool plugins, project management, user interface improvements	Multiple modes of input and output, environmental building systems, renewable energy, user interface improvements	Multiple modes of input and output, environmental building systems, user interface improvements
Maximum cost for obtaining provided benefits	$30.00, $200.00, $400.00, $500.00, $500.00, $1,000.00, and $2,000.00	$100.00, $300.00, $2,000.00, and $2,500.00	N/A

As summarized in Table 5-2, DSS seem to be useful for building practitioners who are responsible for making decisions in building projects. Based on decision-making responsibilities, architects or other building designers tend to be the primary end users of the DSS. In addition, the DSS may also be used by practitioners who provide consultation to the decision makers, for collaborating on identifying decision-making variables and interpreting decision support information. Commonly, the end users would consider using the DSS because it helps improve their decision-making and design processes by reducing the time usually used to achieve decision-making and design tasks. When it comes to system features and functions, the end users would primarily decide to use the DSS based on their value as a decision support tool and the importance of the decision support information they provide. Once the end users understand these primary benefits, other features and functions would be seen as complementary benefits to the primary benefits. Among the complementary benefits, the features and functions that help promote collaborations between building project stakeholders tend to be the most important features and functions of the DSS due to the nature of the practitioners' decision-making and design processes, which require a high degree of collaboration. The capabilities of the DSS proposed in this study that help facilitate collaborations include the BIM support and digital report capabilities. As web-based applications, the DSS are proved to be useful when they can be used on multiple operating systems whereby desktop and laptop computers seem to be the platforms mostly used by the practitioners to perform decision making. Due to the level of details of decision support information that the DSS provide, the DSS may not be beneficial to the end users when it comes to litigation. For the future versions of DSS, the domains of the problems encountered by the end users that should be taken into consideration include the problems in the environmental building system and renewable energy domains. Based on their potential benefits, the maximum cost that the end users anticipate paying would be around $500.00 per year.

Based on the detailed and generalized information about the usefulness of the DSS as seen by building practitioners presented throughout this chapter, it can be concluded that web-based, knowledge-driven DSS can be developed and are useful as a BIM-based platform for facilitating building design practitioners' decision-making and design processes. It is essential to note that the DSS prototyping process, by its very nature, is very useful as a means for gaining an in-depth understanding of the features and functions of the DSS and for examining the benefits and cost threshold of the full DSS development effort. Therefore, after the DSS is fully developed and released, additional studies should be conducted to extend the knowledge about the benefits of implementing BIM-compatible web-based knowledge-driven DSS as a platform for providing decision support for building design practitioners.

Chapter 6

6. Conclusion

6.1. Research Summary

This study aims to understand the possibilities for developing a BIM interoperable web-based DSS for vegetated roofing system (green roof) selection and the usefulness of the DSS as seen by building design practitioners. To fulfill these purposes, this study was conducted based on two hypotheses:

Hypothesis 1 - The BIM interoperable web-based DSS can be developed to improve the quality of green roof design decisions

Hypothesis 2 - The BIM interoperable web-based DSS can improve the quality of green roof design decisions.

Based on the hypotheses above, this study encompassed two phases. The first phase of this study adopted the prototyping process which comprises five activities: 1) communication, 2) quick planning, 3) modeling/quick design, 4) prototype construction, and 5) deployment, delivery, and feedback. The prototyping process was used as a basis for the prototype DSS development discussed in Chapter 4. The second phase of the study adopted the qualitative research process which includes four activities: 1) data collection through interviews, 2) data reduction/coding, 3) data display, and 4) conclusion drawing and/or verifying. The qualitative research process was used as a basis for the post-use interview study presented in Chapter 5.

Referring to the information provided in Chapter 4, a prototype knowledge-driven web-based DSS can be developed based on today's Internet and WWW technologies using multipurpose programming languages together with Web 2.0 scripting and markup languages. The prototype DSS is capable of providing decision support for design practitioners who are choosing appropriate green roofs for building projects based on the following five factors: 1) storm water flow reduction, 2) potential energy savings, 3) approximate sound transmission class, 4) surplus dead load of roof systems, and 5) potential contribution to LEED certification. The prototype DSS is also compatible with the Industry Foundation Classes (IFC) specification version IFC2x Edition 3 Technical Corrigendum 1 (IFC2x3 TC1), which allows the DSS to be used in conjunction with major BIM authoring tools employed by design practitioners. As well, the prototype DSS can automatically produce Portable Document Format (PDF) reports, which design practitioners can view offline, print, and share with the project team members. The presence of the prototype DSS suggests that a BIM interoperable web-based DSS can be developed to improve the quality of green roof design decisions; thus, Hypothesis 1 is accepted.

As mentioned in Chapter 5, the developed prototype DSS was delivered to twelve research participants who represent a larger population of building design practitioners for evaluations. Based on the feedback

given by the research participants, the prototype DSS can possibly help improve the quality of green roof design decision in particular ways. First, the prototype DSS can potentially help save the amount of time used for selecting a green roof for a building project. Second, the prototype DSS seems to provide useful decision support information for green roof selection. Third, the prototype DSS seems to be compatible with BIM authoring tools used by the research participants. Fourth, the prototype DSS tends to be easily accessible across computer devices and operating systems. Lastly, the prototype DSS tends to be beneficial for decision documentation and multidisciplinary collaboration. Overall, the research participants are in agreement that the prototype DSS is useful as a decision support tool for green roof selection. Therefore, Hypothesis 2 is also accepted.

In a boarder perspective, the prototype DSS would probably be beneficial for professional practice and education. For professional practice, a collection of DSS could possibly change the decision-making and design processes currently used by building design professionals through organizational-wide knowledge distribution. For education, the collection of DSS could potentially be used to provide students with building design knowledge and structured decision-making processes.

In conclusion, the findings of this study suggest that a BIM interoperable web-based DSS can be developed and can potentially help improve the quality of green roof design decisions. The prototype DSS development and the post-use interview study on the usefulness of the DSS performed as part of this research helped to, 1) identify the core requirements for system features and functions and 2) determine the operational benefits of the system. The in-depth understanding gained from this study has a considerable potential to be used as a basis for the full DSS development in the near future.

6.2. Suggestions on the DSS Development and Post-Use Interview Study

6.2.1. Suggestions on the DSS Development

Based on the knowledge gained from this study, there are some practical suggestions for the next iterations of the prototype DSS development that should be taken into consideration. These suggestions can be categorized into three broad categories: exploring DSS design alternatives, adding new features to the prototype DSS, and developing new DSS to be included to the DSS collection. The suggestions that fall into these three categories are given below.

As presented in Chapter 4, this study adopts an evolutionary software engineering process, also referred to as the prototyping process, for developing the prototype DSS. This process typically serves as a mechanism for defining requirements for software features and functions. Therefore, the design efforts involved in this process often disregard the quality and maintainability aspects of the software being developed. To improve the system quality and maintainability in the next iterations of DSS development, suggestions on DSS design derived from the knowledge gained from this study are provided in Table 6-1.

Table 6-1: Possible design alternatives

Web application layer	Design alternative
Interface design	Exploring the use of AJAX technology to help improve the user-friendliness of the DSS
Aesthetic design	Designing alternative CSS schemes
Content design	Improving the user support mechanism Adding details to downloadable PDF reports and IFC files
Navigation design	Offering alternative navigation path ways
Architecture design	Reconfiguring the MVC framework and DSS components
Component design	Examining the implementation of ifcXML and IFC/BIM servers

In Table 6-1, the alternative design suggestions are categorized based on the six layers of web application design. For user interface design, the user experience may be improved by integrating the AJAX technology, which is widely used as a basis for RIA development, into the user interface of the DSS. For aesthetic design, the CSS schemes used for defining the formats of HTML elements may be modified to improve the visual aesthetic of web pages produced by the DSS. For content design, the user support mechanism may be improved especially by providing additional information that helps users justify the values of decision-making variables required by the DSS. In addition, detailed information may be added to PDF reports and IFC files produced by the DSS. For navigation design, alternative navigation path ways may be added to the user interface to help increase the effectiveness of system navigation. For architecture design, the DSS and the MVC framework components may be rearranged to improve system quality and enhance system maintenance. Lastly, for component design, alternative data exchange formats such as ifcXML and data repositories such as IFC servers may be implemented to improve the effectiveness of BIM data utilization.

According to the information given in Chapter 5, there are four essential features that should be taken into consideration in the next iterations of DSS development. The first feature is the products and/or vendors search, which would allow building designers to find information about products and/or vendors using a variety of filters such as distance and cost. The second feature is the report customization, which would enable end users to create reports for different audiences who are involved in a building project. The third feature is the project management, which would allow end users to create, retrieve, update, and delete their projects. Lastly, the plug-ins should be developed for facilitating the data exchange between the DSS and BIM authoring tools.

The information provided in Chapter 5 also helps indicate the demands for new DSS that should be included to the DSS library. These DSS can be divided into three groups. The first group includes DSS for assisting building designers with the decisions about environmental building systems such as integrated lighting and building envelope design systems. The second group comprises DSS for aiding building designers in the decisions concerning green or sustainable building compliances such as LEED and Energy Star. The last group encompasses DSS that provide decision support information about renewable energy such as solar and geothermal energy.

In summary, this study involves developing a prototype DSS for vegetated roofing system selection and studying its usefulness as seen by building design practitioners. The understanding gained from this

study helps reveal the information about possible design alternatives, new system features, and new DSS. This information can be used as a basis for the full DSS development that may take place in the near future.

6.2.2. Suggestions on Post-Use Interview Study

The understanding gained from conducting the post-use interview study suggests that the data collection procedures seem to have a significant impact on the study results. Therefore, to expand the knowledge about the usefulness of the prototype DSS, there are some suggestions on the data collection techniques and their potential impact on the research outcomes that should be taken into consideration. These suggestions are provided in this section.

Based on the lessons learned from this study, the execution of interview questions and the language used in the questionnaire seem to have a significant impact the study results and should, therefore, be reexamined. For example, the rating scale used in the questions that ask about the interviewees' familiarities with green roof design and building information modeling tends to be problematic when it comes to data interpretation. As presented in Appendix C, the in-use rating scale was defined as follows: Not at all familiar, Slightly familiar, Moderately familiar, Very familiar, and Extremely familiar. The problem with this rating scale was discovered when the interviewees were grouped based on their familiarities with green roof design and building information modeling as presented in Figure 6-1. The problem occurred with the interviewees who are moderately familiar with green roof design and building information modeling. As shown in Figure 6-1, these interviewees (e.g., Interviewee 11 and Interviewee 12) tend to fall onto the boundary of the groups rather than falling within a group. Thus, a new rating scale could be redefined to increase the understanding about the interviewees' background as follows: Not at all familiar, Slightly familiar, Less than moderately familiar, More than moderately familiar, Very familiar, and Extremely familiar. This alternative rating scale would probably help clarify the ambiguity about the interviewees' background presented in Figure 6-1.

In addition to reexamining the execution of interview questions and language used in the questionnaire, expanding the range of sampling variation and increasing the sampling size would possibly impact the study results. As mentioned in Chapter 5, the conclusion of the post-use interview study was drawn based on the data obtained from the interviewees whose background fall within Region A, B, and C presented in Figure 6-1. Therefore, it is possible to increase the understanding about the usefulness of the prototype DSS by expanding the range of sampling variation to include the building design practitioners whose background fall with within Region D shown in Figure 6-1. Also discussed in Chapter 5, the number of research participants seems to be too small to be used as a sampling size for statistical analyses based on normal population distributions. Therefore, increasing the sampling size would probably help increase the understanding of the study results in certain areas. For example, it would help to accurately identify the maximum yearly subscription fee that end users anticipate paying.

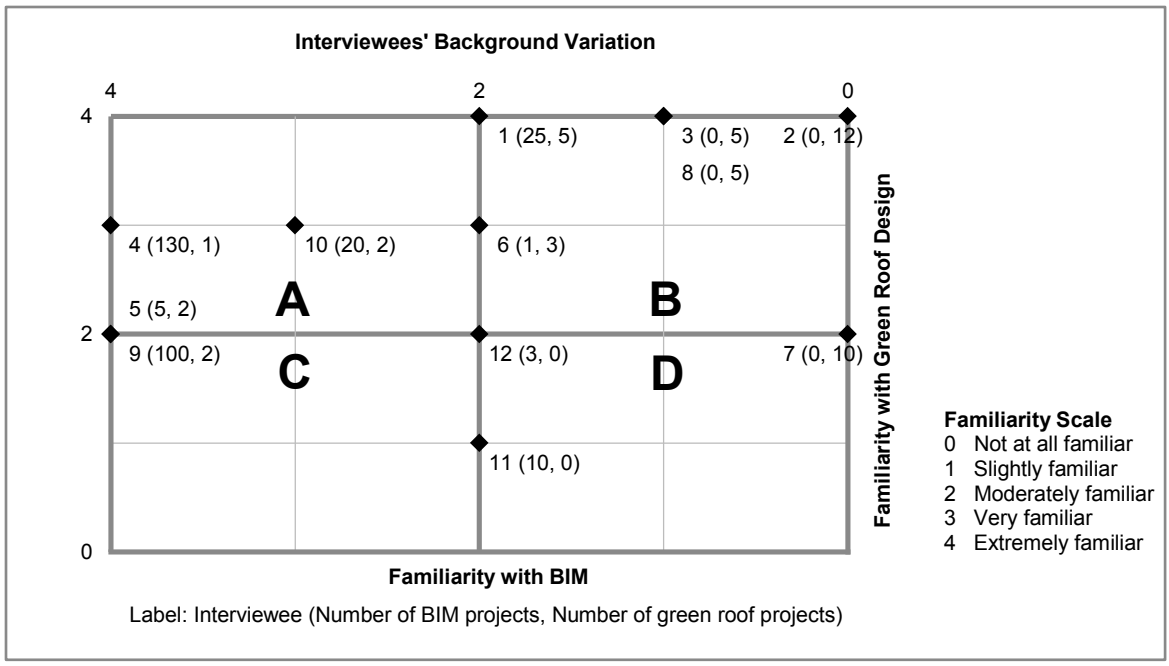


Figure 6-1: Interviewees grouped by their familiarities with green roof design and building information modeling

As presented in Chapter 5, the in-depth interviewing seems to be the dominant data collection method used in the post-use interview study. In addition to this method, there are other methods that may be used to help extend the knowledge about the usefulness of the prototype DSS presented in this study. Among a variety of data collection methods, observation seems to be the most appropriate method to be used in addition to in-depth interviewing, because it provides opportunities to learn about the effectiveness of the prototype DSS execution directly during the time that research participants are interacting with the system.

The data collection techniques typically used in the data collection phase of the qualitative research process tend to be associated with a set of limitations. Based on the lessons learned from the post-use interview conducted as part of this study, such limitations seem to impact the study results to a certain degree. Therefore, the understanding gained from this study could possibly be extended if the potential improvements suggested above were taken into consideration in the additional studies that may be conducted in the future.

6.3. Future Research

In 2010, the U.S. building sector alone consumed about 40 quadrillion Btu, which accounted for about 7.9% of global energy consumption, and produced 2,268 million metric tons of carbon dioxide (CO_2) emissions, which accounted for about 7.4% of global emissions (D&R International 2012). Based on this information, the U.S. building sector in 2010 consumed about 1.38% more energy than in 2007 and produced about 2.28% less CO_2 emissions than in 2007. These insignificant changes in energy consumption and CO_2 emissions indicate that the building industry is still in need of new business models

to help achieve the industry sustainability goals. As the most important drivers of new business models, the Integrated Practice (IP) and Building Information Modeling (BIM) adoption rates seem to be rapidly growing. In 2011, architects accounted for 13% of the American Institute of Architects (AIA) members that implemented IP for delivering building projects (AIA 2011). This information suggests that IP had been adopted by the early adopter group of its adopters. In 2012, 75% of building practitioners, which included but not limited to, architects, engineers, and contractors, implemented BIM processes in their projects (McGraw-Hill Construction 2012). This information suggests that BIM began to be adopted by the late majority group of its adopters. This industry background information opens windows of opportunities not only for the full library of DSS to be used by building design practitioners in the near future, but also for future research that would probably help extend the knowledge gained from this study. Potential future studies are briefly introduced below.

The findings of this study indicate that a library of DSS, which can be used in conjunction with BIM authoring tools and facilitate the integrated practice, seems to have a positive impact on building design practitioners' decision-making and design processes, and possibly help them achieve building sustainability goals. The findings also suggest that there is a demand for the DSS library to be used as part of decision-making and design processes in the future. Therefore, a study on DSS solutions for building design businesses (e.g., developing in-house, acquiring off-the-shelf DSS packages or services, and/or outsourcing) should be conducted to help understand how the DSS library should be provided to building design practitioners.

As part of this study, there is a body of evidence which suggests that building design practitioners would be interested in integrating their organizational knowledge base into the DSS. To fulfill the practitioners' need, how the practitioners manage their organizational knowledge and the resulting knowledge representations normally used as a basis for developing knowledge-driven DSS has to be well understood. Therefore, a study on knowledge management in building design companies should be taken into consideration.

When it comes to knowledge capturing, social networking seems to be one of the emerging Web 2.0 technologies that could potentially be used for obtaining knowledge from building design experts. For example, LinkedIn, one of the most popular professional social networks, has provided its Application Programming Interface (API) library for other software applications to interact with any available content across the network. These kind of capabilities would probably be useful for collecting knowledge that experts share in a particular group discussion. Based on the availability of the social networking technology, a study on the effective use of social networking for knowledge capturing should be undertaken to help maintain and extend the design knowledge provided through the DSS library.

As well, the understanding gained from this study shows that the integrated practice has progressively influenced design-practitioners' decision-making and design processes. In a multidisciplinary building design company, for instance, a team of building designers including, but not limited to, architects,

engineers, sustainability specialists, and other consultants tends to be formed early on in the building design process. The team is responsible for making high-level design decisions at the beginning of the building project, including the decisions on the implementation of green roofs. Based on this finding, it is important to conduct a study on collaborative decision-making and design processes to ensure that DSS can effectively facilitate collaborations between the project team members.

Based on the industry need for new business models and the operational benefits that the DSS library provides, there is a considerable potential that building design practitioners would probably use the DSS library as a platform for implementing data contained in BIM models and facilitating the integrated practice to help attain the industry sustainability goals. The in-depth understanding about the DSS gained from this study helps indicate the needs for further studies in different areas. The additional knowledge gained from these studies would very likely help the DSS library effectively and efficiently serve industry needs in the near future.

Appendix A

Appendix A. Attribute Values for the Vegetated Roofing System Selection Framework

Table A-1: Storm water flow reduction

Roof system type	Y value (fraction of annual average rainwater retained or dissipated) or 1 - Annual Coefficient of Discharge (ψ_a)
0 (Reference roof)	0.2
1	0.4
2	0.45
3	0.50
4	0.55
5	0.60
6	0.60
7	0.70
8	0.90

Table A-2: Potential energy savings

Roof system type	Net savings ratio resulting from the U.S. Department of Energy's (DOE) Green Roof Calculation program (http://www.ornl.gov/sci/roofs+walls/facts/CoolCalcEnergy.htm)
0 (Reference roof)	1, for higher netting option in net savings ($/ft2 per year) over black roof or Ratio of net savings of lower netting option ($/ft2 per year) divided by net savings of higher netting option ($/ft2 per year), for lower netting option in net savings ($/ft2 per year) over black roof
1	
2	
3	
4	
5	
6	
7	
8	

Table A-3: Approximate Sound Transmission Class (STC)

Roof system type	Numeric rating for acoustic performance
0 (Reference roof)	1 if the reference roof system is a sound barrier, 0 if otherwise
1	0
2	0
3	0.25
4	0.5
5	0.75
6	1
7	1
8	1

Table A-4: Surplus dead load of roof system

Roof system type	Numeric rating for surplus dead load of roof system	Estimated dead load of roof system
0 (Reference roof)	1 if allowable dead load – Estimated dead load ≥ 0 or 0 if otherwise	User assigned number
1		12 psf or 59 kg/m^2
2		16 psf or 78 kg/m^2
3		27 psf or 130 kg/m^2
4		41 psf or 200 kg/m^2
5		54 psf or 260 kg/m^2
6		68 psf or 330 kg/m^2
7		211 psf or 1030 kg/m^2
8		(>213 psf or >1040 kg/m^2

Table A-5: Potential contribution to LEED certification if Sustainable Site (SS) Credit 2 applies

Roof system type	Numeric rating for Potential contribution to LEED certification	SS Credit 5.1	SS Credit 5.2	SS Credit 6.1 and 6.2	SS Credit 7.2
0 (Reference roof)	(SS Credit 5.1 + SS Credit 5.2 + SS Credit 6.1 and 6.2 + SS Credit 7.2) ÷ 4.8	0	0	0.40 if applies, 0 if otherwise	0
1		0	1 if applies, 0 if otherwise	0.80 if applies, 0 if otherwise	1 if applies, 0 if otherwise
2		0	1 if applies, 0 if otherwise	0.90 if applies, 0 if otherwise	1 if applies, 0 if otherwise
3		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.00 if applies, 0 if otherwise	1 if applies, 0 if otherwise
4		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.10 if applies, 0 if otherwise	1 if applies, 0 if otherwise
5		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.20 if applies, 0 if otherwise	1 if applies, 0 if otherwise
6		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.20 if applies, 0 if otherwise	1 if applies, 0 if otherwise
7		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.40 if applies, 0 if otherwise	1 if applies, 0 if otherwise
8		1 if applies, 0 if otherwise	1 if applies, 0 if otherwise	1.80 if applies, 0 if otherwise	1 if applies, 0 if otherwise

Table A-6: Potential contribution to LEED certification if Sustainable Site (SS) Credit 2 does not apply

Roof system type	Numeric rating for Potential contribution to LEED certification	SS Credit 6.1 and 6.2	SS Credit 7.2
0 (Reference roof)	(Credit 6.1 and 6.2 + SS Credit 7.2) ÷ 4.8	0.40 if applies 0 if otherwise	0
1		0.80 if applies 0 if otherwise	1 if applies 0 if otherwise
2		0.90 if applies 0 if otherwise	1 if applies 0 if otherwise
3		1.00 if applies 0 if otherwise	1 if applies 0 if otherwise
4		1.10 if applies 0 if otherwise	1 if applies 0 if otherwise
5		1.20 if applies 0 if otherwise	1 if applies 0 if otherwise
6		1.20 if applies 0 if otherwise	1 if applies 0 if otherwise
7		1.40 if applies 0 if otherwise	1 if applies 0 if otherwise
8		1.80 if applies 0 if otherwise	1 if applies 0 if otherwise

Appendix B

Appendix B. Previous Throwaway Prototypes

This research can be considered a study in a series initiated by the Center for High Performance Environments (CHPE). This series of research concentrates on the decision making processes involved in the design of high performance buildings. This study comprises the development of a prototype DSS, which extends the findings from Elizabeth J. Grant's (2007) dissertation entitled *A Decision-Making Framework for Vegetated Roofing System Selection*. The DSS prototyping project was initially launched by the author of this dissertation in 2009. The first prototype was developed by the author as a throwaway prototype with an emphasis on the data interoperability between a BIM authoring tool and a DSS. In 2010, the author then developed another throwaway prototype with concentrations on the implementation of the Internet and Web infrastructure and the development of a knowledge-based management subsystem. Although these two prototypes were thrown away rather than evolved into the prototype DSS developed as part of this dissertation, they provide useful background information concerning the technologies, designs, and attributes required for developing the DSS. Such background information is summarized below.

The First Throwaway Prototype

The first throwaway prototype was developed in order to examine the issue of data interoperability between a BIM authoring tool and a decision support system. The BIM authoring tool used in this study was Autodesk's Revit Architecture. The DSS was developed in Microsoft's Excel using Microsoft's Visual Basic for Applications (VBA) as a programming language. The Excel-based DSS was developed for the analysis of the storm water retention capability of green roof systems. Figure B-1 demonstrates the architecture of the first throwaway prototype based on the DSS architecture suggested by Turban et al. (2007).

Figure B-2 demonstrates that the user can interact with the DSS through the user interfaces provided in Excel. Within the input spreadsheet presented in Figure B-2a, the user can choose a building database generated by Revit Architecture, located on the user's local hard drive. After the connection is established, the user can then select a building location from a list of predefined locations. Next, the user can select the green roof which is needed to be analyzed. Finally, the user can choose either to clear all the inputs or to process the calculation. If the user chooses to clear all the inputs in the input spreadsheet, all the inputs will be reset for the user to reenter the analysis information. If the user chooses to process the calculation, the system then begins to perform the storm water retention analysis. When the analysis is complete, the user will be directed to the analysis result spreadsheet shown in Figure B-2b. In the result spreadsheet, the use can modify the analysis numeric inputs using the dialog box shown in Figure

B-2c. This system does not allow the user to save the analysis inputs for later use. However, the user can print the analysis results for future reference.

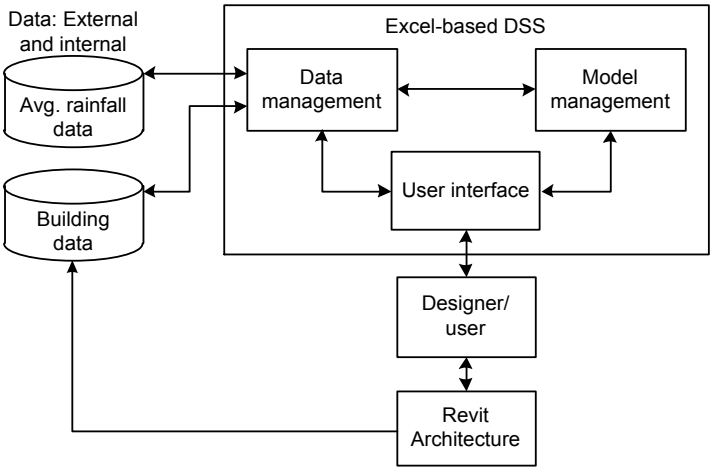
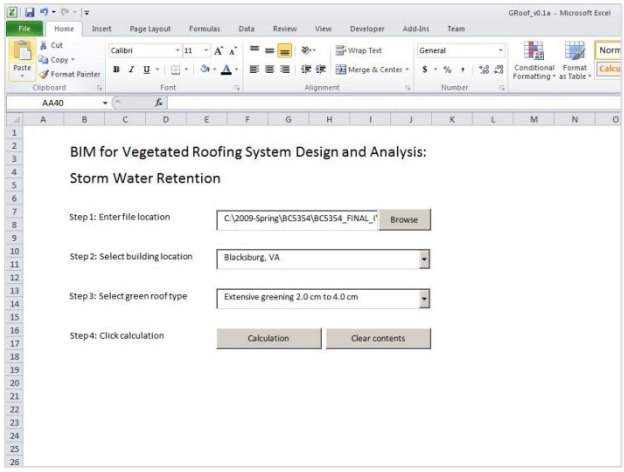
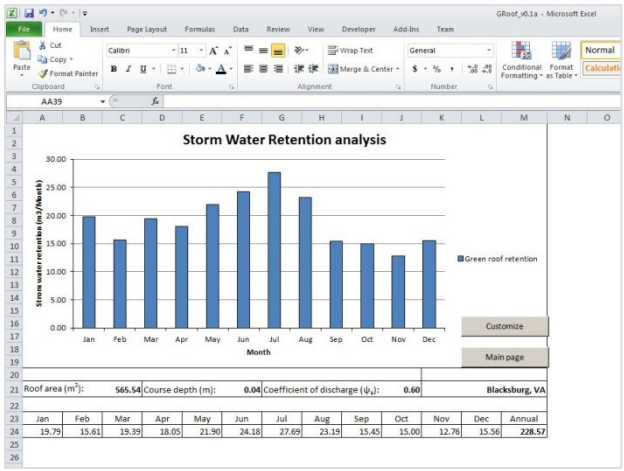


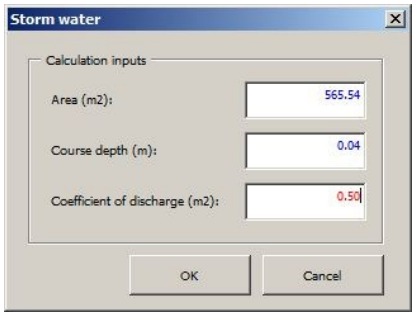
Figure B-1: Excel-based DSS architecture



(a) Input spreadsheet



(b) Analysis result spreadsheet



(c) Modification dialog box

Figure B-2: Interfaces of the Excel-based DSS

In this Excel-based DSS, the model management subsystem manages the model base which contains a calculation method for the calculation of the monthly storm water retention. Equation B-1 represents such a method. It is interesting to note that Equation B-1 has three variables including the roof area, monthly rainfall quantity, and annual coefficient of discharge (ψ_a) of green roof systems. The numeric values of the roof area and monthly rainfall quantity can be queried from the databases through the data management subsystem. In this system, the data management subsystem can interact to the data repositories through the Microsoft's ActiveX Data Objects .NET (ADO.NET) database middleware as shown in Figure B-3. For the annual coefficient of discharge, the numeric values were integrated into the model base using the `Select Case` building block provided in the VBA editor as presented in Box B-1. A particular value of the annual coefficient of discharge can be selected from virtue of the depth of its growing medium. In Equation B-1, the value of 1 subtracted by the annual coefficient of discharge (1 - ψ_a) for a particular generic green roof type can be found in Appendix A.

$$Monthly\ rainfall\ collection = Roof\ area\ (sq.ft.) \times Monthly\ rainfall\ (inches) \times (1 - \psi_a)$$ Equation B-1

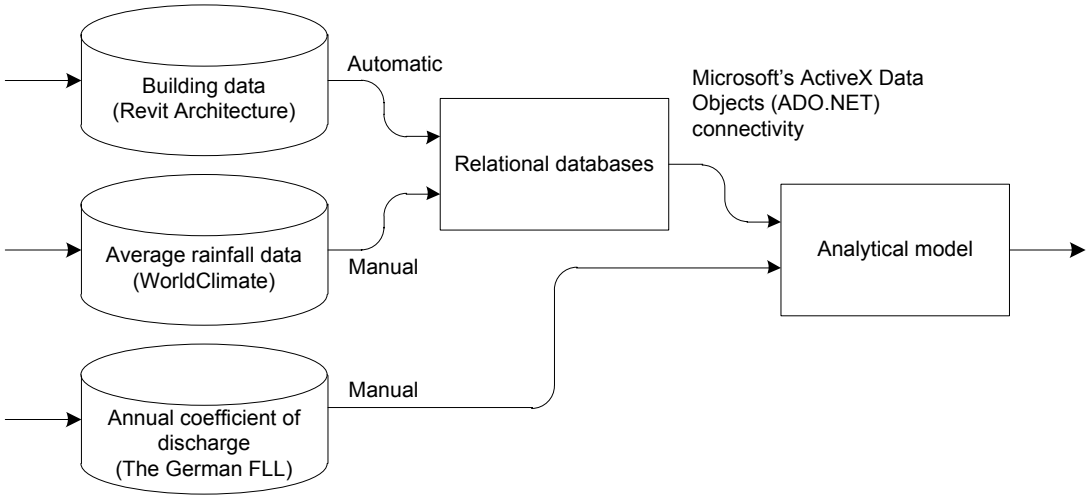


Figure B-3: Data flow diagram demonstrates how the values of three variables are provided for the analytical model

Box B-1: Example of `Select Case` building block

```
Select Case depth
   Case Is < 0.02
      strSQL = "SELECT * FROM FLL_Metric WHERE ID = 1"
   Case Is >= 0.02, Is < 0.04
      strSQL = "SELECT * FROM FLL_Metric WHERE ID = 1"
   Case Is >= 0.04, Is < 0.06
      strSQL = "SELECT * FROM FLL_Metric WHERE ID = 2"
   ...
   Case Else
      strSQL = "SELECT * FROM FLL_Metric WHERE ID = 8"
End Select
```

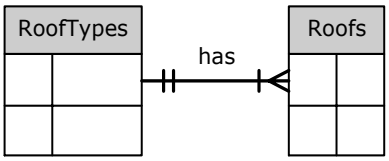


Figure B-4: Entity relationship model of the ROOFTYPES and ROOFS tables using Crow's Foot notation

In addition to the ER model, a relational schema of the RoofTypes and Roofs tables can be demonstrated in Box B-2. The relational schema can be seen as a textual representation of the two database tables whereby each entity in the RoofTypes and Roofs tables has certain attributes described in the parentheses. As shown in the schema, Id is a primary key that identifies each entity in each particular table. The primary key is responsible for maintaining the entity integrity of a database table; as a result, it cannot not be null and must be unique. It is also interesting to mention that TypeId is a foreign key in the Roofs table. In general, a foreign key must be a primary key in a particular table. The foreign key can also be one of the attributes in other tables. In this case, TypeId is a primary key in the RoofTypes table and a foreign key in the Roofs table. TypeId is responsible for maintaining the referential integrity between the RoofTypes and Roofs tables.

Box B-2: Relational schema of the RoofTypes and Roofs tables

```
RoofTypes(Id, Keynote, Model, Manufacturer, TypeComments, URL,
        Description, AssemblyCode, FamilyName, TypeName, TypeMark, Cost)

Roofs(Id, TypeId, PhaseCreated, PhaseDemolished, DesignOption, Volume,
        Area, Comments, RafterorTruss, FascialDepth, RafterCut,
        BaseLevel, CutoffOffset, CutoffLevel, BaseOffsetFromLevel, Mark)
```

Based on the relational schema demonstrated in Box B-2, the model management subsystem can query the area and volume data from the building database through the data management subsystem. The area of green roofs that share a similar green roof type can be directly used in Equation B-1. The volume of green roofs can be used to determine the depth of green roofs by dividing the volume by the area of green roofs. The depth of green roofs then can be used to identify the numeric value of the annual coefficient of discharge required in Equation B-1. According to this relational schema, a compromise needs to be made. That is the green roofs modeled in Revit Architecture must represent only the growing medium layer of the green roof not the whole assembly of the green roofs.

In addition to the building database, the rainfall database contains the average rainfall data which were retrieved from WorldClimate at www.worldclimate.com and input into the database. The monthly average rainfall data can be directly used in Equation B-1. Box B-3 provides the relational schema of the rainfall database which contains only one table. Because WorldClimate provides the rainfall data in millimeters and inches, the AvgRianfall table was designed to use Location and Unit as a composite key to uniquely identify each entity in the table.

Box B-3: Relational schema of the rainfall database

```
AvgRainfall(Location, Unit, Jan, Fab, Mar, Apr,
        May, Jun, July, Aug, Sept, Oct, Nov, Dec)
```

The lessons learned from the first throwaway prototype suggest that there is a certain degree of interoperability between a BIM authoring tool and a DSS. In this case, Revit Architecture has a capability to generate relational databases that contain data about buildings. An Excel-based DSS developed for the analysis of the storm water retention of green roof systems also has capabilities to connect to such databases, analyze the data contained in the databases, and inform the decision maker or designer through the analysis results. Nevertheless, it is important to note that the analysis of the storm water retention capability of green roof systems does not represent the primary feature of the prototype DSS developed as part of this dissertation, which is the vegetated roofing system selection capability discussed in Chapter 2.3. Thus, the next prototyping process should begin to give emphasis to the primary feature of the DSS. Furthermore, the attributes of RoofTypes and Roofs entities apparently lack of information required in Grant's framework for vegetated roofing system selection. Therefore, the data provided in BIM models needs to be further investigated. While the first throwaway prototype can be developed with a capability to connect to relational databases generated using a BIM authoring tool, it can be used only by users who have spreadsheet applications that support MS Excel Macro-Enabled files (.xlsm). This limitation raised a concern over how it should be distributed to a large population of users. Therefore, the next prototype should be developed in the form of a web-based DSS to overcome this issue. The next part of this document will discuss the second throwaway prototype developed by the author.

The Second Throwaway Prototype

The second throwaway prototype was developed in the form of a web-based DSS with a concentration on the interconnections between the data, model, user interface, and knowledge-based management subsystems built upon the Internet and Web infrastructure. Figure B-5 demonstrates the overall architecture of the prototype based on the generic web-based DSS architecture suggested by Power (2000). This generic architecture was rearranged to match the web-based application architecture defined by the Microsoft's ASP.NET framework used for DSS prototyping. In addition to the interconnections of the DSS components, the development of this prototype also concentrated on the development of an intelligent system in the form of an expert system, which was omitted in the development of the first throwaway prototype. For this throwaway prototype, the intelligent system represented the first step of the Decision-Making phase of the CBA tabular method, which included capturing the attributes of particular roofing systems in relation to decision-making factors, and displaying such attributes in the CBA table. It is important to note that this prototyping process largely put emphasis on the internal mechanism of the DSS rather than the issues concerning the data interoperability between BIM authoring tools and DSS. As

a result, this prototype is incapable of utilizing building data contained in any BIM model. Furthermore, in order to reduce the degree of complexity, the development of this prototype considered only four decision-making factors involved in the vegetated roofing system selection including the storm water flow reduction, potential energy savings, approximate sound transmission class, and surplus dead load of roof system.

As shown in Figure B-5, the user can interact with the DSS via the major Web browsers such as Microsoft's Internet Explorer, Mozilla's Firefox, and Google's Chrome. When the user enters the main page of the website as shown in Figure B-6a, the user can choose between creating a new project and working on the saved projects. If the user chooses to create a new project, the user will be directed to the input page, as shown in Figure B-6b, which allows the user to input the attributes of a reference roof system. The reference roof can be compared with up to three generic green roof systems. A particular generic green roof system can be simply selected from the drop down lists. When a particular green roof system is selected, the attributes of such a system will be automatically input into the input form. Within the input form, the user is also allowed to change each attribute of a green roof system if the user has a specific value for such an attribute. After the user is satisfied with the inputs, the user then can choose to process the decision. Subsequently, the user will be directed to the result page which is the same page that the user will be directed to if the user chooses to work on the saved projects on the main page. The result page demonstrated in Figure B-6c displays the first step of the Decision-Making phase of the CBA tabular process. The result page also provides a table that summarizes the attribute values that were input on the input page. On this page, the user can also choose to edit the inputs for a project, update the inputs for a project, add a project, or delete a project.

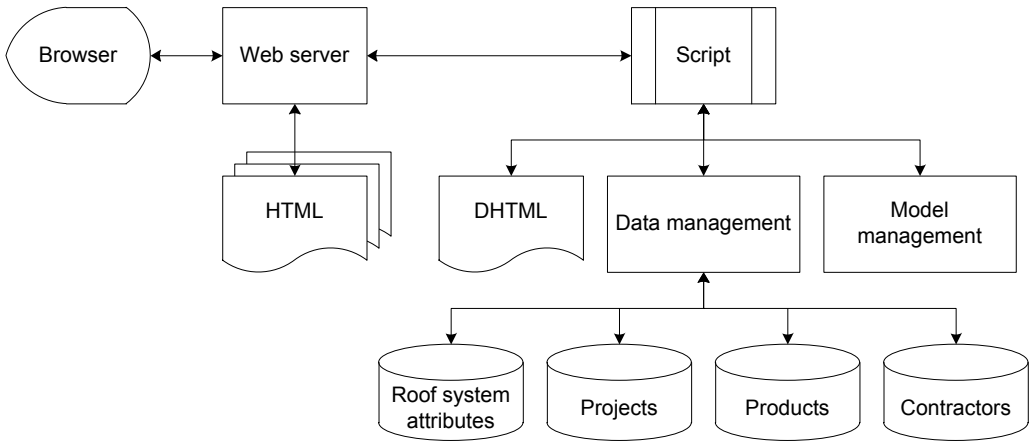
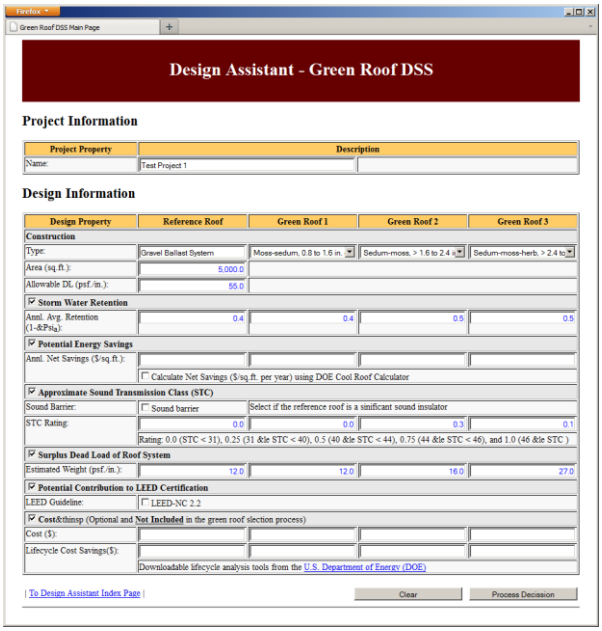
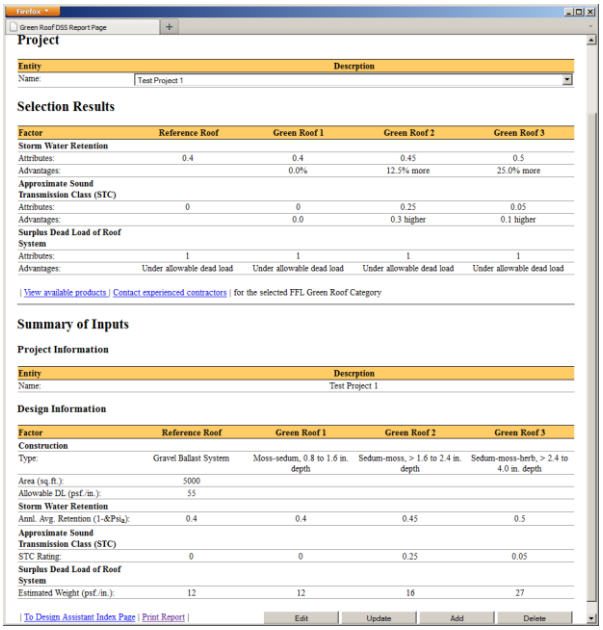


Figure B-5: Web-based DSS architecture

a) Main page



b) Input page



c) Result page

d) Product page

e) Contractor page

Figure B-6: User interfaces of the web-based DSS

In addition to the vegetated roofing system selection feature, this prototype also has an additional data mining capability which can be value-added to the DSS. This secondary feature becomes available after the decision on the roof systems was made. As shown in Figure B-6c, the user can choose either to visit the product page or the contractor page. The product page demonstrated in Figure B-6d provides a list of products that fall within the chosen category of roof types. For example, if the CBA process suggests that Green Roof Type 1 is the most appropriate for the project, the product list will provide the green roof products that have the substrate depth ranging from 0.8 inches to 1.6 inches (2.0 centimeters to 4.0

centimeters) with the moss-sedum plant type for the user. Similarly, the contractor page demonstrated in Figure B-6e provides a list of contractors who have considerable experience with each particular green roof type. While the primary feature of the DSS has significant potential to serve the main users of the system, which include licensed architects, landscape architects, and roofing consultants or engineers, the secondary feature of the system tends to have considerable potential to serve the extended users of the system, which include roofing contractors, horticulturists, specifiers, developers, and policy makers.

As part of the system, the model management subsystem manages the model base that contains an analytical model and certain routines used for determining the attribute numeric values of a particular alternative based on the information given in Appendix A. For the analytical model, this DSS implements the DOE Cool Roof Calculator model developed by the U.S. Department of Energy (DOE) to determine the potential energy savings of the green roofs compared to a black roof. The net energy savings resulting from the comparison then can be used to determine the attribute numeric value of a particular alternative in the potential energy savings factor. In this prototype, however, this process is not automated. The user still has to input the analysis information into the DOE Cool Roof Calculator, which is embedded in the input page for a particular roof type and then manually input the calculated results to the input area provided for the potential energy savings attribute of each roof system as shown in Figure B-7.

In addition to the analytical model, there are certain routines that serve to provide a mechanism for the CBA tabular process. In this prototype DSS, a routine used for identifying the least-preferred attribute within a particular decision-making factor was developed and included into the model base. In addition, a routine used for determine the advantage in a particular decision-making factor provided by each roof system compared to the least-preferred attribute in such a factor was also provided.

For the data aspect of the system, the data management subsystem manages four databases which contain data about the eight generic green roof systems, design projects, green roof products, and experienced green roof contractors. The relational schemas of these databases are provided in Box B-4. It is important to note that these databases were developed in the form of quick-design databases. Therefore, they still need to be optimized or even redesigned and reconstructed.

As shown in Box B-4, the generic green roof system database has one table, the `GR_SYSTEM` table. This table contains general data about the eight green roof systems such as the green roof category and type. The table also contains the attribute data of such green roof systems in the storm water flow reduction, approximate sound transmission class, and surplus dead load of roof system factors used in the CBA process. The data contained in the `GR_SYSTEM` table for a particular green roof type are automatically assigned as default values in the input form when the user chooses a particular green roof system from each dropdown list provided in the input page. The values input into the input page are then used in the CBA process.

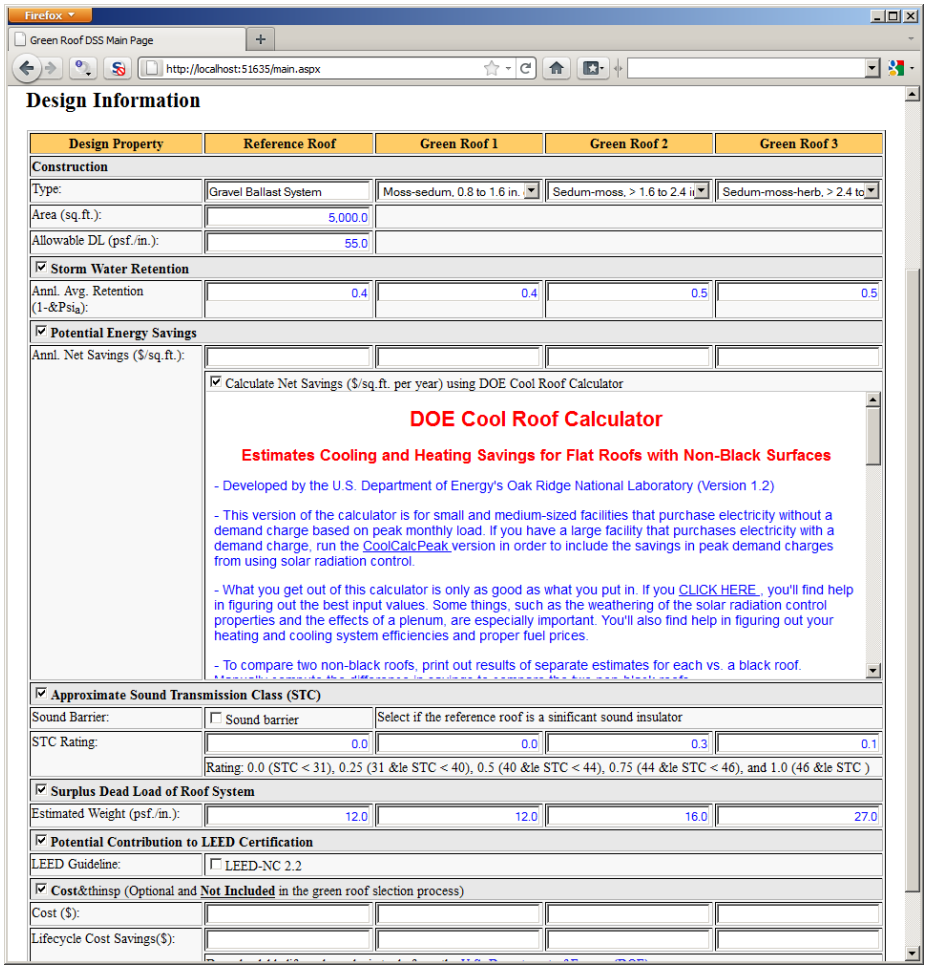


Figure B-7: DOE Cool Roof Calculator embedded in the input interface

As mentioned earlier, this prototype allows the user to save the created projects for later uses. The project data are stored in the design project database which contains two tables: `PROJECT` and `DESIGN`. When the user chooses a particular project from the project dropdown list provided in the result page, the project data will be loaded to the result page from the design project database. The user then can edit the inputs via the input page and update the changed data on the database.

In addition to the generic green roof system and design project database, the green roof product and contractor databases are provided to support the supplementary function of the DSS. The product and contractor data are used after the most appropriate green roof system for a building project is selected as a result of the CBA process. On the result page discussed earlier, the user can choose to view a list of green roof products that share similar characteristics with the selected green roof system. The list of products displays product data retrieved from the green roof product database which contains two tables: `PRODUCT` and `VENDER`. On the result page, the user can also choose to view a list of green roof contractors who have previous experience of the selected green roof system. The list of contractors

displays contractor data retrieved from the green roof contractor database which has only one table, the `CONTRACTOR` table.

Box B-4: Relational schemas of databases

Generic Green Roof Systems

```
GR_SYSTEM(GR_ID, GR_CATEGORY, GR_TYPE, GR_PLANT, GR_SUBDEPTH_CM, GR_SUBDEPTH_IN, GR_AVGRETAIN,
        GR_STCRATE, GR_DLOAD_KGSM, GR_DLOAD_PSF)
```

Design Projects

```
PROJECT(PROJECT_ID, PROJECT_NAME, PROJECT_DESCRIPTION)

DESIGN(DESIGN_ID, PROJECT_ID, DESIGN_ALT, DESIGN_DESCRIPTION, DESIGN_AREA_SQM,
        DESIGN_AREA_SQFT, DESIGN_ALLOAD_KGSM, DESIGN_ALLOAD_PSF, DESIGN_CATEGORY, DESIGN_TYPE,
        DESIGN_PLANT, DESIGN_SUBDEPTH_CM, DESIGN_SUBDEPTH_IN, DESIGN_AVGRETAIN,
        DESIGN_STCRATE, DESIGN_DLOAD_KGSM, DESIGN_DLOAD_PSF)
```

Green Roof Products

```
VENDER(VENDER_ID, VENDER_NAME, VENDER_PHONE, VENDER_URL)

PRODUCT(PRODUCT__ID, VENDER_ID, PRODUCT_NAME, PRODUCT_DESCRIPTION, PRODUCT_CATEGORY,
        PRODUCT_TYPE, PRODUCT_PLANT, PRODUCT_SUBDEPTH_CM, PRODUCT_SUBDEPTH_IN,
        PRODUCT_AVGRETAIN, PRODUCT_STCRATE, PRODUCT_DLOAD_KGSM, PRODUCT_DLOAD_PSF)
```

Green Roof Contractors

```
CONTRACTOR(CONTRACTOR_ID, CONTRACTOR_NAME, CONTRACTOR_ADDRESS, CONTRACTOR_PHONE,
        CONTRACTOR_URL, CONTRACTOR_GRCATEGORY, CONTRACTOR_GRTYPE)
```

The development of the second throwaway prototype indicates that the general-purpose development environments such as .NET Framework can be used to develop a web-based DSS/ES built upon the framework for vegetated roofing system selection. Although the prototype development concentrated on the first step of the Decision-Making phase of the CBA tabular method, it has considerable potential to be developed further in order to provide the full vegetated roofing system selection capability. The prototype development also shows that there is a considerable degree of interconnectivity between the data, model, and user interface subsystems even though these components tend to be located on different computers. In addition to the primary feature of the DSS, the development of this prototype led to the discovery of one possible secondary feature of the system, which is the data mining capability. This additional feature has considerable potential to expand the user base of the system. While this prototyping effort helps increase the understanding about the internal mechanism of the DSS, it is important to note that the developed prototype is still incapable of utilizing building data contained in any BIM model. Therefore, the prototype DSS developed as part of this dissertation should give an emphasis on the issues concerning the data interoperability between BIM authoring tools and the web-based DSS.

Comparative Analysis on Previous Throwaway Prototypes and Conclusion

To better understand what can be learned from two of the throwaway prototypes, a comprehensive comparison between the two systems is provided in Table B-1. As shown in Table B-1, the two throwaway prototypes were developed with different objectives and emphases. Initially, the first throwaway prototype was developed as an Excel-based DSS, because prototyping using Excel has been

recognized as one of the best strategies for prototyping by experts in the field of DSS such as Power (2000) and Turban et al. (2007). In addition, the development of this prototype focused only on exploring the interrelationship between a BIM authoring tool and the DSS by omitting the development of a knowledge-based management subsystem that manages the green roof selection mechanism.

Table B-1: Comparison between the first and second throwaway prototypes

Subject	First Throwaway Prototype	Second Throwaway Prototype
Objective	Develop a prototype of an Excel-based DSS for vegetated roofing system selection	Develop a prototype of a web-based DSS for vegetated roofing system selection
Emphasis	Explore the interrelationship between BIM authoring tools and DSS	Arrange the generic web-based DSS architecture to match the Model-View-Controller (MVC) architecture Develop an intelligent system
Potential user	Licensed architects, landscape architects, and roof consultants or engineers	Licensed architects, landscape architects, and roof consultants or engineers
Function	Help design practitioners select green roof systems for their projects	Help design practitioners select green roof systems for their projects
Feature	Provide decision support information to help decision makers choose green roof systems for their projects based on the storm water retention property of green roof systems Connect to relational databases generated using the BIM authoring tools Print out results for sharing	Provide decision support information to help decision makers choose green roof systems for their projects based on the storm water retention, sound transmission class, and structural surplus dead load properties of green roof systems View results online Print out results for sharing
DSS category	Excel-based, model-driven DSS	Web-based, knowledge-driven DSS
Data management subsystem	DMBS: MS Access and Revit Architecture Database: Relational and object-oriented Data directory: Varied, relatively defined Query facility: SQL	DMBS: MS Access Database: Relational Data directory: App_Data Query facility: SQL
Models management subsystem	MBMS: MS Office VBA and MS Excel Model base: Analytical Modeling language: VBA Modeling directory: COM Structure Storage Model execution, integration, and command processor: Application Visual Machine	MBMS: MS Visual Studio Model base: Analytical Modeling language: VB .NET Modeling directory: App_Code Model execution, integration, and command processor: ASP.NET Compiler Tool
Knowledge-based management subsystem	KBMS: None Intelligent system: None	KBMS: MS Visual Studio Intelligent system: Expert System (ES)
User interface subsystem	DGMS: MS Office VBA and MS Excel Representation: Table and bar chart Operation: Active X controls Memory aid: None Control aid: Excel menus Natural language processor: Standard objects through GUI	DGMS: MS Visual Studio Representation: Table Operation: ASP.NET form controls Memory aid: Input descriptions, mouse over messages, and error messages Control aid: Web browser menus Natural language processor: Standard objects through GUI
Compromise	Modeling method using the BIM authoring tools	Incomplete inference engine No graphic representations for results
Cost	Prototyping: N/A Adoption: N/A	Prototyping: N/A Adoption: N/A

While the first throwaway prototype can be developed with a capability to connect to relational databases generated using a BIM authoring tool, it can be used only by users who have spreadsheet applications that support MS Excel Macro-Enabled files (.xlsm). This limitation raised a concern over how the system should be distributed to a large population of users. Therefore, the second throwaway prototype was developed in the form of a web-based DSS, which seems not to have any significant compatibility and distribution problems in real world applications. However, in order to develop the prototype, the generic web-based DSS architecture needs to be adjusted to match the architecture defined by the web-based

application development framework used for DSS prototyping. This concern had become an emphasis of the second throwaway prototype development. In addition, the development of a knowledge-based management subsystem that was omitted in the development of the first throwaway prototype had become another emphasis of this prototype development.

As presented in Table B-1, although both throwaway prototypes have different objectives and emphases, they were developed for the same group of users initially defined by Grant (2007) as a target group of the framework for vegetated roofing system selection. The potential users in this group include licensed architects, landscape architects, and roof consultants or engineers. In addition to the potential users, both throwaway prototypes were developed to perform a similar function, which is to help design practitioners select appropriate green roof systems for their projects as suggested by Grant. However, with a similar function, both throwaway prototypes were developed with different features influenced by the objectives and emphases of each development effort. These features can be used to help formulate an initial set of features of the prototype DSS developed as part of this study.

In conclusion, it can be considered that the previous DSS prototyping efforts have provided certain useful background information for the DSS prototyping process conducted as part of this study. One is that they address the need to examine several issues relating to the data interoperability between BIM authoring tools and decision support systems. Although the development of the first throwaway prototype suggested that Excel-based DSS can utilize data contained in BIM models, the BIM models may not provide sufficient data required in the CBA tabular process for vegetated roofing system selection. For this issue, the data contained in BIM models should be studied in detail. Furthermore, although BIM authoring tools such as Revit Architecture can generate relational data models that can be used by various types of DSS, the relational data models are not likely to be used for transporting data across the Internet and Web which is a common technological platform of web-based DSS such as the second throwaway prototype. For this issue, the application of XML data models used for transporting data across the Internet and Web or other data models used by AEC practitioners for exchanging data should be examined. Besides the issues of data interoperability, the previous DSS prototyping efforts, especially the development of the second throwaway prototype, also raise one interesting issue concerning the internal mechanism of the DSS. That is the knowledge-based management subsystem composed of the knowledge-based management software and intelligent system has to be developed and integrated onto the system in order to make the vegetated roofing selection process fully functional. Even though the previous prototypes are unlikely to be evolved into the prototype DSS developed as part of this dissertation, this background information should still be taken into consideration when the new prototype is developed.

Appendix C

Appendix C. IRB Approved Documents

Appendix C provides the Virginia Tech Institution Review Board (IRB) approved documents as follows:

- VT IRB-12-189 New Protocol Approval Letter, approved March 5, 2012 to March 4, 2013
- VT IRB-12-189 Phase 1 Informed Consent, approved March 5, 2012 to March 4, 2013
- VT IRB-12-189 Phase 2 Informed Consent, approved March 5, 2012 to March 4, 2013
- VT IRB-12-189 Continuing Review Approval Letter, approved March 5, 2013 to March 4, 2014
- VT IRB-12-189 Amendment Approval Letter, approved March 5, 2013 to March 4, 2014
- VT IRB-12-189 Phase 1 Informed Consent, approved March 5, 2013 to March 4, 2014
- VT IRB-12-189 Phase 2 Informed Consent, approved March 5, 2013 to March 4, 2014
- VT IRB-12-189 Recruitment Letter
- VT IRB-12-189 White Paper
- VT IRB-12-189 Phase 2 Interview Questions
- VT IRB-12-189 Introductory Questionnaire

VirginiaTech

Office of Research Compliance
Institutional Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, Virginia 24060
540/231-4606 Fax 540/231-0959
e-mail irb@vt.edu
Website: www.irb.vt.edu

MEMORANDUM

DATE: March 5, 2012

TO: James R. Jones, Kongkun Charoenvisal

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires May 31, 2014)

PROTOCOL TITLE: A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

IRB NUMBER: 12-189

Effective March 5, 2012, the Virginia Tech IRB Administrator, Carmen T. Green, approved the new protocol for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at http://www.irb.vt.edu/pages/responsibilities.htm (please review before the commencement of your research).

PROTOCOL INFORMATION:
Approved as: **Expedited, under 45 CFR 46.110 category(ies) 6, 7**
Protocol Approval Date: **3/5/2012**
Protocol Expiration Date: **3/4/2013**
Continuing Review Due Date*: **2/18/2013**
*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:
Per federally regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals / work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?

*Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

cc: File

VIRGINIA POLYTECHNIC AND STATE UNIVERSITY

Informed Consent for Participants

in Research Projects Involving Human Subjects

Title of Project: A BIM Interoperable Web-based DSS for Vegetated Roofing System Selection

Primary Investigator: Professor James R. Jones

Co-Investigator: Kongkun Charoenvisal

I. Purpose of the Research

This study concentrates on the benefits of Building Information Modeling (BIM) and Decision Support Systems (DSS) technologies with regards to knowledge capturing and improving the quality of design decisions within the sustainable architecture design domain. This involves developing a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection and includes interviewing potential users of the system to understand the usefulness of the prototype for improving decisions related to vegetative roofing systems. The anticipated findings of this study include an in-depth understanding of the possibility for developing a BIM-interoperable DSS and the usefulness of the DSS as perceived by design practitioners. The understanding gained from this study will be used to develop guidelines for developing the full version of the system that will be released for public use in the near future. The full version of the DSS is expected to be a platform from which multiple DSS applications can be integrated as a means of supporting various decisions involved in the design of sustainable buildings.

II. Procedure

This study involves two phases. The first phase comprises the development of a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection. The second phase encompasses the interview study which aims to gain an in-depth understanding of the usefulness of the prototype. Research participants including architects, landscape architects, roof consultants, engineers, and/or other professionals who work in the fields of Architecture, Engineering, and Construction (AEC) will be invited to voluntarily participate during the prototyping process and/or during the interview study which will focus on the usefulness of the prototype. This informed consent is provided for the participants who participate in the first phase of this study.

As a research participant in the prototyping process, you will be asked to participate in one or more project meetings in order to establish system requirements and provide feedback on the proposed system's functions and features. The meetings will be conducted by the investigator at Virginia Tech's College of Architecture and Urban Studies. The meetings will be recorded using a digital voice recorder. Each meeting will last one to two hours. The prototyping is anticipated to take place from March to May, 2012.

III. Risks

There is a minimal risk that in the process of establishing system requirements and providing feedback on the proposed system's functions and features, you may be reminded of unpleasant or adversarial project conditions that may lead to difficult emotions or reactions. The investigator agrees to refrain from asking any leading questions that seek to deliberately evoke a potential damaging negative reaction. As a research participant, you have complete freedom to stop the meetings or withdraw from this study at any

point. You will have full access to the meeting transcripts as well as the opportunity to provide feedback to the investigator.

IV. Benefits

No promise or guarantee of benefits are offered from the research group to you unless the full version of a BIM-interoperable, Web-based DSS for vegetated roofing system selection is released for public use. However, the full version of the DSS developed, based on the guidelines generated in part from the outcome of system prototyping and interview study, is expected to be of benefit to the larger design community, and to be of particular and significant interest to architects, landscape architects, and roof consultants or engineers.

V. Extent of Anonymity and Confidentiality

All information collected from you will be confidential. All meetings will be recorded with a digital voice recorder. Transcriptions of voice data will be performed by the investigator. Voice data and transcriptions of meetings will be stored in a secure location by the investigator. A coding system will be used to label the meetings. These materials will only be accessible to the investigator and his advisor.

Voice data and transcriptions will be destroyed when research involving these items is deemed complete by the research group. The investigator will be forced to break confidentiality if any abuse incidents are known or strongly suspected or if you are believed to be a threat to yourself or others.

In certain cases, due to the relatively small size of the vegetated roofing system design community, it may be possible for the reader of the final dissertation or papers generated therefrom to deduce your identity based on your feedback on the requirements, functions, and features of the prototype DSS. This is an unavoidable outcome of research in the design filed, where the identity of the designers of buildings is generally considered to be public knowledge. By agreeing to participate in this study, you consent to accept the risk of this possibility. The investigator agrees to not deliberately divulge your identity without your expressed prior written consent.

VI. Compensation

You will receive no compensation for your participation in this study.

If members of the research group determine that you should seek counseling or medical treatment, a list of local services will be provided.

VII. Freedom to Withdraw

You will have full freedom to stop the meeting or withdraw from this study at any point in the process. You are free to not provide feedback on any requirements, functions, and features of the prototype DSS that you choose.

There may be situations where the investigator may determine that you should not continue to be involved in the study.

VIII. Subject's Responsibilities

I voluntarily agree to participate in this study. I have the following responsibilities:

1. To participate in one or more, one to two hour prototyping project meeting and/or to access web-based survey tools and participate in one or more one-half to one hour recorded interviews, either in person or over the telephone.

2. To provide feedback to the research group as needed.

IX. Subject's Permission

I have read the Consent Form and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent:

_____ Date _____
Subject Signature

Should I have any pertinent questions about this research or its conduct, and research subject's rights, and whom to contact in the event of a research-related injury to the subject, I may contact:

Kongkun Charoenvisal Investigator	(540) 808-7995/kongkun_c@vt.edu Telephone/e-mail
James R. Jones Faculty Advisor	(540) 231-7647/wolverine@vt.edu Telephone/e-mail
Robert P. Schubert Departmental Reviewer/Department Head	(540) 231-5607/silver@vt.edu Telephone/e-mail
David M. Moore Chair, Virginia Tech Institutional Review Board for the Protection of Human Subjects Office of Research Compliance 2000 Kraft Drive, Suite 2000 (0497) Blacksburg, VA 24060	(540) 231-4991/moored@vt.edu Telephone/e-mail

VIRGINIA POLYTECHNIC AND STATE UNIVERSITY

Informed Consent for Participants

in Research Projects Involving Human Subjects

Title of Project: A BIM Interoperable Web-based DSS for Vegetated Roofing System Selection

Primary Investigator: Professor James R. Jones

Co-Investigator: Kongkun Charoenvisal

I. Purpose of the Research

This study concentrates on the benefits of Building Information Modeling (BIM) and Decision Support Systems (DSS) technologies with regards to knowledge capturing and improving the quality of design decisions within the sustainable architecture design domain. This involves developing a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection and includes interviewing potential users of the system to understand the usefulness of the prototype for improving decisions related to vegetative roofing systems. The anticipated findings of this study include an in-depth understanding of the possibility for developing a BIM-interoperable DSS and the usefulness of the DSS as perceived by design practitioners. The understanding gained from this study will be used to develop guidelines for developing the full version of the system that will be released for public use in the near future. The full version of the DSS is expected to be a platform from which multiple DSS applications can be integrated as a means of supporting various decisions involved in the design of sustainable buildings.

II. Procedure

This study involves two phases. The first phase comprises the development of a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection. The second phase encompasses the interview study which aims to gain an in-depth understanding of the usefulness of the prototype. Research participants including architects, landscape architects, roof consultants, engineers, and/or other professionals who work in the fields of Architecture, Engineering, and Construction (AEC) will be invited to voluntarily participate during the prototyping process and/or during the interview study which will focus on the usefulness of the prototype. This informed consent is provided for the participants who participate in the second phase of this study.

As a research participant in the interview study, you will be asked to use the prototype DSS for a short period of time and participate in a face-to-face or phone interview. Unstructured (open ended) interviews will be conducted by the investigator either at Virginia Tech's College of Architecture and Urban Studies or in your office at a mutually agreeable time. These interviews will be prefaced by Web-based survey tools accessed by you at survey.vt.edu, which will give you the opportunity to review the questions and prepare responses. The interviews will be recorded using a digital voice recorder. Interviews will last between one-half and one hour. A follow-up interview or interviews may be requested. You will have complete power to grant or deny this request. No more than three (3) one-hour interview sessions may be requested from you. The interview study is expected to take place from June to August, 2012 and is anticipated to take only a few hours of your time over this period.

III. Risks

There is a minimal risk that in the process of relating your computer-based information systems usage and design experiences, you may be reminded of unpleasant or adversarial project conditions that may

lead to difficult emotions or reactions. The investigator agrees to refrain from asking any leading questions that seek to deliberately evoke a potential damaging negative reaction. As a research participant, you have complete freedom to stop the interview or withdraw from this study at any point. You will have full access to your interview transcripts as well as the opportunity to provide feedback to the investigator.

IV. Benefits

No promise or guarantee of benefits are offered from the research group to you unless the full version of a BIM-interoperable, Web-based DSS for vegetated roofing system selection is released for public use. However, the full version of the DSS developed, based on the guidelines generated in part from the outcome of system prototyping and interview study, is expected to be of benefit to the larger design community, and to be of particular and significant interest to architects, landscape architects, and roof consultants or engineers.

V. Extent of Anonymity and Confidentiality

All information collected from you will be confidential. All interviews will be recorded with a digital voice recorder. Transcriptions of voice data will be performed by the investigator. Voice data and transcriptions of interviews and responses to web-based survey tools, will be stored in a secure location by the investigator. A coding system will be used to label web-based surveys and interviews. These materials will only be accessible to the investigator and his advisor.

Voice data, transcriptions and web-based survey responses will be destroyed when research involving these items is deemed complete by the research group. The investigator will be forced to break confidentiality if any abuse incidents are known or strongly suspected or if you are believed to be a threat to yourself or others.

In certain cases, due to the relatively small size of the vegetated roofing system design community, it may be possible for the reader of the final dissertation or papers generated therefrom to deduce your identity based on your responses to interview questions. This is an unavoidable outcome of research in the design filed, where the identity of the designers of buildings is generally considered to be public knowledge. By agreeing to participate in this study, you consent to accept the risk of this possibility. The investigator agrees to not deliberately divulge your identity without your expressed prior written consent.

VI. Compensation

You will receive no compensation for your participation in this study.

If members of the research group determine that you should seek counseling or medical treatment, a list of local services will be provided.

VII. Freedom to Withdraw

You will have full freedom to stop the interview or withdraw from this study at any point in the process. You are free to not answer any interview questions that you choose.

There may be situations where the investigator may determine that you should not continue to be involved in the study.

VIII. Subject's Responsibilities

I voluntarily agree to participate in this study. I have the following responsibilities:

1. To participate in one or more, one to two hour prototyping project meeting and/or to access web-based survey tools and participate in one or more one-half to one hour recorded interviews, either in person or over the telephone.

2. To provide feedback to the research group as needed.

IX. Subject's Permission

I have read the Consent Form and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent:

_____ Date _____
Subject Signature

Should I have any pertinent questions about this research or its conduct, and research subject's rights, and whom to contact in the event of a research-related injury to the subject, I may contact:

Kongkun Charoenvisal Investigator	(540) 808-7995/kongkun_c@vt.edu Telephone/e-mail
James R. Jones Faculty Advisor	(540) 231-7647/wolverine@vt.edu Telephone/e-mail
Robert P. Schubert Departmental Reviewer/Department Head	(540) 231-5607/silver@vt.edu Telephone/e-mail
David M. Moore Chair, Virginia Tech Institutional Review Board for the Protection of Human Subjects Office of Research Compliance 2000 Kraft Drive, Suite 2000 (0497) Blacksburg, VA 24060	(540) 231-4991/moored@vt.edu Telephone/e-mail

Office of Research Compliance
Institutational Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, VA 24060
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE: February 18, 2013

TO: James R Jones, Kongkun Charoenvisal

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires May 31, 2014)

PROTOCOL TITLE: A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

IRB NUMBER: **12-189**

Effective February 14, 2013, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the Continuing Review request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Expedited, under 45 CFR 46.110 category(ies) 6,7**
Protocol Approval Date:	**March 5, 2013**
Protocol Expiration Date:	**March 4, 2014**
Continuing Review Due Date*:	**February 18, 2014**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

Office of Research Compliance
Institutational Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, VA 24060
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE: March 13, 2013

TO: James R Jones, Kongkun Charoenvisal

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires May 31, 2014)

PROTOCOL TITLE: A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

IRB NUMBER: **12-189**

Effective March 13, 2013, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the Amendment request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Expedited, under 45 CFR 46.110 category(ies) 6,7**
Protocol Approval Date:	**March 5, 2013**
Protocol Expiration Date:	**March 4, 2014**
Continuing Review Due Date*:	**February 18, 2014**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

VIRGINIA POLYTECHNIC AND STATE UNIVERSITY

Informed Consent for Participants

in Research Projects Involving Human Subjects

Title of Project: A BIM Interoperable Web-based DSS for Vegetated Roofing System Selection

Primary Investigator: Professor James R. Jones

Co-Investigator: Kongkun Charoenvisal

I. Purpose of the Research

This study concentrates on the benefits of Building Information Modeling (BIM) and Decision Support Systems (DSS) technologies with regards to knowledge capturing and improving the quality of design decisions within the sustainable architecture design domain. This involves developing a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection and includes interviewing potential users of the system to understand the usefulness of the prototype for improving decisions related to vegetative roofing systems. The anticipated findings of this study include an in-depth understanding of the possibility for developing a BIM-interoperable DSS and the usefulness of the DSS as perceived by design practitioners. The understanding gained from this study will be used to develop guidelines for developing the full version of the system that will be released for public use in the near future. The full version of the DSS is expected to be a platform from which multiple DSS applications can be integrated as a means of supporting various decisions involved in the design of sustainable buildings.

II. Procedure

This study involves two phases. The first phase comprises the development of a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection. The second phase encompasses the interview study which aims to gain an in-depth understanding of the usefulness of the prototype. Research participants including architects, landscape architects, roof consultants, engineers, and/or other professionals who work in the fields of Architecture, Engineering, and Construction (AEC) will be invited to voluntarily participate during the prototyping process and/or during the interview study which will focus on the usefulness of the prototype. This informed consent is provided for the participants who participate in the first phase of this study.

As a research participant in the prototyping process, you will be asked to participate in one or more project meetings in order to establish system requirements and provide feedback on the proposed system's functions and features. The meetings will be conducted by the investigator at Virginia Tech's College of Architecture and Urban Studies. The meetings will be recorded using a digital voice recorder. Each meeting will last one to two hours. The prototyping is anticipated to take place from March to May, 2012.

III. Risks

There is a minimal risk that in the process of establishing system requirements and providing feedback on the proposed system's functions and features, you may be reminded of unpleasant or adversarial project conditions that may lead to difficult emotions or reactions. The investigator agrees to refrain from asking any leading questions that seek to deliberately evoke a potential damaging negative reaction. As a research participant, you have complete freedom to stop the meetings or withdraw from this study at any

point. You will have full access to the meeting transcripts as well as the opportunity to provide feedback to the investigator.

IV. Benefits

No promise or guarantee of benefits are offered from the research group to you unless the full version of a BIM-interoperable, Web-based DSS for vegetated roofing system selection is released for public use. However, the full version of the DSS developed, based on the guidelines generated in part from the outcome of system prototyping and interview study, is expected to be of benefit to the larger design community, and to be of particular and significant interest to architects, landscape architects, and roof consultants or engineers.

V. Extent of Anonymity and Confidentiality

All information collected from you will be confidential. All meetings will be recorded with a digital voice recorder. Transcriptions of voice data will be performed by the investigator. Voice data and transcriptions of meetings will be stored in a secure location by the investigator. A coding system will be used to label the meetings. These materials will only be accessible to the investigator and his advisor.

Voice data and transcriptions will be destroyed when research involving these items is deemed complete by the research group. The investigator will be forced to break confidentiality if any abuse incidents are known or strongly suspected or if you are believed to be a threat to yourself or others.

In certain cases, due to the relatively small size of the vegetated roofing system design community, it may be possible for the reader of the final dissertation or papers generated therefrom to deduce your identity based on your feedback on the requirements, functions, and features of the prototype DSS. This is an unavoidable outcome of research in the design filed, where the identity of the designers of buildings is generally considered to be public knowledge. By agreeing to participate in this study, you consent to accept the risk of this possibility. The investigator agrees to not deliberately divulge your identity without your expressed prior written consent.

VI. Compensation

You will receive no compensation for your participation in this study.

If members of the research group determine that you should seek counseling or medical treatment, a list of local services will be provided.

VII. Freedom to Withdraw

You will have full freedom to stop the meeting or withdraw from this study at any point in the process. You are free to not provide feedback on any requirements, functions, and features of the prototype DSS that you choose.

There may be situations where the investigator may determine that you should not continue to be involved in the study.

VIII. Subject's Responsibilities

I voluntarily agree to participate in this study. I have the following responsibilities:

1. To participate in one or more, one to two hour prototyping project meeting and/or to access web-based survey tools and participate in one or more one-half to one hour recorded interviews, either in person or over the telephone.

2. To provide feedback to the research group as needed.

IX. Subject's Permission

I have read the Consent Form and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent:

_____ Date _____
Subject Signature

Should I have any pertinent questions about this research or its conduct, and research subject's rights, and whom to contact in the event of a research-related injury to the subject, I may contact:

Kongkun Charoenvisal Investigator	(540) 808-7995/kongkun_c@vt.edu Telephone/e-mail
James R. Jones Faculty Advisor	(540) 231-7647/wolverine@vt.edu Telephone/e-mail
Robert P. Schubert Departmental Reviewer/Department Head	(540) 231-5607/silver@vt.edu Telephone/e-mail
David M. Moore Chair, Virginia Tech Institutional Review Board for the Protection of Human Subjects Office of Research Compliance 2000 Kraft Drive, Suite 2000 (0497) Blacksburg, VA 24060	(540) 231-4991/moored@vt.edu Telephone/e-mail

VIRGINIA POLYTECHNIC AND STATE UNIVERSITY

Informed Consent for Participants

in Research Projects Involving Human Subjects

Title of Project: A BIM Interoperable Web-based DSS for Vegetated Roofing System Selection

Primary Investigator: Professor James R. Jones

Co-Investigator: Kongkun Charoenvisal

I. Purpose of the Research

This study concentrates on the benefits of Building Information Modeling (BIM) and Decision Support Systems (DSS) technologies with regards to knowledge capturing and improving the quality of design decisions within the sustainable architecture design domain. This involves developing a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection and includes interviewing potential users of the system to understand the usefulness of the prototype for improving decisions related to vegetative roofing systems. The anticipated findings of this study include an in-depth understanding of the possibility for developing a BIM-interoperable DSS and the usefulness of the DSS as perceived by design practitioners. The understanding gained from this study will be used to develop guidelines for developing the full version of the system that will be released for public use in the near future. The full version of the DSS is expected to be a platform from which multiple DSS applications can be integrated as a means of supporting various decisions involved in the design of sustainable buildings.

II. Procedure

This study involves two phases. The first phase comprises the development of a prototype BIM-interoperable, Web-based DSS for vegetated roofing system selection. The second phase encompasses the interview study which aims to gain an in-depth understanding of the usefulness of the prototype. Research participants including architects, landscape architects, roof consultants, engineers, and/or other professionals who work in the fields of Architecture, Engineering, and Construction (AEC) will be invited to voluntarily participate during the prototyping process and/or during the interview study which will focus on the usefulness of the prototype. This informed consent is provided for the participants who participate in the second phase of this study.

As a research participant in the interview study, you will be asked to use the prototype DSS for a short period of time and participate in a face-to-face or phone interview. Unstructured (open ended) interviews will be conducted by the investigator either at Virginia Tech's College of Architecture and Urban Studies or in your office at a mutually agreeable time. These interviews will be prefaced by Web-based survey tools accessed by you at survey.vt.edu, which will give you the opportunity to review the questions and prepare responses. The interviews will be recorded using a digital voice recorder. Interviews will last between one-half and one hour. A follow-up interview or interviews may be requested. You will have complete power to grant or deny this request. No more than three (3) one-hour interview sessions may be requested from you. The interview study is expected to take place from June to August, 2012 and is anticipated to take only a few hours of your time over this period.

III. Risks

There is a minimal risk that in the process of relating your computer-based information systems usage and design experiences, you may be reminded of unpleasant or adversarial project conditions that may

lead to difficult emotions or reactions. The investigator agrees to refrain from asking any leading questions that seek to deliberately evoke a potential damaging negative reaction. As a research participant, you have complete freedom to stop the interview or withdraw from this study at any point. You will have full access to your interview transcripts as well as the opportunity to provide feedback to the investigator.

IV. Benefits

No promise or guarantee of benefits are offered from the research group to you unless the full version of a BIM-interoperable, Web-based DSS for vegetated roofing system selection is released for public use. However, the full version of the DSS developed, based on the guidelines generated in part from the outcome of system prototyping and interview study, is expected to be of benefit to the larger design community, and to be of particular and significant interest to architects, landscape architects, and roof consultants or engineers.

V. Extent of Anonymity and Confidentiality

All information collected from you will be confidential. All interviews will be recorded with a digital voice recorder. Transcriptions of voice data will be performed by the investigator. Voice data and transcriptions of interviews and responses to web-based survey tools, will be stored in a secure location by the investigator. A coding system will be used to label web-based surveys and interviews. These materials will only be accessible to the investigator and his advisor.

Voice data, transcriptions and web-based survey responses will be destroyed when research involving these items is deemed complete by the research group. The investigator will be forced to break confidentiality if any abuse incidents are known or strongly suspected or if you are believed to be a threat to yourself or others.

In certain cases, due to the relatively small size of the vegetated roofing system design community, it may be possible for the reader of the final dissertation or papers generated therefrom to deduce your identity based on your responses to interview questions. This is an unavoidable outcome of research in the design filed, where the identity of the designers of buildings is generally considered to be public knowledge. By agreeing to participate in this study, you consent to accept the risk of this possibility. The investigator agrees to not deliberately divulge your identity without your expressed prior written consent.

VI. Compensation

You will receive no compensation for your participation in this study.

If members of the research group determine that you should seek counseling or medical treatment, a list of local services will be provided.

VII. Freedom to Withdraw

You will have full freedom to stop the interview or withdraw from this study at any point in the process. You are free to not answer any interview questions that you choose.

There may be situations where the investigator may determine that you should not continue to be involved in the study.

VIII. Subject's Responsibilities

I voluntarily agree to participate in this study. I have the following responsibilities:

1. To participate in one or more, one to two hour prototyping project meeting and/or to access web-based survey tools and participate in one or more one-half to one hour recorded interviews, either in person or over the telephone.

2. To provide feedback to the research group as needed.

IX. Subject's Permission

I have read the Consent Form and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent:

_____ Date _____
Subject Signature

Should I have any pertinent questions about this research or its conduct, and research subject's rights, and whom to contact in the event of a research-related injury to the subject, I may contact:

Kongkun Charoenvisal Investigator	(540) 808-7995/kongkun_c@vt.edu Telephone/e-mail
James R. Jones Faculty Advisor	(540) 231-7647/wolverine@vt.edu Telephone/e-mail
Robert P. Schubert Departmental Reviewer/Department Head	(540) 231-5607/silver@vt.edu Telephone/e-mail
David M. Moore Chair, Virginia Tech Institutional Review Board for the Protection of Human Subjects Office of Research Compliance 2000 Kraft Drive, Suite 2000 (0497) Blacksburg, VA 24060	(540) 231-4991/moored@vt.edu Telephone/e-mail

VIRGINIA POLYTECHNIC AND STATE UNIVERSITY

Recruitment Letter

Title of Project: A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

Primary Investigator: Professor James R. Jones

Co-Investigator: Kongkun Charoenvisal

Dear [Recipient Name]

My name is Kongkun Charoenvisal; I am a doctoral candidate in the School of Architecture and Design at Virginia Polytechnic Institute and State University. I am conducting a study as part of the requirements for the degree of Doctor of Philosophy in Architecture and Design Research (ADR), and I would like ask for your participation. You are invited because of your outstanding reputation and expertise in one or more of the following areas: sustainable architecture, integration practice, and building information modeling.

The study that you are invited to participate in is a research project initiated by the Director for the Center of High Performance Environments (CHPE) at Virginia Tech, Dr. Jim Jones, which has a goal to improve the quality of the design process and decisions, involved in the lifecycle of high performance environments. This research comprises the development of a prototype Web-based Decision Support System (DSS) for vegetated roofing system (green roof) selection and an interview study focused on the usefulness of the developed prototype for improving practitioners' decisions related to green roof selection. The understanding gained from this study will be used to develop guidelines for developing the full version of the DSS that will be released in the near future.

Based on the research description above, I have identified you as one of leading practitioners who is involved in the implementation and/or development of Building Information Modeling (BIM) to promote sustainable and integration practices. I would like to request your participation in the prototyping process and/or in the interview study focusing on the usefulness of the developed prototype DSS. For the prototyping process, I anticipate that your involvement will include participating in a project stakeholder meeting and providing feedback on the DSS functions and features. The prototyping process is expected to take place from March to May, 2012. For the interview study, I anticipate the participation will include time to test the prototype DSS in your work environment and participate in a personal and/or phone interview. The interview phase of this study is expected to take place from June to August, 2012. I would estimate that your total participation should not be more than a few hours.

I have enclosed a document which describes the research problem statement, research objectives, participation requests, and participation benefits, via this e-mail. I also would like to thank you very much in advance for your kind consideration in participating in this research, and look forward to hearing from you. It is only through the participation of experts such as yourself that I can fully meet the goals of this work. I may contact you within the next two weeks to answer any questions you may have.

Sincerely,

Kongkun Charoenvisal
Ph.D. Candidate
Architecture and Design Research Program
School of Architecture and Design
Virginia Polytechnic Institute and State University
E-mail: kongkun_c@vt.edu

Enclosure

A BIM Interoperable Web-Based DSS for Vegetated Roofing System Selection

White Paper by
Kongkun Charoenvisal [kongkun_c@vt.edu]
Virginia Polytechnic Institute and State University

Executive Summary

This research is a study initiated by the Center of High Performance Environments (CHPE), Virginia Tech, which concentrates on improving the quality of design processes and decisions involved in the lifecycle of high performance environments. This research encompasses developing a prototype version of a decision support system for vegetated roofing system selection and studying the usefulness of the developed prototype for improving practitioners' decisions. The knowledge gained from this study will be summarized in the form of guidelines for developing the full version of the system that will be released to the public domain in the near future. In order to ensure that the fully released version of the system will be able to fulfill the needs of practitioners, potential users of the system including architects, landscape architects, and roof consultants or engineers are invited to participate in this study.

Research Problems

After a decade since the term "Building Information Modeling (BIM)" was coined, BIM has gained growing interest by Architecture, Engineering, and Construction (AEC) practitioners for decision support in addition to construction-level modeling purposes. While BIM is beginning to be used for informing practitioners' decisions in the AEC fields, Decision Support Systems (DSS) seem to be one of the Computer-Based Information Systems (CBIS) that have been maturely used in other fields, especially in engineering and business practices but not directly in AEC. Based on its theory and current state of technology, DSS has considerable potential to be implemented as a platform for utilizing building information contained in BIM models together with other databases, analytical models, and expert knowledge employed by practitioners to better improve practitioners' decisions. Therefore, the development of a BIM interoperable DSS and the usefulness of such a DSS in improving AEC practitioners' decisions should be examined.

Research Objectives

This study comprises two objectives. The first objective is to develop a prototype BIM interoperable Web-based DSS for vegetated roofing system (green roof) selection. The primary feature of the prototype DSS is to help design practitioners select appropriate green roofs for their projects based on the following five factors: 1) storm water flow reduction, 2) potential energy savings, 3) approximate sound transmission class, 4) surplus dead load of roof system, and 5) potential contribution to LEED certification. The second objective is to study the usefulness of the developed prototype in improving the quality of design decisions. The final outcomes of this study will be used to develop guidelines for the development of the full version of the DSS that will be released in the near future. The full version DSS is expected to be a platform on which multiple DSS applications similar to the DSS for vegetated roofing system selection can be combined to support various decisions involved in the design of high performance environments.

Participation Requests

In order to achieve the research objectives, research participants including architect, landscape architects, and roof consultants or engineers are invited to voluntarily participate during the prototyping process and/or during the interview study focusing on the usefulness of the prototype. The practitioners who participate in the prototyping process are asked to participate in one or more project meetings in order to establish system requirements and provide feedback on the proposed system functions and features. The prototyping is anticipated to take place from March to May, 2012. The practitioners who participate in the interview study are asked to use the prototype DSS for a short period of time and participate in a face-to-face or phone interview. The interview study is expected to take place from June to August, 2012 and is anticipated to take only a few hours of your time over this period.

Participation Benefits

The practitioners who participate in this study will be part of the development of a technology that has a large potential to shift BIM process from the information domain to the knowledge domain. The practitioners will have considerable opportunities to address their needs, problems, and challenges related to the selection of green roof systems as well as to learn more about how the DSS might help them make sound decisions on green roof design. Furthermore, because this study needs your participation and input, the practitioners who participate in this study will have the opportunity to be the first group of authorized users of the full version DSS that will be released in the near future.

Conclusion

This study involves the development of a prototype DSS for vegetated roofing system selection and a qualitative study on the usefulness of the developed prototype in improving practitioners' decisions related to green roof selection. In order to ensure that the developed system will be able to provide considerable benefits for end users, design practitioners are invited to participate in this study.

A BIM INTEROPERABLE WEB-BASED DSS FOR VEGETATED ROOFING SYSTEM SELECTION: PHASE 2 INTERVIEW QUESTIONS

Vegetated Roofing System or Green Roof Design Experiences

To establish a background about the respondent, the roles he/she played in designing green roofs and the tools he/she used to aid in designing green roofs

1. What is the nature of your experience with green roof design?
2. How many green roof projects have you been involved with?
3. What were your responsibilities on the green roof projects you have worked on?
4. What were the tools that you used for helping you make green roof design decisions?

Building Information Modeling (BIM) Experiences

To establish a background about the respondent, the roles he/she played in BIM projects, and the BIM tools/platforms he/she used in previous BIM projects

1. What is the nature of your experience with Building Information Modeling (BIM)?
2. How many BIM projects have you been involved with?
3. What were your responsibilities on the BIM projects you have worked on?
4. What were the BIM tools/platforms that you used in your previous BIM projects?

Prototype BIM-Operable Web-Based DSS for Vegetated Roofing System Selection Experiences

To understand the respondent's perception of the usefulness of the prototype DSS, satisfaction of the prototype DSS functions and features, and expectations for the fully developed DSS version

1. In general, do you think the prototype DSS is useful as a decision support tool for green roof selection? Please explain why?
2. Does the prototype DSS provide useful decision support information concerning green roof selection? Please explain why?
3. Does the prototype DSS seem appealing to you because it can be used in conjunction with BIM authoring tools? Please explain why?
4. Is the prototype DSS useful to you when considering that it can be used on multiple internet assessable devices or operating systems? Please explain why?
5. Is it beneficial that the prototype DSS can produce printable, digital reports, which you can review, printout, or share with your project team? Please explain why?
6. Is it possible for the decision support information provided by the prototype DSS to be involved in legal issues associated with your green roof selection decisions? Please explain why?
7. Do you think the prototype DSS can help you reduce the time normally used for making green roof selection decisions? Please explain why?
8. At what level do you agree with the following statement? Please explain why?

"Having a collection or library of decision support systems, similar to the prototype DSS for vegetated roofing system selection, would positively impact my decision-making and help improve my design process."

9. Are you willing to use the fully developed DSS version, which will provide you with multiple decision support tools, in the near future? Please explain why?
10. What are your expectations for the fully developed DSS version?
11. Design knowledge acquisition tends to be the highest investment cost involved in developing and maintaining knowledge-driven DSS; thus, there are possible chances that the full DSS version will be provided with a yearly subscription fee. What would be the maximum yearly subscription fee in US dollars that you anticipate paying?

A BIM INTEROPERABLE WEB-BASED DSS FOR VEGETATED ROOFING SYSTEM SELECTION: INTRODUCTORY QUESTIONNAIRE

Please answer all questions to the best of your knowledge, indicating “don’t know” or “N/A” where applicable.

Part 1: Research Participant Background

Please enter your name. This information will only be assessable to the researcher and will remain confidential.

Please rate your familiarity with vegetated roofing system or green roof design.

- ○ Not at all familiar
- ○ Slightly familiar
- ○ Moderately familiar
- ○ Very familiar
- ○ Extremely familiar

What is the nature of your experience with green roof design?

How many green roof projects have you been involved with?

What were your responsibilities on the green roof projects you have worked on?

What were the decision-making/design tools that you used in such green roof projects?

Please rate your familiarity with Building Information Modeling (BIM)

○ Not at all familiar ○ Slightly familiar ○ Moderately familiar ○ Very familiar ○ Extremely familiar

What is the nature of your experience with BIM?

How many BIM projects have you been involved with?

What were your responsibilities on the BIM projects you have worked on?

What were the BIM tools that you used on such projects?

Part 2: The Usefulness of the prototype BIM-Interoperable, Web-Based Decision Support System (DSS)

In general, do you think the prototype DSS is useful as a decision support tool for green roof selection?

○ Yes ○ No ○ Not sure

Please explain why?

Does the prototype DSS provide useful decision support information concerning green roof selection?

○ Yes ○ No ○ Not sure

Please explain why?

Does the prototype DSS seem appealing to you when it can be used in conjunction with BIM authoring tools?

○ Yes ○ No ○ Not sure

Please explain why?

Is the prototype DSS useful to you when considering that it can be used on multiple internet assessable devices or operating systems?

○ Yes ○ No ○ Not sure

Please explain why?

Is it beneficial that the prototype DSS can produce printable, digital reports, which you can review, printout, or share with your project team?

○ Yes ○ No ○ Not sure

Please explain why?

Is it possible for the decision support information provided by the prototype DSS to be involved in legal issues associated with your green roof selection decisions?

○ Yes ○ No ○ Not sure

Please explain why?

Do you think the prototype DSS can help you reduce the time normally used for making green roof selection decisions?

○ Yes ○ No ○ Not sure

Please explain why?

At what level do you agree with the following statement?

> "Having a collection or library of decision support systems, similar to the prototype DSS for vegetated roofing system selection, would positively impact my decision-making and help improve my design process."

○ Strongly disagree ○ Disagree ○ Neither agree nor disagree ○ Agree ○ Strongly agree

Please explain why?

Are you willing to use the fully developed DSS version, which will provide you with multiple decision support tools, in the near future?

Yes No Not sure

Please explain why?

What are your expectations for the fully developed DSS version?

Design knowledge acquisition tends to be the highest investment cost involved in developing and maintaining knowledge-driven DSS; thus, there are possible chances that the full DSS version will be provided with a yearly subscription fee. What would be the maximum yearly subscription fee in US dollars that you anticipate paying?

Appendix D. Prototype VRSS DSS

As mentioned in Chapter 4.5, the demonstration website and the prototype Vegetated Roofing System Selection (VRSS DSS) may be available online for a limited time. Therefore, the screenshots of the prototype DSS and the demonstration website are provided here.

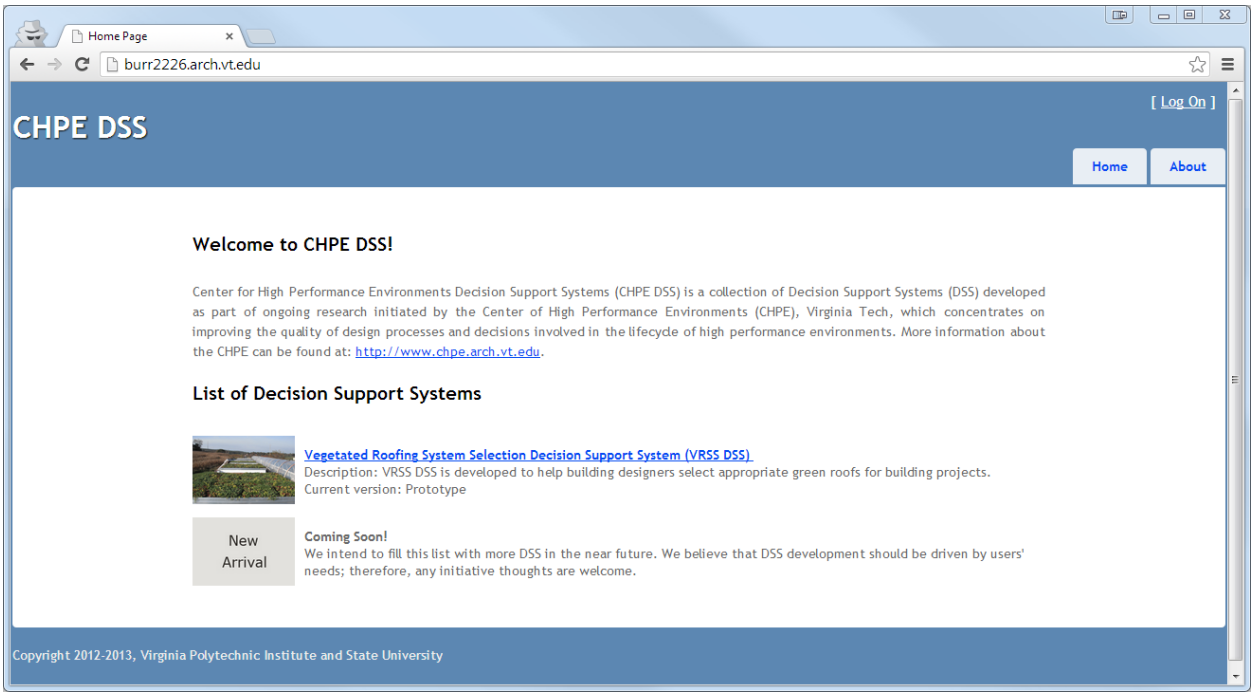


Figure D-1: Home Page

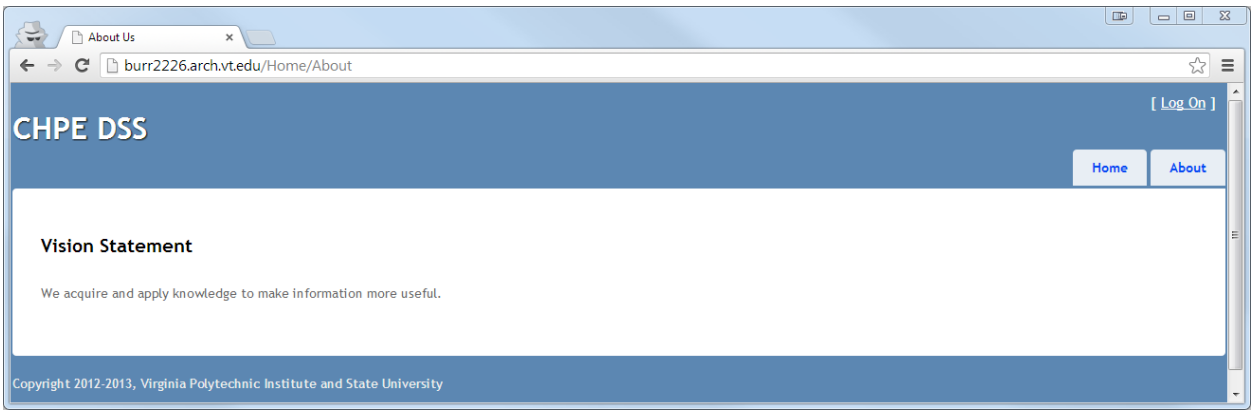


Figure D-2: About Us

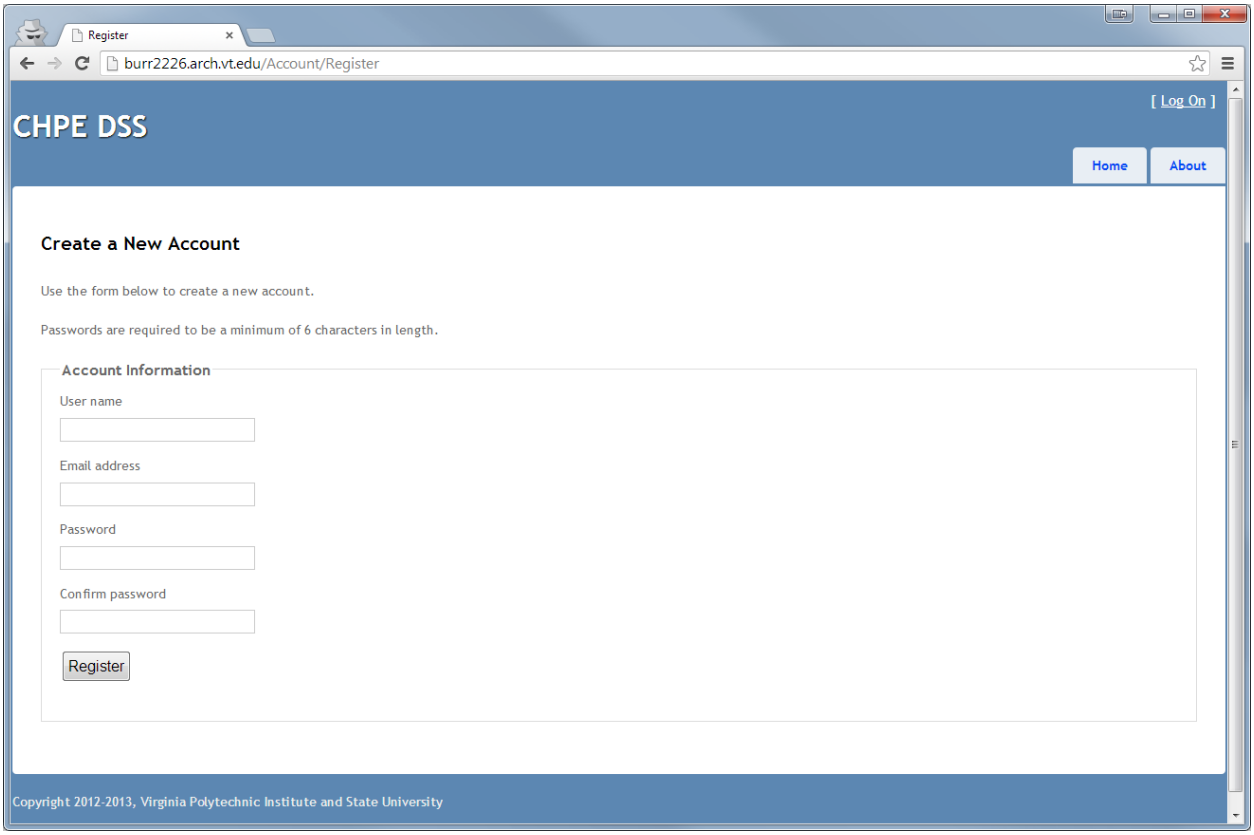


Figure D-3: Register

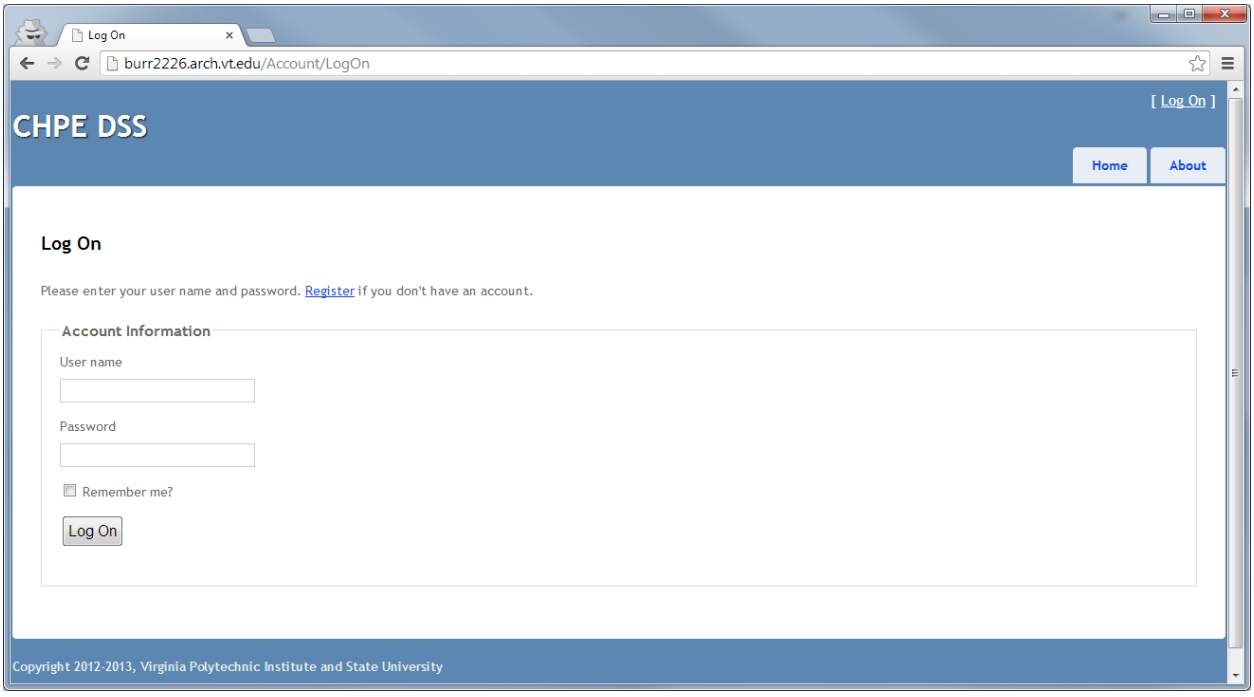


Figure D-4: Log On

Figure D-5: Main Page

Decision-making variable	Acceptable value	Value
Project		
Name	Project name	L-Shape Roof
Reference roof		
Name	Roof name	RT - 101
Description	Roof description	White latex coating w/ highest solar reflectance on 5” foam insulation
Storm water		
Storm water retention		
Annual coefficient of discharge, Ψ_a ($0.0 \leq \Psi_a \leq 1.0$)	0.00 ≤ Value ≤ 1.00	0.8
Energy		
Potential energy savings		
Weather data		
Country	Selected country	United States
State or province	Selected subdivision	Virginia
Location	Selected location	Sterling
Reference Roof Properties		
Area of the proposed roof [sq ft]	Value > 0.00	13000
Reference roof R-value (HIGH=20; AVG=10; LOW=5) [$h \cdot ft^2 \cdot °F/Btu$]	5.00 ≤ Value ≤ 32.00	32
Reference roof infrared emittance, IE (HIGH=90; AVG=60; LOW=10) [%]	5.00 ≤ Value ≤ 95.00	90
Reference roof solar reflectance, SR (HIGH=80; AVG=50; LOW=10) [%]	5.00 ≤ Value ≤ 85.00	70
Equipment Efficiencies		
Air conditioner efficiency (Coefficient of Performance) (HIGH=2.5; AVG=2.0; LOW=1.5)	Value > 1.00	2
Heating system	Selected system	Heat Pump
Heating efficiency of the selected heating system	Value > 1.00	3
Energy Costs		
Summertime cost of electricity (HIGH=0.20; AVG=0.10; LOW=0.05) [$/KWh]	Value ≥ 0.00	0.25
Energy source for heating	Selected source	Electricity
Wintertime cost of the selected energy source	Value ≥ 0.00	0.25
Acoustics		
Approximate Sound Transmission Class		
Is "Reference Roof" a sound barrier?	Yes/No	Yes
Approximate sound transmission class, STC (POOR=25; FAIR=30; GOOD=35; VERY GOOD=42-45; EXCELLENT=46-50)	Value > 0	46
Surplus dead load of roof system		
Allowable dead load of roof structure [psf]	Value > 0.00	30
Reference roof maximum dead load [psf]	Value > 0.00	15
Potential contribution to LEED certification		
Is LEED-NC Sustainable Sites (SS) Credit 2 applied?	Yes/No	Yes
Sustainable Site (SS) Credit 5.1: Site Development: Protect or Restore Habitat	Check/Uncheck	Check
Sustainable Site (SS) Credit 5.2: Site Development: Maximize Open Space	Check/Uncheck	Check
Sustainable Site (SS) Credit 6.1 and 6.2: Stormwater Design: Quantity and Quality Control	Check/Uncheck	Check
Sustainable Site (SS) Credit 7.2: Heat Island Effect: Roof	Check/Uncheck	Check

Figure D-6: Downloadable Input Worksheet

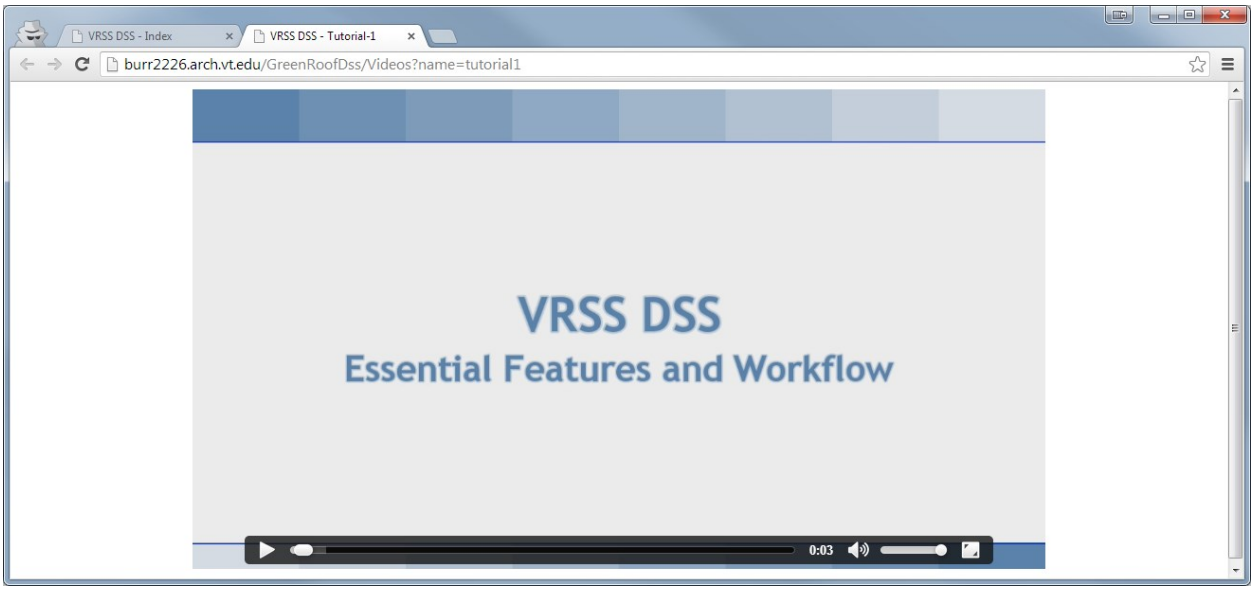


Figure D-7: Tutorial video

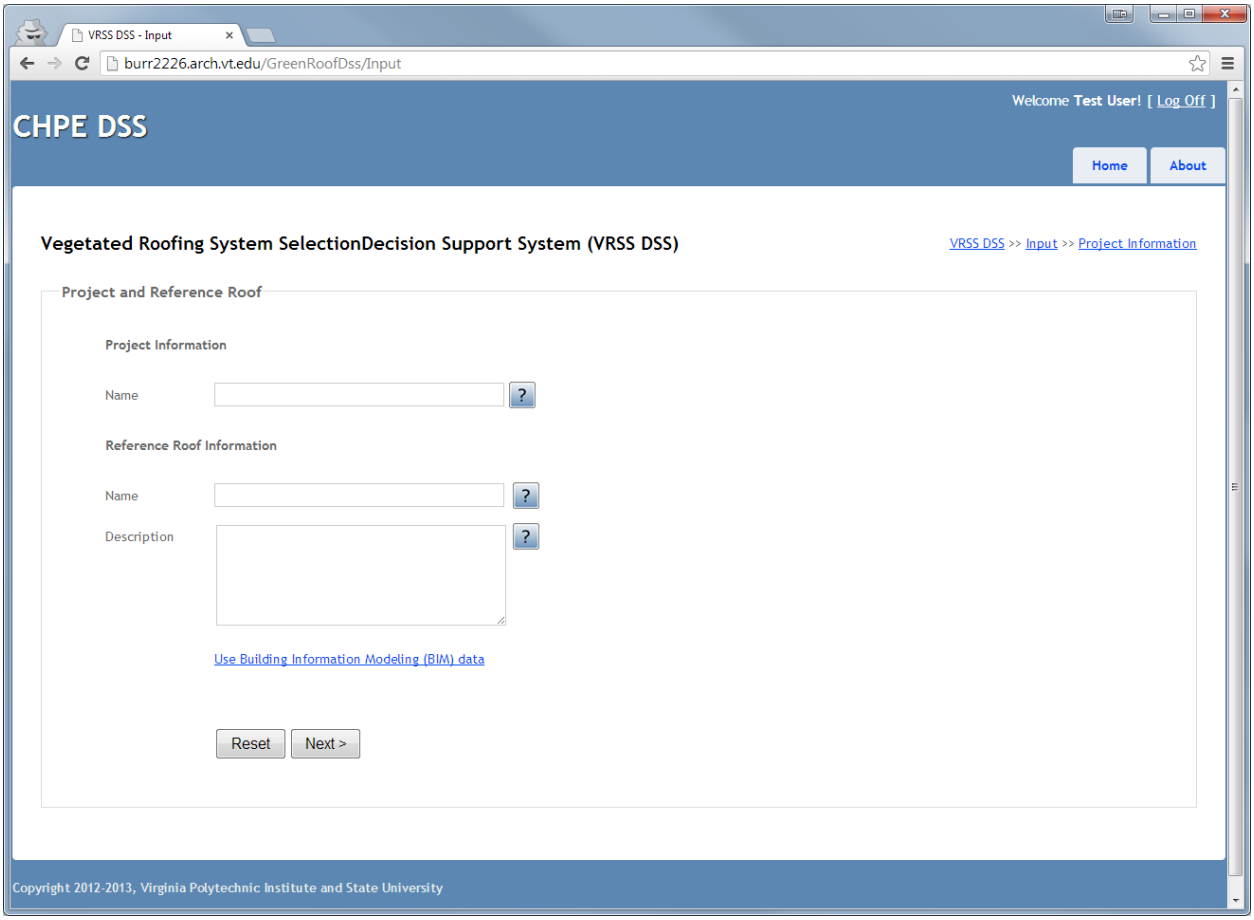


Figure D-8: Project and Reference Roof Information (Manual)

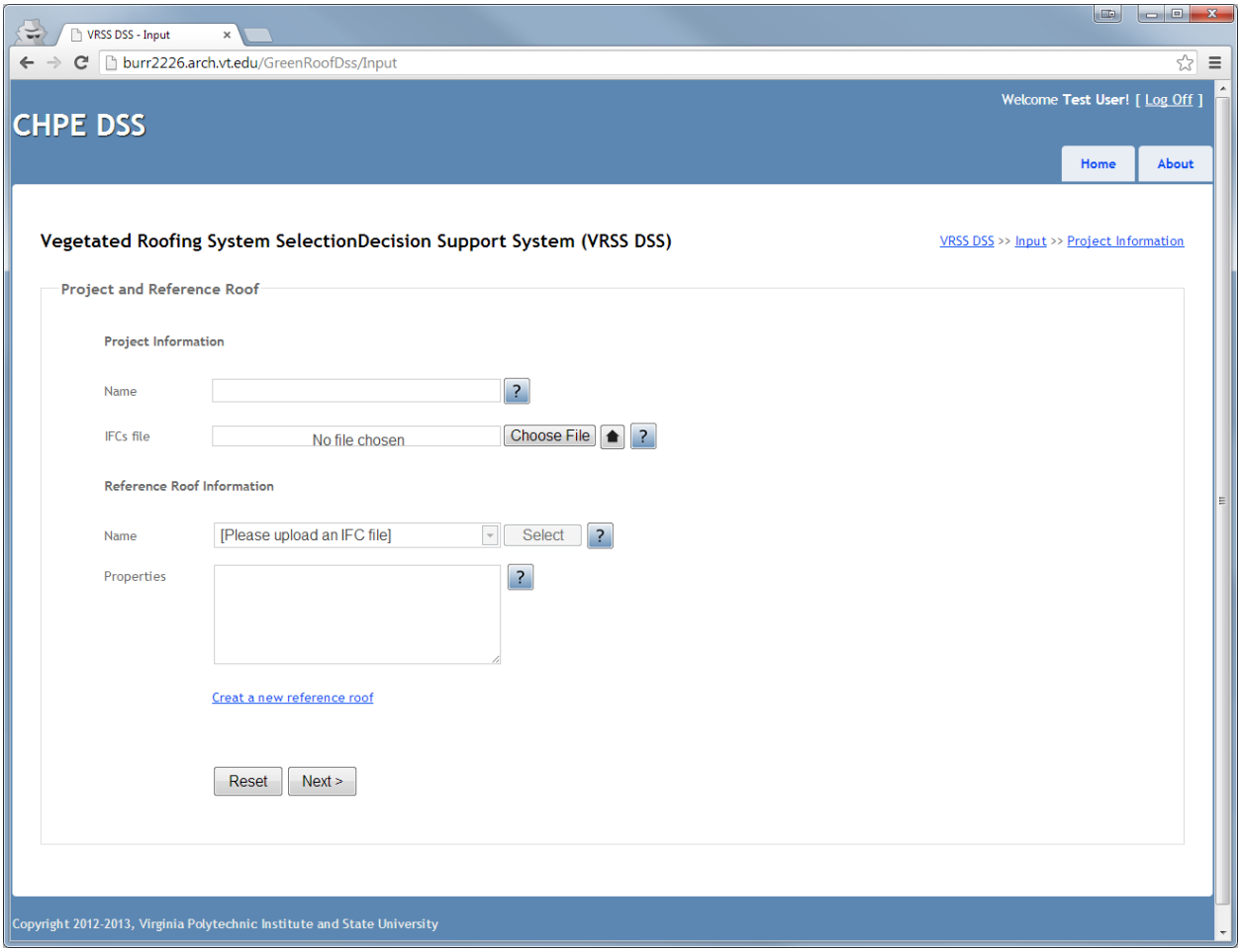


Figure D-9: Project and Reference Roof Information (IFC)

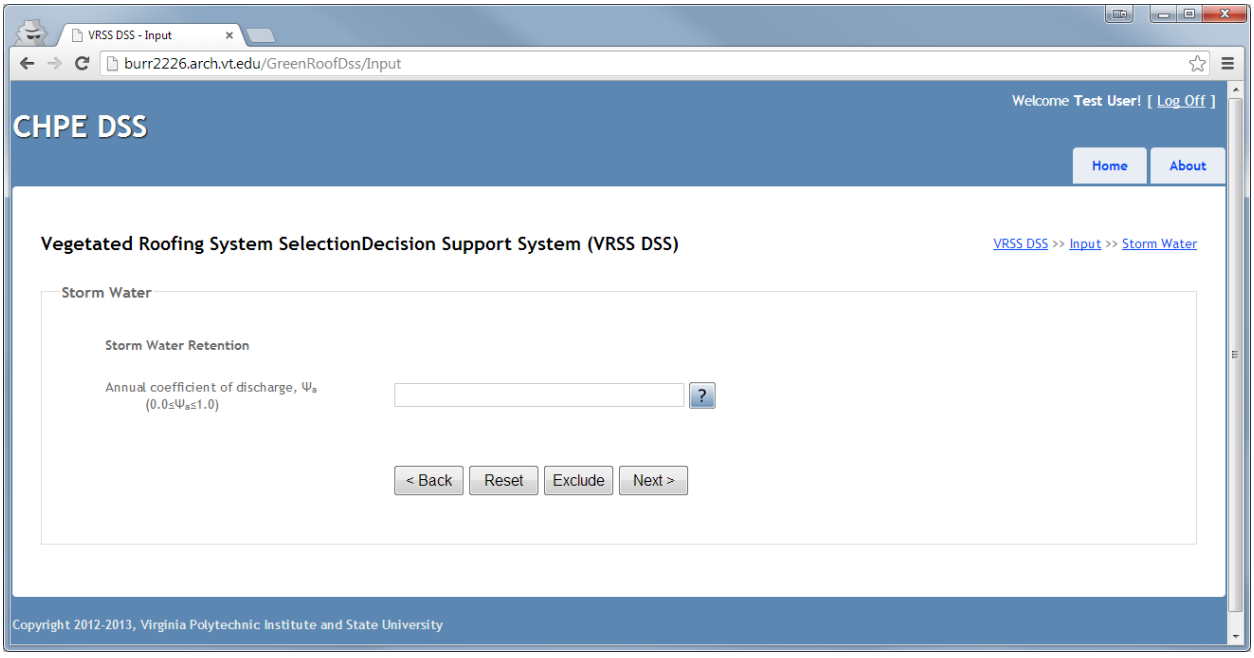


Figure D-10: Storm Water Retention

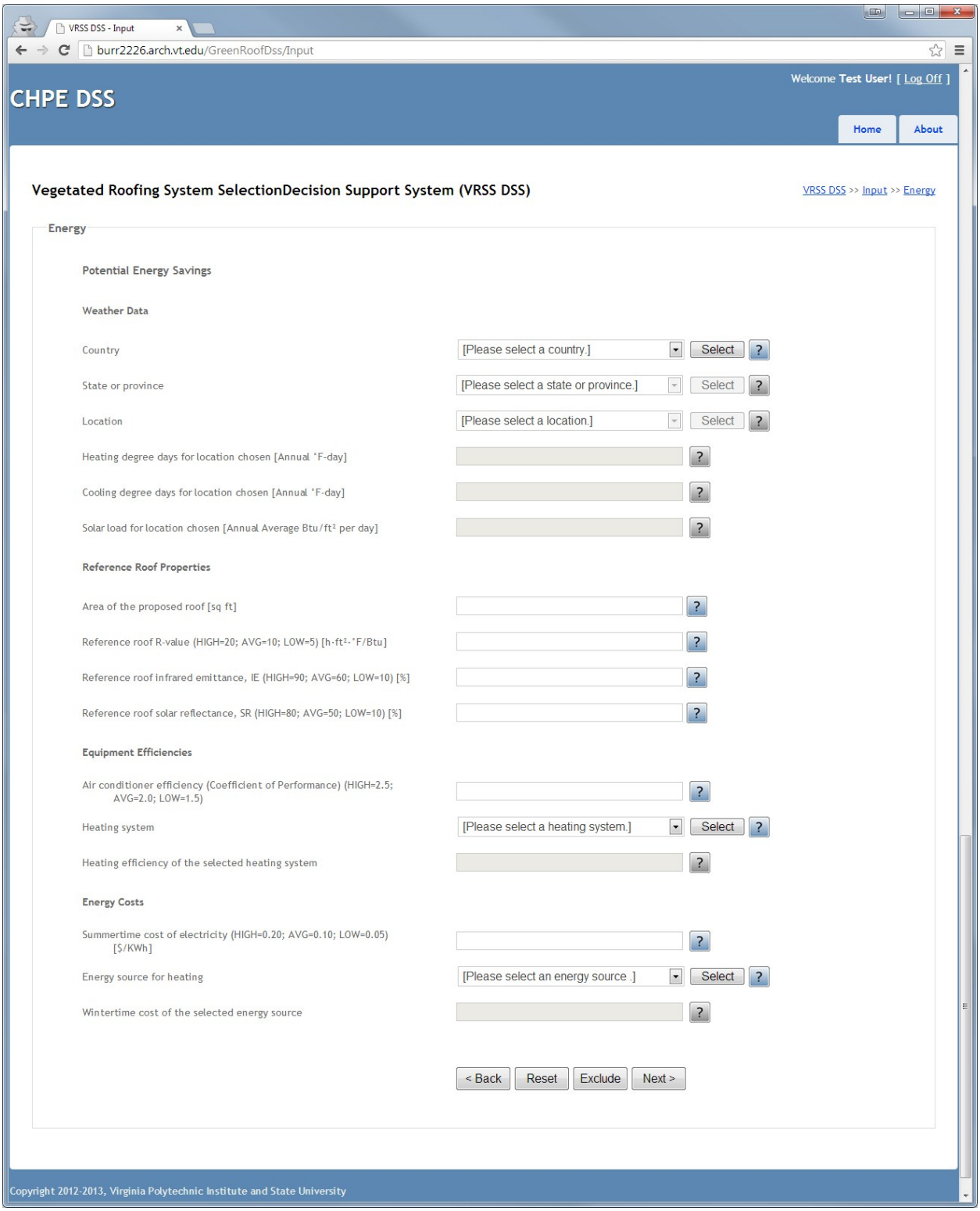


Figure D-11: Potential Energy Savings

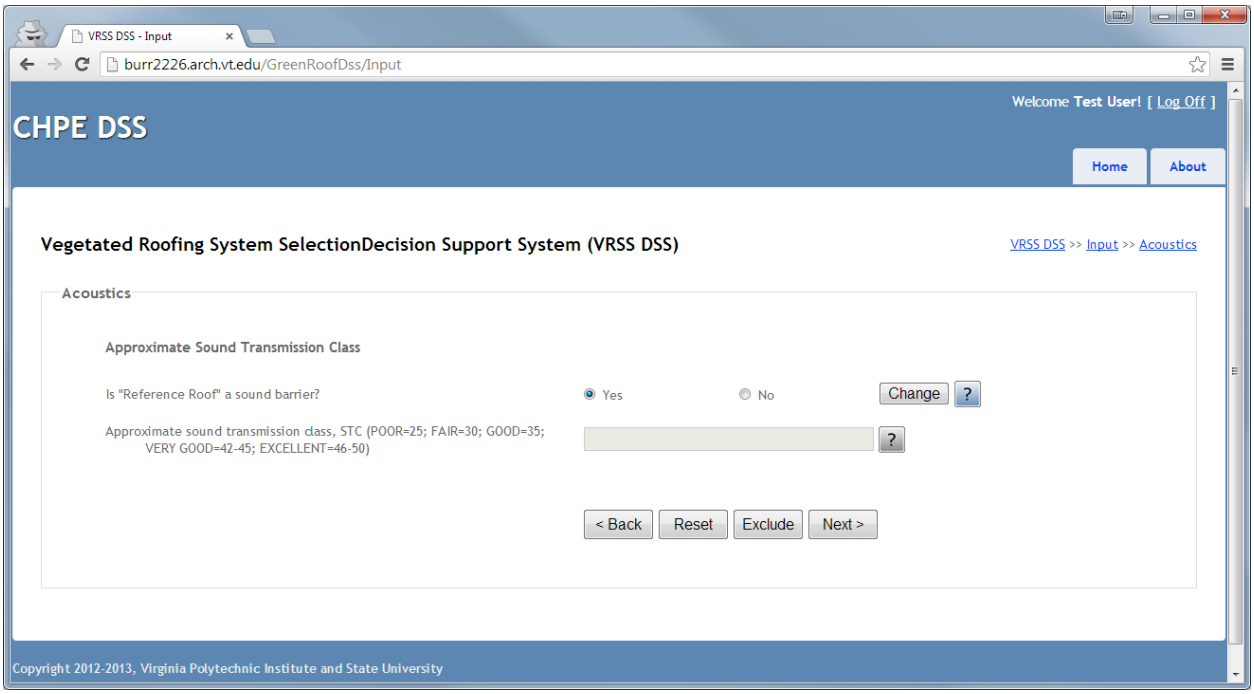


Figure D-12: Approximate Sound Transmission Class

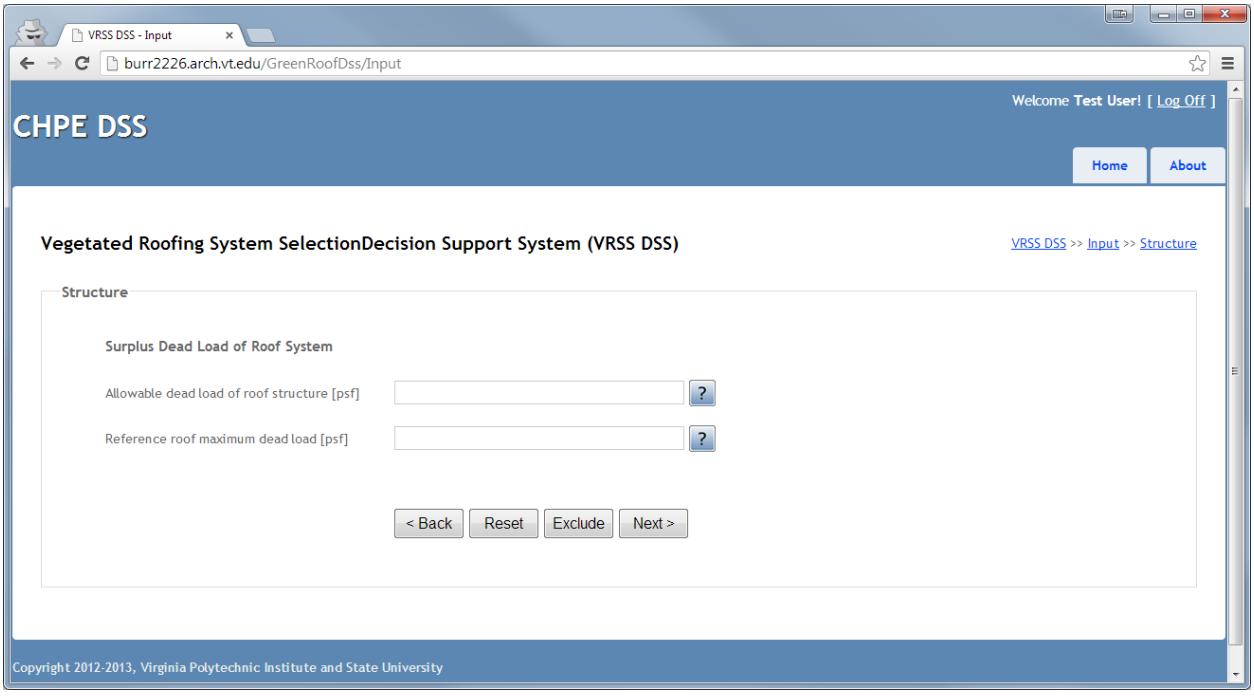


Figure D-13: Surplus Dead Load of Roof System

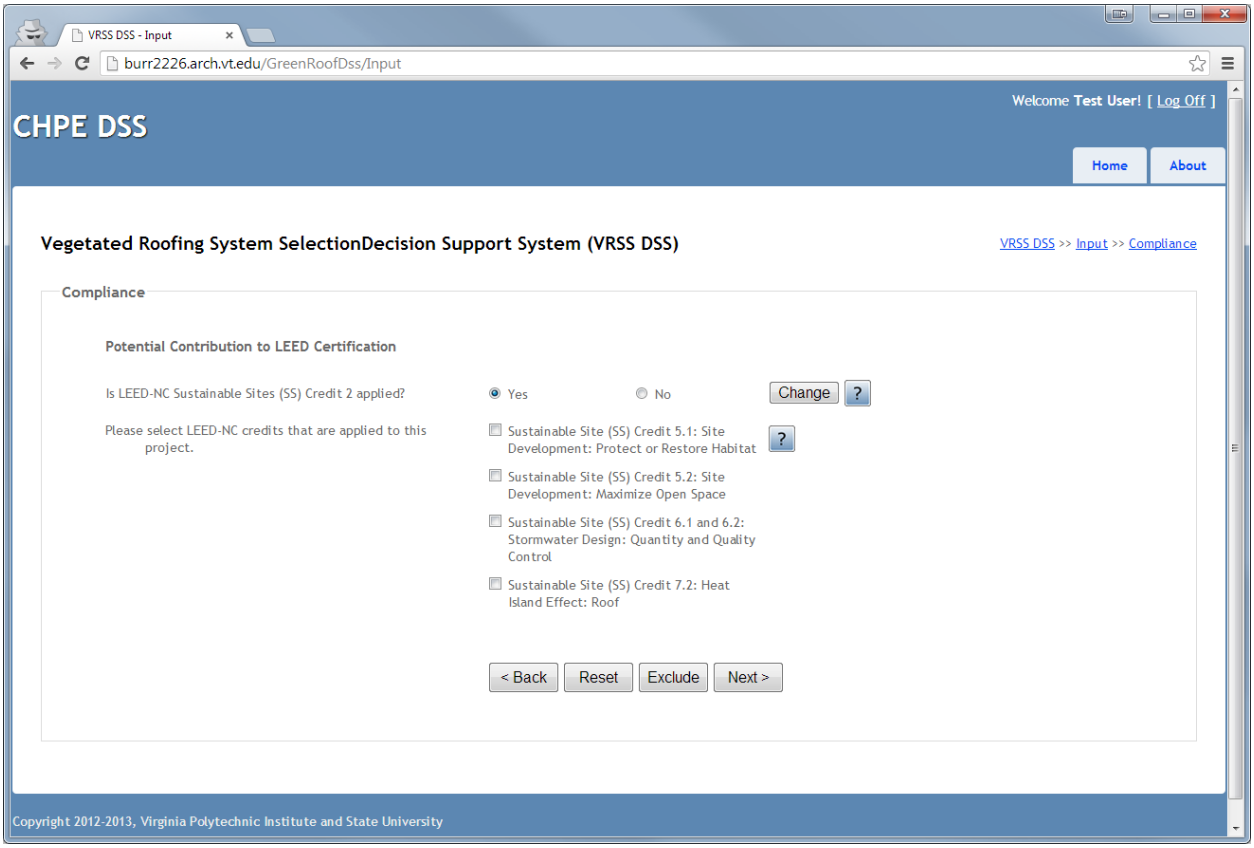


Figure D-14: Potential Contribution to LEED Certification

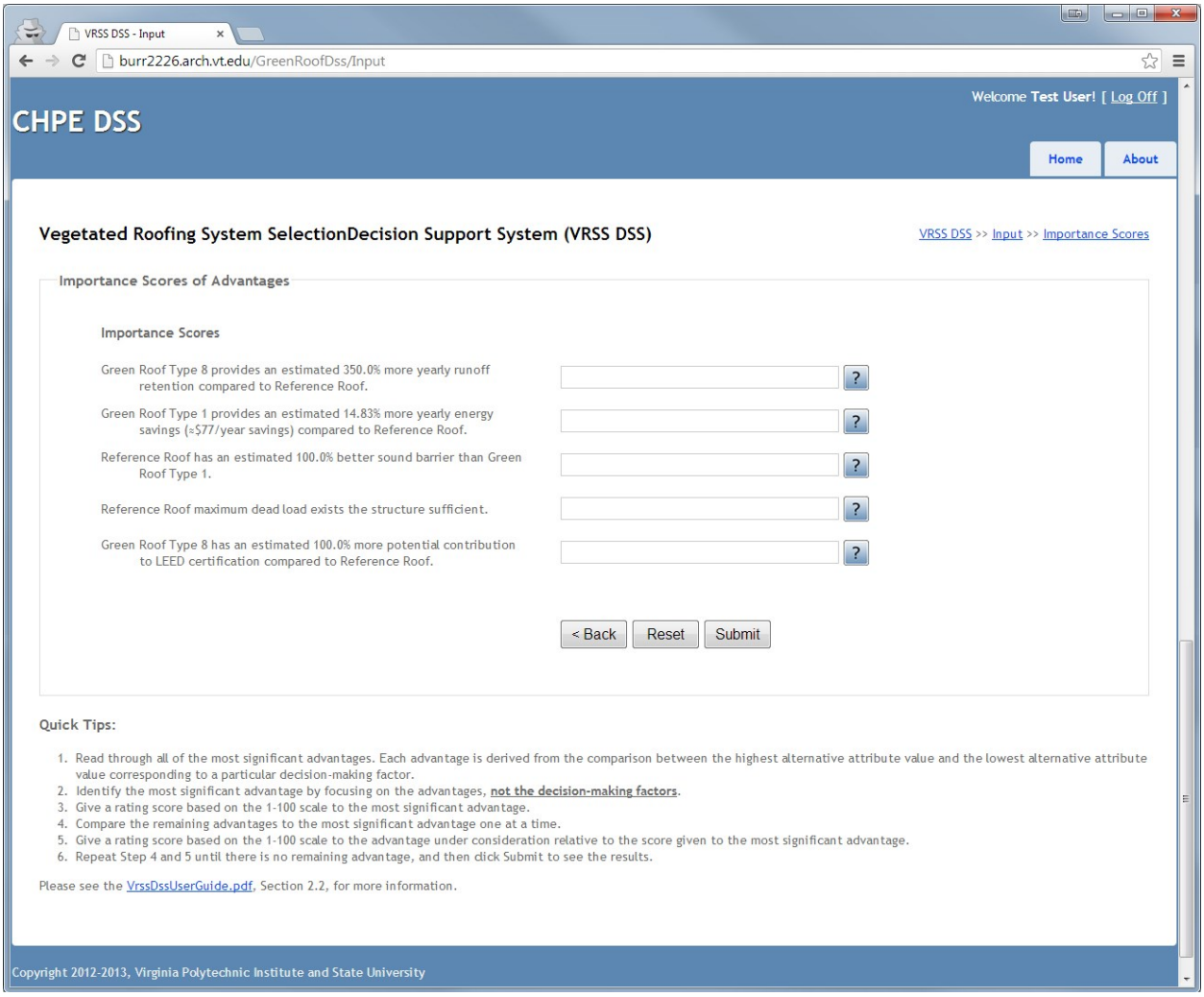


Figure D-15: Important Scores

Figure D-16: Report Page

Vegetated Roofing System Selection Decision Support System (VRSS DSS)

Project

Name: Demo Project

Reference Roof

Name: RT - 001

Description: White latex coating with highest solar reflectance on 5" foam insulation

Vegetated Roofing System

Selected system: Green Roof Type 8 with the total importance of advantages score of 240

Alternative system: Green Roof Type 3 with the total importance of advantages score of 223

Alternative system: Green Roof Type 6 with the total importance of advantages score of 198

Tabular Format Results

FACTORS	ALTERNATIVES															
	RR		G1		G2		G3		G4		G5		G6		G7		G8	
SW RETENTION																		
Attributes:	0.20		0.40		0.45		0.50		0.55		0.60		0.65		0.70		0.90	
Advantages:		0	100.0% more	26	125.0% more	32	150.0% more	39	175.0% more	45	200.0% more	51	225.0% more	58	250.0% more	0	350.0% more	**90**
POT. ENERGY SAVINGS																		
Attributes:	0.88		1.00		1.00		1.00		1.00		1.00		1.00		1.00		1.00	
Advantages:		**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**	14.0% more, $67/year	**0**
APPROXIMATE STC																		
Attributes:	1.00		0.00		0.00		0.25		0.50		0.75		1.00		1.00		1.00	
Advantages:	100.0% more	**70**		0		0	25.0% more	18	50.0% more	35	75.0% more	53	100.0% more	**70**	100.0% more	0	100.0% more	**70**
SURPLUS DL OF ROOF SYS.																		
Attributes:	1.00		1.00		1.00		1.00		0.00		0.00		0.00		0.00		0.00	
Advantages:	Exist structure sufficient	**100**	Exist structure sufficient	**100**	Exist structure sufficient	**100**	Exist structure sufficient	**100**		0		0		0		0		0
POT. CONTR. TO LEED CERT.																		
Attributes:	0.00		0.58		0.60		0.83		0.85		0.88		0.88		0.92		1.00	
Advantages:		0	58.3% more	47	60.4% more	48	83.3% more	67	85.4% more	68	87.5% more	70	87.5% more	70	91.7% more	0	100.0% more	**80**
TOTAL IMPORTANCE:	170		172		180		223		148		174		198		0		**240**	
TOTAL COST:	Not specified																	

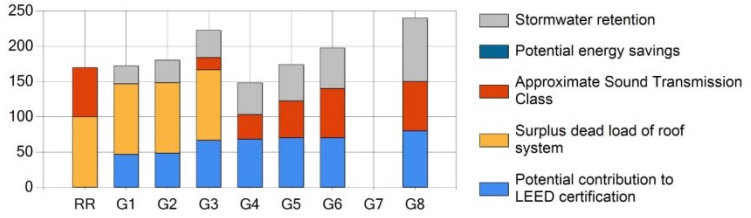
Key:

Value (least preferred is underlined)	
Advantage (greatest is bordered)	Importance of advantage (greatest is emphasized)

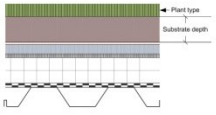
Gray rows - excluded fators

Gray rows - excluded alternatives

Total Importance of Advantages



Reference Table adapted from FFL[1] Standard 2002

Basic construction	Type of greening	System type	Substrate depth (Dcm)	Substrate depth (Din)	Plant type
	Extensive greening	G1	2.0≤Dcm≤4.0	0.8≤Din≤1.6	Moss-sedum
		G2	4.0<Dcm≤6.0	1.6<Din≤2.4	Sedum-moss
		G3	6.0<Dcm≤10.0	2.4<Din≤4.0	Sedum-moss-herb
		G4	10.0<Dcm≤15.0	4.0<Din≤6.0	Sedum-moss-herb
		G5	15.0<Dcm≤20.0	6.0<Din≤8.0	Grass-herb
	Intensive greening	G6	15.0<Dcm≤25.0	6.0<Din≤10.0	Lawn-perennial-shurb
		G7	25.0<Dcm≤50.0	10.0<Din≤20.0	Lawn-perennial-small shurb
		G8	50.0<Dcm	20.0<Din	Lawn-perennial-shurb tree

[1]Forschungsgesellschaft LandschaftsentwicklungLandschaftsbau e. V. (FLL). (2002). Guideline for the planning,execution and upkeep of green-roof sites. Bonn: FLL.

Figure D-17: Downloadable PDF Report

References

AIA. *Integrated Practice | Integrated Project Delivery*. The American Institute of Architects 2009. Available from http://www.aia.org/about/initiatives/AIAS078435.

———. 2011. 2011 Integrated Project Delivery Awareness Survey. The American Institute of Architects.

Allen, Scott K. 2011. TDD and Unit Testing with ASP.NET MVC 3. In Introduction to ASP.NET MVC 3: Pluralsight.

ASHRAE. 2006. *ASHRAE Greenguide : The Design, Construction, and Operation of Sustainable Buildings*. 2nd ed. Atlanta, GA: American Society of Heating, Refrigerating, and Air-conditioning Engineers.

Boddy, S., Y. Rezgui, M. Wetherill, and G. Cooper. 2006. "Knowledge Informed Decision Making in the Building Lifecycle: An application to the design of a Water Drainage System." *Automation in Construction* no. 16 (2007):596-606.

bSI. 2011. *Model - Industry Foundation Classes (IFC)*. The buildingSMART International 2011 [cited November 2011]. Available from http://buildingsmart.com/standards/ifc.

Cachero, Cristina, Nora Koch, Jaime Gomez, and Oscar Pastor. 2002. Conceptual Navigation Analysis: A Device and Platform Independent Navigation Specification. *Proc. 2nd Intl. Workshop on Web-Oriented Technology* (June 2002), http://users.dsic.upv.es/~west/iwwost02/papers/cachero.pdf.

Christensen, Clayton. 1997. "Patterns in the Evolution of Product Competition." *European Management Journal* no. Vol. 15 (No. 2):117-127.

D&R International, Ltd. 2012. 2011 Buildings Energy Data Book. Energy Efficiency & Renewable Energy Program, The U.S. Department of Energy.

Davis, Alan Mark. 1995. *201 principles of software development*. New York: McGraw-Hill.

Dee, Norbert, Janet Baker, Neil Drobny, Ken Duke, Ira Whitman, and Dave Fahringer. 1973. "An environmental evaluation system for water resource planning." *WATER RESOURCES RESEARCH* no. VOL. 9 (NO. 3):PP. 523-535.

Deitel, Paul J., and Harvey M. Deitel. 2008. *Internet & World Wide Web how to program*. 4th ed. Upper Saddle River, N.J.: Pearson Prentice Hall.

Eastman, Chuck, Paul Teicholz, Rafael Sacks, and Liston Kathleen. 2008. BIM Handbook: A Guilde to Building Information Modeling for Owners, Managers, Designers, Engineerings, and Contractors. Haboken, NJ: John Wiley & Sons.

Eastman, M., Chuck. 1975. "The Use of Computers Instead of Drawings in Building Design." *Journal of the American Institute of Architects* no. March:46-50.

Ekström, Martin A., and Hans C. Björnsson. 2003. "Evaluation IT Investments in Construction - Accounting for strategic Flexibility." *CIFE Technical Report* no. #136.

Flager, Forest, Grant Soremekun, Benjamin Welle, John Haymaker, and Prasum Bansal. 2008. "Multidisciplinary Process Integration & Design Optimization of a Classroom Building." *ITcon* no. 0 (October 2008):19.

Freeman, Adam, and Steven Sanderson. 2011. *Pro ASP.NET MVC 3 Framework*. 3rd ed, *The expert's voice in NET*. New York: Apress ;

Distributed to the book trade in the United States by Springer Science+Business Media.

Gnaho, C., and F. Larcher. 1999. "A User Centered Methodology for Complex and Customizable Web Applications Engineering." *Proc. 1st ICSE Workshop on Web Engineering, ACM, Los Angeles* no. May 1999.

Gorry, Anthony G., and Michel S. Scott-Morton. 1971. "A Framework for Management Information Systems." *Sloan Management Review* no. Vol. 13 (1):pp. 55-70.

Grant, Elizabeth Joyce. 2007. "A Decision-Making Framework for Vegetated Roofing System Selection." In *VPI & SU Architecture Ph D 2007*. Blacksburg, Va.: University Libraries, Virginia Polytechnic Institute and State University,. http://scholar.lib.vt.edu/theses/available/etd-11062007-232745.

GRAPHISOFT. 2012. "IFC 2x3 Reference Guide for ArchiCAD 16." In: GRAPHISOFT. http://www.archicadwiki.com/IFC.

Groat, Linda N., and David Wang. 2002. *Architectural Research Methods*. New York: J. Wiley.

Guba, Egon. 1981. "Criteria for Assessing the Truthworthiness of Naturalistic Inquiries." *Education Communication and Technology Journal* no. 29 (no. 2):76-91.

Haymaker, John, and Ben Suter. 2006. Communicating, Integrating and Improving Multidisciplinary Design and Analysis Narratives.

J. Fjermestad, and S. Hiltz. 2000. "Group Support Systems: A Descriptive Evaluation of Group Support Systems Case and Field Studies " *Journal of Management Information Systems* no. 17 (3):115-159.

Kaiser, Jennifer. *Elements of Effective Web Design*. About, Inc 2002. Available from http://webdesign.about.com/library/weekly/aa091998.htm.

Keen, Peter G W, and Michael S Scott-Morton. 1978. *Decision support systems : an organizational perspective*. Reading, MA: Addison-wesley.

Keen, Peter G. W. 1980. *Value Analysis : Justifying Decision Support Systems*, *Cisr 64*. Cambridge, Mass.: Center for Information Systems Research, Alfred P. Sloan School of Management.

Keysar, E., and A. R. Pearce. 2007. "Decision Support Tools for Green Building: Facilitating Selection among New Adopters on Public Sector Projects." *Journal of Green Building* no. 2 (3):153-171.

Krasner, Glenn E., and Stephen T. Pope. 1988. "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80." *Journal of Object Oriented Program* no. vol. 1 (no. 3):pp 26-49.

Krygiel, Eddy, and Bradley Nies. 2008. *Green BIM: Successful Sustainable Design with Building Information Modeling*. Indianapolis, IN: Wiley Publishing.

Kuhn, Thomas S. 1970. The Structure of Scientific Revolutions. 2nd ed, International encyclopedia of unified science Volumes I and II Foundations of the unity of science. Chicago: University of Chicago Press.

Laiserin, Jerry. *Graphisoft on BIM* 2002. Available from http://www.laiserin.com/features/issue19/feature01.php.

Liebich, Thomas. 2009. *IFC 2x Edition 3: Model Implementation Guide Version 2.0*: The buildingSMART International.

Love, Peter E.D., Zahir Irani, and David J. Edwards. 2004. "Industry-Centric Benchmarking of Information Technology Benefits, Costs, and Risks for Small-toMedium Sized Enterprises in Construction." *Automation in Construction* no. 13 (2004):507-524.

Lowagie, Bruno. 2011. *iText in action*. 2nd ed. Stamford, CT: Manning Publications.

Malkawi, Ali M. 2004a. "Developments in Environmental Performance Simulation." *Automation in Construction* no. 13:437-445.

———. 2004b. "Virtual Agents as a Decisiosn Support for Thermal Simulation." *Journal of Architectural and Planning Research* no. 21:4 (Winter 2004):363.

Malkawi, Ali M., Ravi S. Srinivasan, Yun K. Yi, and Ruchi Choudhary. 2005. "Decision Support and Design Evolution: Integrating Genetic Algorithms, CFD and Visualization." *Automation in Construction* no. 14:33-44.

Marsh, Laurence, and Roger Flanagan. 2000. "Measuring the Costs and Benefits of Information Technology in Construction." *Engineering, Construction, and Architectural Management* no. 7 (4):423-435.

McGraw-Hill Construction. 2012. The Business Value of BIM in North America: Multi-Year Trend Analysis and User Ratings (2007-2012). In *SmartMarket Report*: McGraw-Hill Financial, Inc.

Miles, Matthew B., and A. M. Huberman. 1994. *Qualitative data analysis : an expanded sourcebook*. 2nd ed. Thousand Oaks, Calif.: Sage.

Moore, Gordon E. 1965. "Cramming More Components onto Integrated Circuits." *Electronics* (April 19):114-117.

MSDN. *Platform Invocation Services*. Microsoft 2013. Available from http://msdn.microsoft.com/en-us/library/aa712982%28v=vs.71%29.aspx.

Myers, Glenford J. 1979. The art of software testing, Business data processing, a Wiley series. New York: Wiley.

Nguyen, Hung Q. 2000. Testing Web-based Applications. *Software Testing & Quality Engineering* (May/June 2000), http://www.logigear.com/logi_media_dir/Documents/whitepapers/testing_web_applications.pdf.

Nguyen, Hung Quoc. 2001. Testing applications on the Web : test planning for Internet-based systems. New York: Wiley.

NIBS. 2007. "Section 2 – Prologue to the National BIM Standard." In *United States National Building Information Modeling Standard*, 20. Washington, DC: National Institute of Building Sciences.

Offutt, Jeff. 2002. "Web Software Applications Quality Attributes." *IEEE Software* no. March-April 2002:pp. 25-32.

Olsina, L., D. Godoy, G.J. Lafuente, and G. Rossi. 1999. "Specifying Quality Characteristics and Attributes for Web Sites." *Proc. 1st ICSE Workshop on Web Engineering, ACM, Los Angeles* no. May 1999.

Onuma. 2010. *Open Architecture*. Onuma Inc. 2010 [cited 07/05/ 2010]. Available from http://onuma.com/Contact/.

Onuma, Kimon. 2009. From Abundance to Scaricity: A Strateggy for the 21st Century Building Industry. (October 5), http://wbdg.org/pdfs/abundance_to_scarcity.pdf.

Ott, Lyman, and Michael Longnecker. 2001. *An introduction to statistical methods and data analysis*. 5th ed. Australia ; Pacific Grove, CA: Duxbury/Thomson Learning.

Patton, Michael Quinn. 1989. *Qualitative evaluation methods*. 10th ed. Beverly Hills: Sage Publications.

Pollalis, Spiro N., and J. Yannis Bakos. 1996. "Technology in the Design Process " *Journal of Architectural and Planning Research* no. v.13:no.2 (Summer 1996).

Power, Daniel J. 2000. "Decision Support Systems Hyperbook." In. Cedar Falls, IA: DSSResources.COM. http://dssresources.com/subscriber/password/dssbookhypertext (accessed October 2010).

Pressman, Andrew. *Integrated practice in perspective: A new model for the architectural profession*. Architecture Record 2007. Available from http://archrecord.construction.com/practice/projDelivery/0705proj-1.asp.

Pressman, Roger S. 2010. *Software Engineering : A Practitioner's Approach*. 7th ed. New York: McGraw-Hill Higher Education.

Putnam, Lawrence H., and Ware Myers. 1997. "How Solved Is the Cost Estimation Problem?" *IEEE Software* (November 1997):pp. 105-107.

Randolph, John, and Gilbert M. Masters. 2008. *Energy for Sustainability : Technology, Planning, Policy*. Washington, D.C.: Island Press.

Rayport, Jeffrey F., and Bernard J. Jaworski. 2004. *Introduction to e-commerce*. 2nd ed. Boston: McGraw-Hill Irwin MarketspaceU.

Rob, Peter, and Carlos Coronel. 2009. *Database systems : design, implementation, and management*. 8th ed. Boston, Mass.: Thomson/Course Technology.

Rogers, Everett M. 2003. *Diffusion of Innovations*. 5th ed. New York: Free Press.

Rossman, Gretchen B., and Sharon F. Rallis. 2003. *Learning in the field : an introduction to qualitative research*. 2nd ed. Thousand Oaks, Calif.: Sage Publications.

Schilling, Melissa A. 2008. *Strategic Management of Technological Innovation*. 2nd ed. Boston: McGraw-Hill/Irwin.

Seidman, Irving. 2006. Interviewing as qualitative research : a guide for researchers in education and the social sciences. 3rd ed. New York: Teachers College Press.

Shaw, Mary, and David Garlan. 1996. *Software Architecture : Pespectives on an Emerging Discipline*. Upper Saddle River, N.J.: Prentice Hall.

Simon, Herbert A. 1977. *The New Science of Management Decision*. Rev. ed. Englewood Cliffs, N.J.: Prentice-Hall.

———. 1980. "Cognitive Science: The Newest Science of the Artificial." *Cognitive Science* no. Volume 4 (Issue 1):Pages 33-46.

Smith, Dana K., and Michael Tardif. 2009. Building Information Modeling : A Strategic Implementation Guide for Architects, Engineers, Constructors, and Real Estate Asset Managers. Hoboken, N.J.: Wiley.

Sprague, Ralph H., and Eric D. Carlson. 1982. *Building effective decision support systems*. Englewood Cliffs, N.J.: Prentice-Hall.

Stake, Robert E. 2000. "Case Studies. In N.K. Denzin & Y.S. Lincoln (Eds.)." In *Handbook of Qualitative Research* 435-454. Thousand Oaks, CA: Sage.

Stein, Benjamin, John S. Reynolds, Walter T. Grondzik, and Alison G. Kwok. 2006. *Mechanical and Electrical Equipment for Buildings*. 10th ed. Hoboken, N.J.: Wiley.

Suehiro, James M. *Integrated Project Delivery – Expanded Sustainability*. The American Institute of Architects 2008. Available from http://www.aia.org/akr/Resources/Documents/AIAP037899.

Suhr, Jim. 1999. The choosing by advantages decisionmaking system. Westport, Conn.: Quorum.

Turban, Efraim, Jay E. Aronson, Ting-Peng Liang, and Ramesh Sharda. 2007. *Decision Support and Business Intelligence Systems*. 8 ed. Upper Saddle River, NJ: Pearson Prentice Hall.

UN. 2010. *42/187. Report of the World Commission on Environment and Development* United Nations 1987 [cited September 7th 2010]. Available from http://www.un-documents.net/a42r187.htm.

Von Wodtke, Mark. 2000. Design with Digital Tools : Using New Media Creatively, McGraw-Hill professional architecture. New York: McGraw-Hill.

Vries, Mark de, and Harry Wagter. 1990. "A CAAD Model for Use in Early Design Phases." In *The Electronic design studio : architectural knowledge and media in the computer era*, edited by Malcolm McCullough, William J. Mitchell and Patrick A. Purcell, viii, 505 p. Cambridge, Mass.: MIT Press.

W3Schools. 2010. *Introduction to XHTML*. W3Schools 2010 [cited November 2010]. Available from http://www.w3schools.com/xhtml/xhtml_intro.asp.

Zorgdrager, Kelby. *Choosing the Right Java Web Development Framework* 2010. Available from http://www.openlogic.com/wazi/bid/188143/.