A Conceptual Framework for Specification of Network-Centric System Architectures

DZMITRY CHURBANAU

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

May 3, 2010

Blacksburg, Virginia

Keywords and phrases: conceptual framework, network-centric system of systems, system architecting, system architecture, system architecture specification

A Conceptual Framework for Specification of Network-Centric System Architectures

DZMITRY CHURBANAU

ABSTRACT

Software-based system architecture has been recognized as a foundation laying out the underpinnings that are critically important for successful engineering of large-scale complex systems. In recent years, architecting has played a more crucial role in engineering network-centric system of systems. The software paradigm has been shifting from treating software as a product (SaaP) to treating software as a service (SaaS). SaaS is also referred to as the Cloud Computing, where the term "cloud" is used as a metaphor for "network".

As the complexity of the architecture of network-centric software-based system of systems has increased, the description of such architecture has posed significant technical challenges. The U.S. Department of Defense (DoD) has developed the DoD Architecture Framework [DoDAF 2009a, DoDAF 2009b] for describing system architectures. IEEE proposes a Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE 2000]. SEI provides high-level guidelines for Documenting Software Architectures [Clements *et al* 2003]. However, all of the diagrams proposed by DoD, IEEE, and SEI are two-dimensional static graphical and textual representations that do not reveal the dynamic characteristics of a system architecture.

This thesis presents a conceptual framework (CF) for specifying the architecture of a network-centric software-based system of systems. The developed CF provides the beginning part of a larger research effort. The main goal of the overall research is to employ the automation-based software paradigm and to automatically generate a visual simulation model of a system architecture, with which experiments can be conducted to assess the dynamic characteristics of that architecture. The CF, developed in the research described herein, enables the automatic generation of a visual simulation model representing a system architecture. The proposed CF is evaluated in half a dozen case studies to demonstrate that it provides the necessary elements for automatic generation of a simulation model as the description of a complex system of systems architecture.

DEDICATION

This thesis is dedicated to my beautiful wife Marina and my daughter Milla. My wife has always been the closest person during my studies at Virginia Tech. Her constant support in every possible way has always helped me to move forward to reach my goals. I would not be able to achieve my success without her support.

I also want to dedicate this work to my parents and parents of my wife. Thank you all for all your advice and support.

ACKNOWLEDGMENTS

I am very grateful to my advisor Dr. Osman Balci for his academic, professional and personal support throughout my education at Virginia Tech. He helped me to overcome the research obstacles I had and always was my source of inspiration. I am inspired of his diligence and professionalism.

I have been lucky to work with Dr. James D. Arthur. His teaching has helped me a lot and I am very happy that I was able to learn from him. I also want to thank Dr. Eli Tilevich for his valuable advice and for serving on my committee.

I also want to thank everyone in the Software Engineering Research Group, especially Amine Chigani and Shvetha Soundararajan. I want to express my appreciation to Dr. Pardha Pyla who guided me a lot during my first year at Virginia Tech.

And finally, I want to thank the graduate program committee of the Computer Science department for giving me the opportunity to pursue my graduate study and everyone in the College of Engineering Dean's Office for supporting me in the hard moments I had.

TABLE OF CONTENTS

ABSTRACT .... **ii**

ACKNOWLEDGMENTS .... **iv**

LIST OF FIGURES .... **vii**

LIST OF TABLES .... **ix**

LIST OF ACRONYMS .... **x**

CHAPTER 1: INTRODUCTION **1**

1.1 NETWORK-CENTRIC ARCHITECTURE 1
1.1.1 Client-Server Architecture 1
1.1.1.1 Two-tier client server architecture 2
1.1.1.2 Three-tier and n-tier architectures 2
1.1.1.3 Thin-client model 3
1.1.1.4 Thick-client model 3
1.1.2 Distributed Objects Architecture 3
1.1.2.1 Common Object Request Broker Architecture (CORBA) 4
1.1.3 Peer-to-Peer Architecture 5
1.1.4 Service-Oriented Architecture 7
1.2 STATEMENT OF THE PROBLEM 8
1.3 STATEMENT OF OBJECTIVES 8
1.4 OVERVIEW OF THESIS 9

CHAPTER 2: RELATED WORK **10**

2.1 FRAMEWORKS 10
2.1.1 Zachman Framework 10
2.1.1.1 Architecture Framework 10
2.1.2 Department of Defense Architecture Framework (DoDAF) 12
2.1.2.1 Architecture Framework 12
2.1.3 The Open Group Architecture Framework (TOGAF) 16
2.1.3.1 Architecture Framework 16
2.1.3.1.1 Architecture Development Model 16
2.1.3.1.2 Architecture Content Framework 17
2.1.3.1.3 Enterprise Continuum 18
2.1.3.1.4 Architecture Capability Framework 18
2.1.4 Institute of Electrical and Electronics Engineers (IEEE) 1471 Standard 19
2.1.4.1 Conceptual Framework 19
2.1.4.1.1 Architectural Description 19
2.1.4.1.2 Stakeholders 21
2.1.4.2 IEEE 1471 Requirements and Unresponsiveness 21
2.1.5 Software Engineering Institute (SEI) Views and Beyond Approach 21
2.1.5.1 Core Concepts 21
2.1.5.2 Documenting Software Architecture 22
2.2 TOOLS 22
2.2.1 Telelogic System Architect 23
2.2.2 Metastorm ProVision 23

CHAPTER 3: THE PROPOSED CONCEPTUAL FRAMEWORK **25**

3.1 STATIC STRUCTURE 25
3.1.1 Component 25
3.1.1.1 Overview 25
3.1.1.2 Defining a component 25
3.1.1.3 Component properties 27
3.1.2 Connection 28
3.1.2.1 Defining a connection 28

3.1.2.2 Connection properties 29
3.1.3 Service *31*
3.1.3.1 Defining a service 31
3.1.3.2 Open system paradigm 31
3.2 DYNAMIC STRUCTURE 32
3.2.1 Message *32*
3.2.1.1 Defining a message 32
3.2.1.2 Message properties 32

CHAPTER 4: EVALUATION OF THE CONCEPTUAL FRAMEWORK 34

4.1 THE PROCESS OF CREATING SYSTEM ARCHITECTURE SPECIFICATION AND SIMULATION MODEL 34
4.2 EXAMPLES 35
4.2.1 Online Banking System *35*
4.2.1.1 Overview 35
4.2.1.2 System Architecture Specification 35
4.2.1.3 Simulation Model 42
4.2.1.4 Simulation Model in Action 46
4.2.2 Online Learning Environment *48*
4.2.2.1 Overview 48
4.2.2.2 System Architecture Specification 48
4.2.2.3 Simulation Model 53
4.2.3 E-Commerce System *56*
4.2.3.1 Overview 56
4.2.3.2 System Architecture Specification 56
4.2.3.3 Simulation Model 60
4.2.4 Research station *63*
4.2.4.1 Overview 63
4.2.4.2 System Architecture Specification 63
4.2.4.3 Simulation Model 66
4.2.5 Travel Reservation System *69*
4.2.5.1 Overview 69
4.2.5.2 System Architecture Specification 69
4.2.5.3 Simulation Model 73
4.2.6 Logistic Information Management System *76*
4.2.6.1 Overview 76
4.2.6.2 System Architecture Specification 76
4.2.6.3 Simulation Model 79

CHAPTER 5: CONCLUSIONS AND FUTURE RESEARCH 81

5.1 CONCLUSIONS 81
5.2 CONTRIBUTIONS 82
5.3 FUTURE RESEARCH 82

BIBLIOGRAPHY 83

LIST OF FIGURES

FIGURE 1. CLIENT-SERVER ARCHITECTURE 1
FIGURE 2. TWO-TIER CLIENT-SERVER ARCHITECTURE 2
FIGURE 3. THREE-TIER CLIENT-SERVER ARCHITECTURE 3
FIGURE 4. EXAMPLE OF DISTRIBUTED OBJECTS ARCHITECTURE. 4
FIGURE 5. EXAMPLE OF INTER-ORB COMMUNICATION 5
FIGURE 6. THE EXAMPLE OF PEER-TO-PEER ARCHITECTURE 6
FIGURE 7. THE EXAMPLE OF DECENTRALIZED PEER-TO-PEER ARCHITECTURE 6
FIGURE 8. THE EXAMPLE OF SEMI-CENTRALIZED PEER-TO-PEER ARCHITECTURE 6
FIGURE 9. SOA WORKING MODEL 7
FIGURE 10. THE ZACHMAN FRAMEWORK [ZACHMAN 2008] [FAIR USE] 11
FIGURE 11. CONCEPTUAL MODEL OF IEEE 1471 ARCHITECTURAL DESCRIPTION [IEEE 2000] [FAIR USE] 20
FIGURE 12. COMPONENTS DECOMPOSITION 26
FIGURE 13. EXAMPLE OF COMPONENTS PROPERTIES 28
FIGURE 14. EXAMPLE OF CONNECTED COMPONENTS 28
FIGURE 15. EXAMPLE OF COMPONENTS CONNECTION PROPERTIES 29
FIGURE 16. MODIFIED EXAMPLE OF COMPONENTS CONNECTION PROPERTIES 30
FIGURE 17. COMPONENTS AND SERVICES ARE DEFINED USING OPEN SYSTEM PARADIGM 31
FIGURE 18. THE PROCESS OF CREATING SYSTEM ARCHITECTURE SPECIFICATION AND SIMULATION MODEL 34
FIGURE 19. TOP VIEW OF THE ONLINE BANKING SYSTEM ARCHITECTURE SPECIFICATION 36
FIGURE 20. SPECIFICATION OF THE ATM COMPONENT 37
FIGURE 21. SPECIFICATION OF THE MAIN BANK COMPONENT 37
FIGURE 22. SPECIFICATION OF THE ACCOUNT SERVICES COMPONENT 38
FIGURE 23. SPECIFICATION OF THE USER MANAGEMENT SERVICES COMPONENT 38
FIGURE 24. SPECIFICATION OF THE ACCOUNT SERVICES COMPONENT 39
FIGURE 25. SPECIFICATION OF THE TRANSACTION SERVICES COMPONENT 39
FIGURE 26. SPECIFICATION OF THE NOTIFICATION SERVICES COMPONENT 40
FIGURE 27. SPECIFICATION OF THE ACCESS SERVICES COMPONENT 40
FIGURE 28. SPECIFICATION OF THE DATA SERVICES COMPONENT 41
FIGURE 29. SPECIFICATION OF THE DATA SERVICES COMPONENT OF DATA SERVICES COMPONENT GROUP 41
FIGURE 30. TOP VIEW OF THE SIMULATION MODEL OF ONLINE BANKING SYSTEM IN VSE 42
FIGURE 31. ATM COMPONENT VIEW IN VSE 42
FIGURE 32. MAIN BANK VIEW IN VSE 43
FIGURE 33. USER APPLICATION INTERFACE COMPONENT VIEW IN VSE 43
FIGURE 34. CLERK APPLICATION INTERFACE COMPONENT VIEW IN VSE 43
FIGURE 35. ACCOUNT SERVICES COMPONENT VIEW IN VSE 44
FIGURE 36. TRANSACTION SERVICES COMPONENT VIEW IN VSE 44
FIGURE 37. DATA SERVICES COMPONENT VIEW IN VSE 45
FIGURE 38. DATA SERVICES SUB COMPONENT VIEW IN VSE 45
FIGURE 39. LOGGING SERVICES COMPONENT VIEW IN VSE 45
FIGURE 40. NOTIFICATION SERVICES COMPONENT VIEW IN VSE 46
FIGURE 41. USER MANAGEMENT SERVICES COMPONENT VIEW IN VSE 46
FIGURE 42. TOP VIEW OF ONLINE BANKING SYSTEM DURING RUNNING SIMULATION MODEL IN VSE 47
FIGURE 43. MAIN BANK VIEW OF ONLINE BANKING SYSTEM DURING RUNNING SIMULATION MODEL IN VSE 47
FIGURE 44. USER APPLICATION INTERFACE VIEW OF ONLINE BANKING SYSTEM DURING RUNNING SIMULATION MODEL IN VSE 47
FIGURE 45. TOP VIEW OF THE OLE ARCHITECTURE SPECIFICATION 48
FIGURE 46. SPECIFICATION OF THE OLE COMPONENT 49
FIGURE 47. SPECIFICATION OF THE APPLICATION SERVICES COMPONENT 49
FIGURE 48. SPECIFICATION OF THE HOME SERVICES COMPONENT 50
FIGURE 49. SPECIFICATION OF THE COLLABORATION SERVICES COMPONENT 50
FIGURE 50. SPECIFICATION OF THE CLASS SERVICES COMPONENT 51
FIGURE 51. SPECIFICATION OF THE LEARNING TOOLS SERVICES COMPONENT 51
FIGURE 52. SPECIFICATION OF THE MOBILE SERVICES COMPONENT 52

FIGURE 53. SPECIFICATION OF THE PURE WEB SERVICES COMPONENT 52
FIGURE 54. TOP VIEW OF THE SIMULATION MODEL OF OLE IN VSE 53
FIGURE 55. ONLINE LEARNING ENVIRONMENT VIEW IN VSE 53
FIGURE 56. APPLICATION SERVICES COMPONENT VIEW IN VSE 54
FIGURE 57. COLLABORATION SERVICES COMPONENT VIEW IN VSE 54
FIGURE 58. CLASS SERVICES COMPONENT VIEW IN VSE 54
FIGURE 59. LEARNING TOOLS SERVICES COMPONENT VIEW IN VSE 55
FIGURE 60. MOBILE SERVICES COMPONENT VIEW IN VSE 55
FIGURE 61. PURE WEB SERVICES COMPONENT VIEW IN VSE 55
FIGURE 62. TOP VIEW OF THE E-COMMERCE SYSTEM ARCHITECTURE SPECIFICATION 56
FIGURE 63. SPECIFICATION OF THE E-COMMERCE SYSTEM COMPONENT 57
FIGURE 64. SPECIFICATION OF THE ACCOUNT SERVICES COMPONENT 57
FIGURE 65. SPECIFICATION OF THE PRODUCT SERVICES COMPONENT 58
FIGURE 66. SPECIFICATION OF THE PAYMENT COMPONENT 58
FIGURE 67. SPECIFICATION OF THE SHOPPING CART COMPONENT 59
FIGURE 68. SPECIFICATION OF THE DELIVERY COMPONENT 59
FIGURE 69. TOP VIEW OF THE SIMULATION MODEL OF E-COMMERCE SYSTEM IN VSE 60
FIGURE 70. E-COMMERCE COMPONENT VIEW IN VCE 60
FIGURE 71. APPLICATION INTERFACE COMPONENT VIEW IN VCE 61
FIGURE 72. ACCOUNT SERVICES COMPONENT VIEW IN VCE 61
FIGURE 73. PRODUCT SERVICES COMPONENT VIEW IN VCE 61
FIGURE 74. PAYMENT COMPONENT VIEW IN VCE 62
FIGURE 75. SHOPPING CART COMPONENT VIEW IN VCE 62
FIGURE 76. DELIVERY COMPONENT VIEW IN VCE 62
FIGURE 77. TOP VIEW OF THE RESEARCH STATION ARCHITECTURE SPECIFICATION 63
FIGURE 78. SPECIFICATION OF THE RESEARCH STATION COMPONENT 64
FIGURE 79. SPECIFICATION OF THE ANTENNA COMPONENT 64
FIGURE 80. SPECIFICATION OF THE SATELLITE COMPONENT 65
FIGURE 81. SPECIFICATION OF THE AIRPLANE COMPONENT 65
FIGURE 82. SPECIFICATION OF THE SHIP COMPONENT 66
FIGURE 83. TOP VIEW OF THE SIMULATION MODEL OF RESEARCH STATION IN VSE 66
FIGURE 84. RESEARCH STATION COMPONENT VIEW IN VSE 67
FIGURE 85. ANTENNA COMPONENT VIEW IN VSE 67
FIGURE 86. SATELLITE COMPONENT VIEW IN VSE 67
FIGURE 87. SHIP COMPONENT VIEW IN VSE 68
FIGURE 88. AIRPLANE COMPONENT VIEW IN VSE 68
FIGURE 89. TOP VIEW OF THE TRAVEL RESERVATION SYSTEM ARCHITECTURE SPECIFICATION 69
FIGURE 90. SPECIFICATION OF THE TRAVEL RESERVATION SYSTEM COMPONENT 70
FIGURE 91. SPECIFICATION OF THE ACCOUNT SERVICES COMPONENT 70
FIGURE 92. SPECIFICATION OF THE RESERVATION SERVICES COMPONENT 71
FIGURE 93. SPECIFICATION OF THE AIRLINE COMPANIES COMPONENT 71
FIGURE 94. SPECIFICATION OF THE CAR RENTAL COMPANIES COMPONENT 72
FIGURE 95. SPECIFICATION OF THE HOTELS COMPONENT 72
FIGURE 96. TOP VIEW OF THE SIMULATION MODEL OF TRAVEL RESERVATION SYSTEM IN VSE 73
FIGURE 97. TRAVEL RESERVATION SYSTEM VIEW IN VSE 73
FIGURE 98. APPLICATION INTERFACE COMPONENT VIEW IN VSE 74
FIGURE 99. ACCOUNT SERVICES COMPONENT VIEW IN VSE 74
FIGURE 100. RESERVATION SERVICES COMPONENT VIEW IN VSE 74
FIGURE 101. AIRLINE COMPANIES COMPONENT VIEW IN VSE 74
FIGURE 102. HOTELS COMPONENT VIEW IN VSE 75
FIGURE 103. CAR RENTAL COMPANIES COMPONENT VIEW IN VSE 75
FIGURE 104. TOP VIEW OF THE LOGISTIC INFORMATION MANAGEMENT SYSTEM ARCHITECTURE SPECIFICATION 76
FIGURE 105. SPECIFICATION OF THE LOGISTIC OFFICES COMPONENT 77
FIGURE 106. SPECIFICATION OF THE SHIPPERS COMPONENT 78
FIGURE 107. SPECIFICATION OF THE STORAGES COMPONENT 78
FIGURE 108. TOP VIEW OF THE SIMULATION MODEL OF LOGISTIC INFORMATION MANAGEMENT SYSTEM IN VSE 79

Figure 109. Logistic Offices component view in VSE 79
Figure 110. Shippers component view in VSE..80
Figure 111. Storages component view in VSE..80

LIST OF TABLES

Table 1. Architecture Viewpoints in DoDAF v2.0 13
Table 2. DoDAF v2.0 Models 13
Table 3. Summary of component properties 27
Table 4. Summary of connection properties 29

LIST OF ACRONYMS

AD	Architectural Description
ADM	Architecture Development Model
ANSI	American National Standard Institute
ATM	Automated Teller Machine
AV	All Viewpoint
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
CF	Conceptual Framework
CIO	Chief Information Officer
CORBA	Common Object Request Broker Architecture
CV	Capability Viewpoint
C&C	Component-and-Connector
DCOM	Distributed Component Object Model
DIV	Data and Information Viewpoint
DOA	Distributed Objects Architecture
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
ESB	Enterprise Service Bus
HTML	HyperText Markup Language
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
IDL	Interface Definition Language
ISO	International Organization for Standardization
JCA	Joint Capability Area
OLE	Online Learning Environment
OMG	Object Management Group
ORB	Object Request Broker
OV	Operational Viewpoint
PC	Personal Computer
PDA	Personal Digital Assistant
PV	Project Viewpoint
SEI	Software Engineering Institute
SOA	Service-Oriented Architectures
StdV	Standard Viewpoint
SV	Systems Viewpoint
SvcV	Services Viewpoint
TAFIM	Technical Architecture Framework for Information Management
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language
VSE	Visual Simulation Environment
V&B	Views and Beyond
W3C	World Wide Web Consortium

CHAPTER 1: INTRODUCTION

With the expansion of network technology, software industry has moved from PC-centric concept to the network-centric concept [Garlan 2000]. Along with this, the way of architecting and designing software systems has changed. Each component of such systems is an independent subsystem which can be architected, designed, developed and managed separately.

"This trend had a number of consequences for software engineering, in general, and software architecture, in particular" [Garlan 2000]. Architecting process for such systems became an essential part in software life cycle model [Balci 2009b]. Architectural descriptions have long been recognized as an essential ingredient of a well-designed system [Monroe 1997]. These descriptions mostly consist of a set of elements and interactions between them. However they are all static and do not show dynamic perspective of the architected system.

1.1 Network-Centric Architecture

Network-centric architectures describe software systems which are loosely-coupled, consist of independent components which more often is an integral of a larger system that involves the collaboration of software, hardware, and people to solve complex computing problems [Chigani 2007].

The following sections describe major network-centric software-based system architectures.

1.1.1 Client-Server Architecture

Client-server architecture is a model of computing which splits computing tasks between client and server. Most often clients and servers are located on different hardware and are communicating over the network (see Figure 1). In such type of architecture, clients request services and servers provide them to the clients.

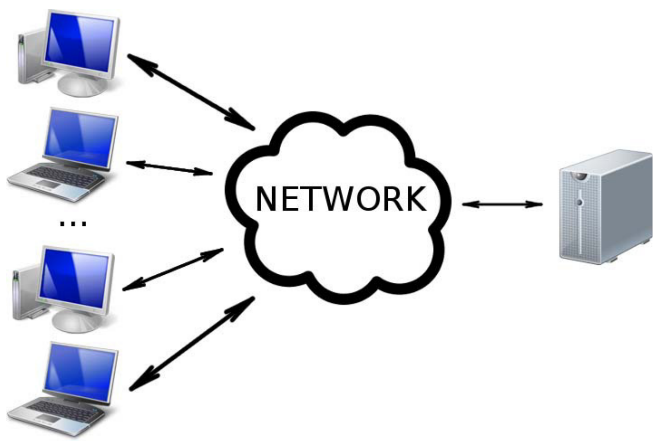


Figure 1. Client-server architecture

1.1.1.1 Two-tier client server architecture

Two-tier is the simplest type of client-server architecture. It basically defines how application processing is divided in the application. In two-tier application presentation layer is provided to multiple clients which communicate with the sever computer over the network. The server computer plays the role of centralized data storage (see Figure 2).

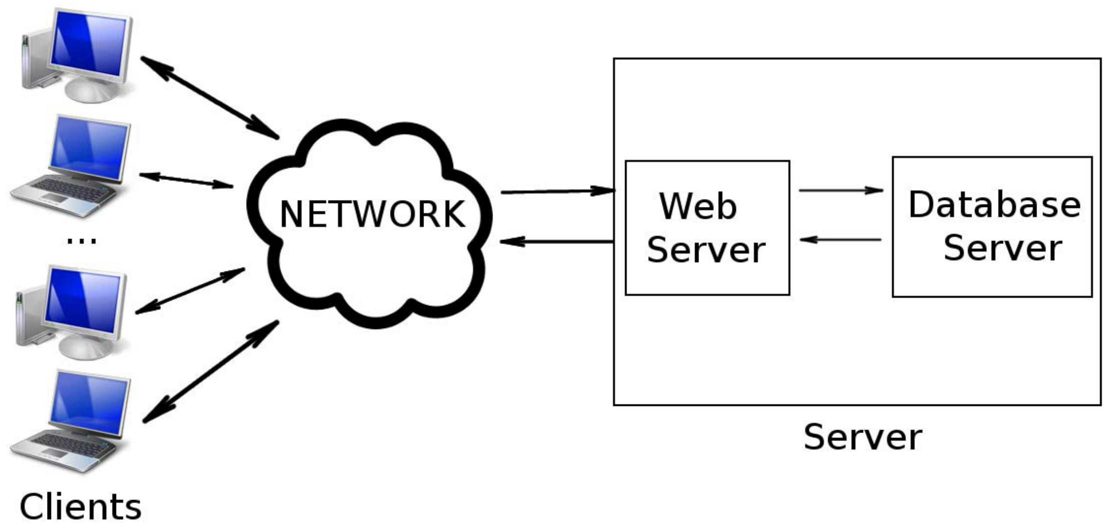


Figure 2. Two-tier client-server architecture

Most internet applications, such as email, telnet, ftp and a lot of web-sites are simple two-tier applications. Without providing a lot of data processing these applications provide a simple interface to access data over the network [Reese 2000].

A simple two-tier application can be a kind of application where clients are responsible for all application logic and servers are only responsible for data management. Therefore client-server applications are categorized as [Balci 2009c]:

- Thin-client model
- Thick-client model

1.1.1.2 Three-tier and n-tier architectures

Three-tier, also commonly called n-tier or multi-tier, architecture is the extension of two-tier client-server architecture. Three-tier architecture adds one more processing layer to the two-tier architecture, which separates application management and data management processes. Such decomposition helps to isolate data processing in a central and maximize object reuse [Reese 2000].

In Figure 3 you can see the example of the application which uses three-tier architecture.

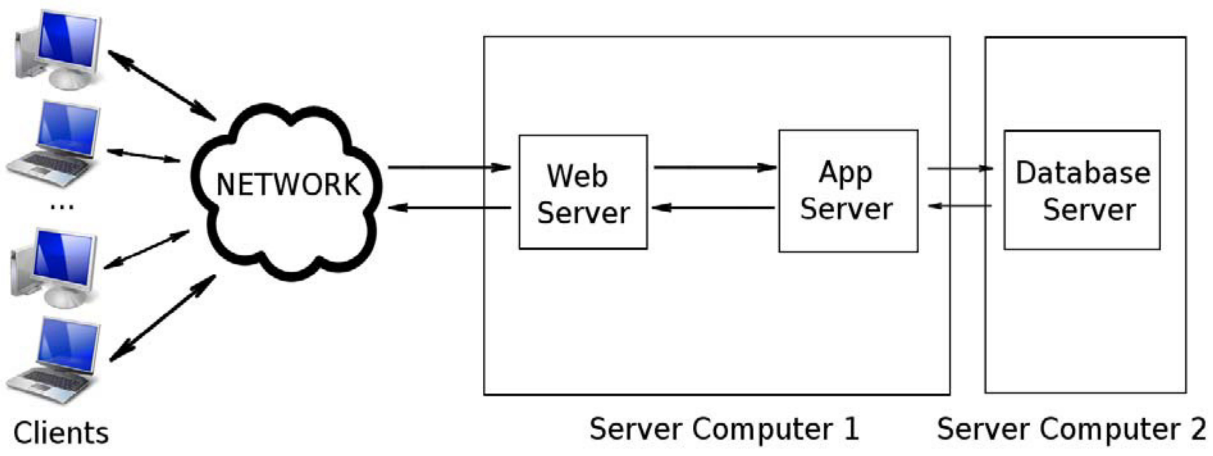


Figure 3. Three-tier client-server architecture

In Figure 3 tier 1 contains presentation layer which is provided to the user. Tier 2 contains business process logic and the data access and is provided by the application server. Tier 3 is a database server which contains and provides business data.

1.1.1.3 Thin-client model

A thin-client model is a type of client-server architecture where all the application processes and data management are performed on the server. The client only provides the presentation with some very simple processing logic. In such type of model clients are heavily depend on the server.

Thin client model is usually used in the areas where there is no many data processing or computations logic, because it "places a heavy processing load on both server and the network" [Balci 2009c].

1.1.1.4 Thick-client model

Thick client model is the opposite of the thin-client model, i.e. in case of thick-client model servers are only responsible for data management and clients are responsible for presentation layer and application logic [Balci 2009c]. In this type of model most of the processing is delegated to the client. Therefore it can cause problems in data management especially if software system is rather complex.

1.1.2 Distributed Objects Architecture

Distributed Objects Architecture (DOA) describes a collection of objects that are interconnected and can be located on multiple computers and connected via the network or inside the same computer. In DOA there is no distinction between clients and servers. Any object of the system may provide services or consume services provided by other objects. In this type of architecture, objects are communicated via the middleware system called an Object Request Broker (ORB) (software bus) [Balci 2009c].

Figure 4 shows the DOA example.

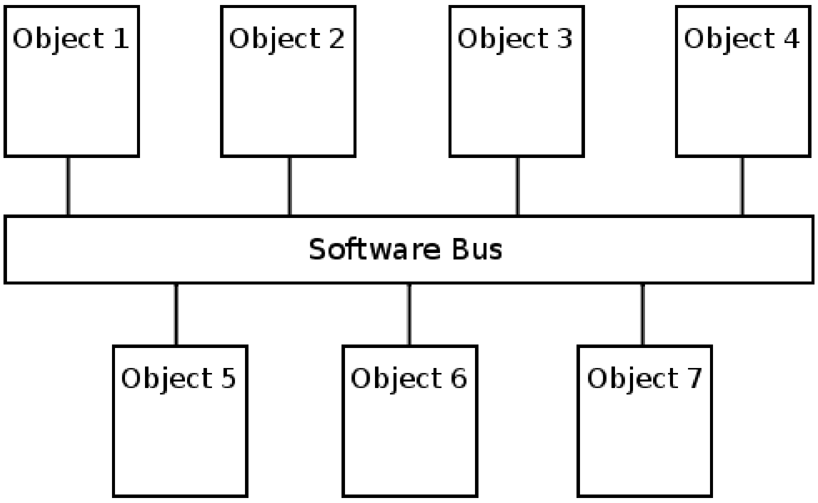


Figure 4. Example of Distributed Objects Architecture.

The advantages of such an architecture are that the system is very flexible and scalable. The system which is architected using DOA becomes very open and new resources can be easily added to it as required. Also, during design stage decisions on where and how services should be provided can be delayed.

From the logical prospective, DOA allows you to think and organize the architecting system in terms of services and combination of services. So, in general, DOA is very similar to client-server architecture. The main distinction is that clients and servers are distributed objects and are communicated through software bus.

Today there are a lot of frameworks that can help you to build software systems using DOA. But Common Object Request Broker Architecture (CORBA) is the leading standard.

1.1.2.1 Common Object Request Broker Architecture (CORBA)

The CORBA standard is defined by Object Management Group (OMG) [OMG 2010]. It enables software components running on different platforms and machines and written on different languages to interoperate with each other. OMG provides only specification which is implemented by different vendors. There are many implementations of CORBA specification; some of them are commercial, while others are open source.

CORBA was developed mostly for enterprise and therefore standardizes a lot of fundamental services, e.g. for banking, insurance and e-commerce. CORBA also specifies such fundamental services as directories and security management.

The CORBA system consists of an ORB, which manages requests to object services. It manages what objects provide what services, route calls from one object to another and then return responses to the requesters.

All objects that provide services should specify their interfaces using common Interface Definition Language (IDL). And then specify a mapping from IDL to a specific programming language, such as C++ or Java.

There is usually more than one computer in a distributed system. And each of these computers will have its own ORB. Figure 5 shows the example of inter-ORB communication which is used for distributed object calls.

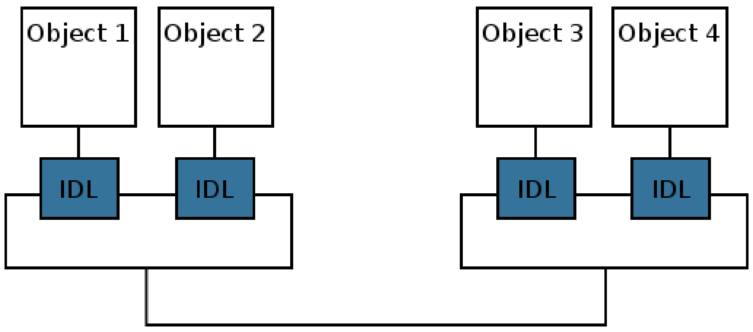


Figure 5. Example of inter-ORB communication

1.1.3 Peer-to-Peer Architecture

The peer-to-peer architecture is a type of architecture where each node has equivalent capabilities and responsibilities and there is no need for central coordination. This differs peer-to-peer architecture from client-server architecture where some workstations are serving others.

When peer-to-peer architecture is used, information is not gathered in one location. Instead, it is spread across the nodes of a distributed system. Figure 6 shows the example of a peer-to-peer architecture. If, for example, user in “Node 1” wants to share any document with others, there is no need to publish it on some particular server. It can be placed in “Node 1” and then will be distributed among other nodes.

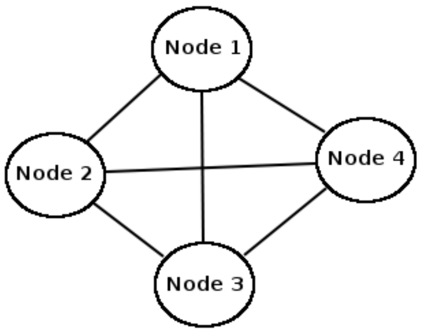


Figure 6. The example of peer-to-peer architecture

There are two different logical peer-to-peer network architectures:

- Decentralized peer-to-peer architecture
- Semi-centralized peer-to-peer architecture

Figure 7 shows the example of decentralized peer-to-peer architecture. And Figure 8 provides the example of semi-centralized peer-to-peer architecture.

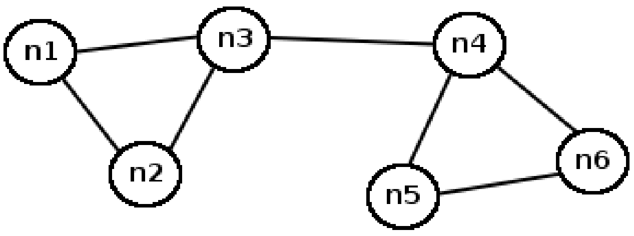


Figure 7. The example of decentralized peer-to-peer architecture

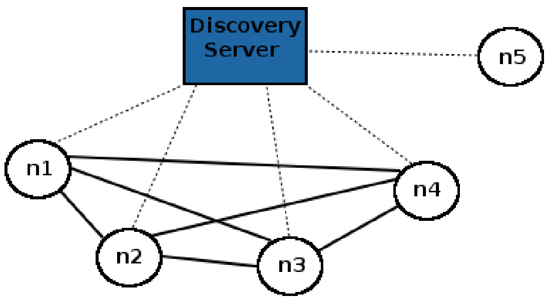


Figure 8. The example of semi-centralized peer-to-peer architecture

The difference is that in the semi-centralized peer-to-peer architecture you can see a discovery server which is used by all other nodes to find out what other nodes are available [Balci 2009c].

The most common examples of peer-to-peer architecture are:

- File sharing systems
- Instant messaging systems

1.1.4 Service-Oriented Architecture

Service-Oriented Architecture (SOA) is a type of architecture which is composed of loosely coupled services and addresses the requirements of standard based and protocol-independent distributed computing.

The SOA is a "design philosophy" and is independent from any specific technology. This is achieved by limiting the number of implementation restrictions to the level of the service interface [Papazoglou and Heuvel 2007]. Services are SOA building blocks. They are well defined, self-contained modules [Fremantle et al 2002]. The fundamental intent of a service is to provide a reusable unit of business functionality and to be independent from the state or context of other services.

SOA requires that services to be defined by a description language and have a published interface that can be discovered and used by the consumers. In general, each service in SOA is a bound pair of a service interface and its implementation [Papazoglou and Heuvel 2007].

Figure 9 shows the way services are discovered and consumed.

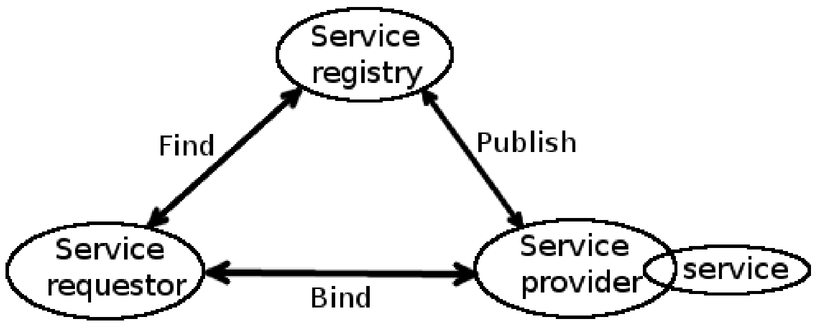


Figure 9. SOA working model

Interoperability is one of SOA characteristics. It means that services can be implemented in different platforms and programming languages and still are expected to be able to work with each other. Service should be *discoverable*, meaning that it should publish its definition in a service registry (service directory) which can be easily found and used by service consumers.

Service should be *loosely coupled*, i.e. it should be independent and try to minimize dependencies.

The advantages of SOA are services *composition* and *reusability*. It is expected that services can be easily reused and combined for more complex business functionalities or other services. Therefore it is important to define services in an appropriate level of granularity.

SOA does not require to be implemented using a specific technology. For example, CORBA [OMG 2010], DCOM [Microsoft 2007] or Web Services [W3C 2006] can be used. However, web services is the most commonly used technology nowadays.

1.2 Statement of the Problem

Software-based system architecture has been recognized as a foundation laying out the underpinnings that are critically important for successful engineering of large-scale complex systems. In recent years, architecting has played a more crucial role in engineering network-centric system of systems.

As the complexity of the architecture of network-centric software-based system of systems has increased, the description of such architecture has posed significant technical challenges. The U.S. Department of Defense (DoD) has developed the DoD Architecture Framework [DoDAF 2009a, DoDAF 2009b] for describing system architectures. IEEE proposes a Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE 2000]. SEI provides high-level guidelines for Documenting Software Architectures [Clements *et al* 2003]. However, all of the diagrams proposed by DoD, IEEE, and SEI are two-dimensional static graphical and textual representations that do not reveal the dynamic characteristics of a system architecture.

1.3 Statement of Objectives

The primary objective of the research is to develop a Conceptual Framework (CF) for specifying the architecture of a network-centric software-based system of systems. The CF should enable the automatic generation of a visual simulation model representing a system architecture.

The CF is the beginning part of a larger research effort. The main goal of the overall research is to employ the automation-based software paradigm and to automatically generate a visual simulation model of a system architecture, with which experiments can be conducted to assess the dynamic characteristics of that architecture.

The secondary objective is to evaluate the proposed CF in half a dozen case studies and demonstrate that it provides the necessary elements for automatic generation of a simulation model as the description of a complex system of systems architecture.

1.4 Overview of Thesis

This thesis is organized as follows. Chapter 2 contains a literature review about the most recognizable and widely used architectural frameworks, their concepts and ideas. Chapter 3 describes the proposed CF for specification and simulation of network-centric system architectures. Chapter 4 shows the applicability of the proposed CF by describing half a dozen examples of using it in creating network-centric system architecture specifications and simulation models. Finally, Chapter 5 presents conclusions, states the contributions and describes future work.

CHAPTER 2: RELATED WORK

2.1 Frameworks

This section describes the most widely used and recognizable architecture frameworks. We also think that these architecture frameworks influenced the field of software-based system architecting the most. Therefore it is important to know and understand its concepts.

2.1.1 Zachman Framework

One of the first who raised a question about difficulty of managing information systems was John Zachman [Zachman 1987]. He identified the main contributing factors: the increasing size and the complexity of the systems, and the tendency of systems distribution because of enterprise operations automation.

2.1.1.1 Architecture Framework

Zachman originally explained his framework as an analogy with the building industry. He organized architectural artifacts in a two-dimensional schema. One dimension consists of:

- Planner (Theorist)
- Owner
- Designer
- Builder
- Sub-contractor (Implementer)
- End User (Participant)

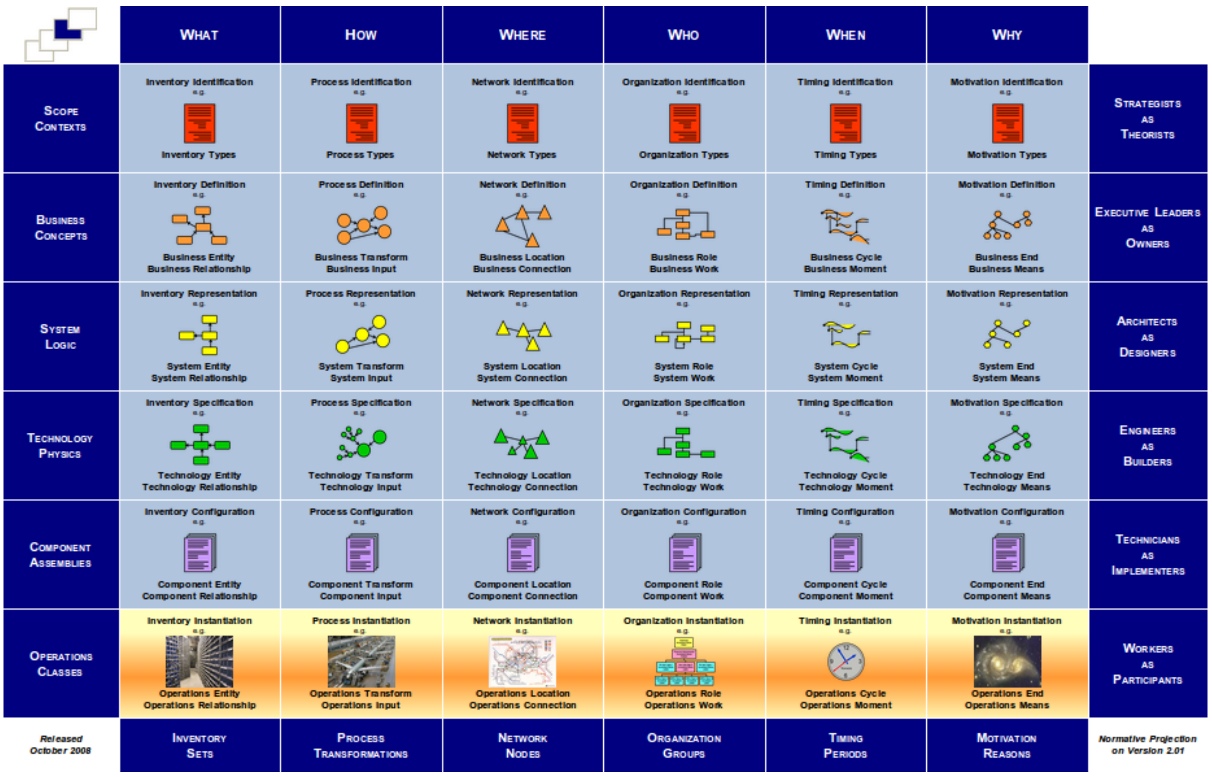
It represents a distinct view of the system during all phases of development. In other words every player requires complete information, but point of view of one player differs for another one. For example, “the owner is interested in a complete description of the functionality and aesthetics of the building. The builder is interested in a complete description of the materials and construction process. The owner doesn’t care about the placement of stubs in the walls. The builder doesn’t care how the bedroom windows line up with the morning sun” [Sessions 2007].

From the IT prospective the owner is interested in a general outlook of what the product will look like. The designer is responsible for the logical view of the system. The builder is creating a physical layout of the system. Sub-contractor implements out-of-context sub components. And the end user has the final product.

The second dimension represents different descriptive focuses of each view. These six focuses are the next: what, how, where, who, when and why.

Each focus is different for each player, i.e. both the planner and the owner need to know *what*, but the planner’s “what” is different from owners “what”. And each of what, how, where, who, when and why depends on who is asking the question [Sessions 2007].

The current version (v2.01) is shown in Figure 10. It consists of 36 cells where the last row (6 cells) represents the final product, while other cells are for the product development.



	WHAT	HOW	WHERE	WHO	WHEN	WHY	
SCOPE CONTEXTS	Inventory Identification e.g. Inventory Types	Process Identification e.g. Process Types	Network Identification e.g. Network Types	Organization Identification e.g. Organization Types	Timing Identification e.g. Timing Types	Motivation Identification e.g. Motivation Types	STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition e.g. Business Entity Business Relationship	Process Definition e.g. Business Transform Business Input	Network Definition e.g. Business Location Business Connection	Organization Definition e.g. Business Role Business Work	Timing Definition e.g. Business Cycle Business Moment	Motivation Definition e.g. Business End Business Means	EXECUTIVE LEADERS AS OWNERS
SYSTEM LOGIC	Inventory Representation e.g. System Entity System Relationship	Process Representation e.g. System Transform System Input	Network Representation e.g. System Location System Connection	Organization Representation e.g. System Role System Work	Timing Representation e.g. System Cycle System Moment	Motivation Representation e.g. System End System Means	ARCHITECTS AS DESIGNERS
TECHNOLOGY PHYSICS	Inventory Specification e.g. Technology Entity Technology Relationship	Process Specification e.g. Technology Transform Technology Input	Network Specification e.g. Technology Location Technology Connection	Organization Specification e.g. Technology Role Technology Work	Timing Specification e.g. Technology Cycle Technology Moment	Motivation Specification e.g. Technology End Technology Means	ENGINEERS AS BUILDERS
COMPONENT ASSEMBLIES	Inventory Configuration e.g. Component Entity Component Relationship	Process Configuration e.g. Component Transform Component Input	Network Configuration e.g. Component Location Component Connection	Organization Configuration e.g. Component Role Component Work	Timing Configuration e.g. Component Cycle Component Moment	Motivation Configuration e.g. Component End Component Means	TECHNICIANS AS IMPLEMENTERS
OPERATIONS CLASSES	Inventory Instantiation e.g. Operations Entity Operations Relationship	Process Instantiation e.g. Operations Transform Operations Input	Network Instantiation e.g. Operations Location Operations Connection	Organization Instantiation e.g. Operations Role Operations Work	Timing Instantiation e.g. Operations Cycle Operations Moment	Motivation Instantiation e.g. Operations End Operations Means	WORKERS AS PARTICIPANTS
Released October 2008	INVENTORY SETS	PROCESS TRANSFORMATIONS	NETWORK NODES	ORGANIZATION GROUPS	TIMING PERIODS	MOTIVATION REASONS	*Normative Projection on Version 2.01*

Figure 10. The Zachman Framework [Zachman 2008] [fair use]

"As we move horizontally in the grid, we see different descriptions of the system – all from the same player's perspective. As we move vertically in the grid, we see a single focus, but change the player from whose perspective we are viewing that focus" [Sessions 2007].

There is some criticism about the framework [Ambler 2007]:

- It is very hard to document everything taking into consideration that there are 36 cells and each cell can contain multiple models
- Too many questions should be asked
- Zachman Framework isn't very well accepted within the development community

As a conclusion it is worth to say that John Zachman laid the foundation for the enterprise architecting frameworks. He was able to highlight the critical issues by asking the right questions and has stimulated the development of other enterprise architecting frameworks, such as DoDAF.

2.1.2 Department of Defense Architecture Framework (DoDAF)

The Department of Defense Architecture Framework (DoDAF) defines the organization of specification of enterprise architecture for all major U.S. Government Department of Defense (DoD) information technology systems.

"The Department of Defense Architecture Framework (DoDAF), Version 2.0 serves as the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate the ability of Department of Defense (DoD) managers at all levels to make key decisions more effectively through organized information sharing across the Department, Joint Capability Areas (JCAs), Mission, Component, and Program boundaries. The DoDAF serves as one of the principal pillars supporting the DoD Chief Information Officer (CIO) in his responsibilities for development and maintenance of architectures". The second version of the DoDAF "provides extensive guidance on the development of architectures supporting the adoption and execution of Net-centric services within the DoD" [DoDAF 2009a].

2.1.2.1 Architecture Framework

The DoDAF has two important terms: DoDAF-described Models and Fit-for-Purpose Views [DoDAF 2009a]:

- "Models are created from the subset of data for a particular purpose. Once the DoDAF-described Models are populated with data, these "views" are useful as examples for presentation purposes, and can be used as needed".
- "Views are user-defined views of a subset of architectural data created for some specific purpose (i.e., "Fit-for-Purpose"). These views are not described or defined in DoDAF, and can be created, as needed, to ensure that presentation of architectural data is easily understood within an agency. This enables agencies to use their own established presentation preferences in their deliberations".

In other words *models* are some kind of templates for organizing and displaying the data in a way which is more understandable for decision makers. After data is collected and presented in such particular way, it is called a *view*. Organized collection of views is called a *viewpoint*, and the collection of all viewpoints is called *Architectural Description*.

The second version of DoDAF groups models in the following eight viewpoints [DoDAF 2009b]:

- All Viewpoint (AV)
- Capability Viewpoint (CV)
- Data and Information Viewpoint (DIV)
- Operational Viewpoint (OV)
- Project Viewpoint (PV)
- Services Viewpoint (SvcV)
- Standard Viewpoint (StdV)
- Systems Viewpoint (SV)

But all models do not have to be created. As it was said, DoDAF is "Fit-for-Purpose" and decision-makers should decide what models should be used. DoDAF is more data-focused as data is the most necessary ingredient for architecture development [DoDAF 2009a]. Table 1 presents the DoDAF viewpoints in a graphic format:

Table 1. Architecture Viewpoints in DoDAF v2.0

Viewpoint Name	Description
All Viewpoint	Overarching aspects of architecture context that relate to all views
Capability Viewpoint	Articulate the capability requirement, delivery timing, and deployed capability
Data and Information Viewpoint	Articulate the data relationships and alignment structures in the architecture content
Operational Viewpoint	Articulate operational scenarios, processes, activities & requirements
Project Viewpoint	Describes the relationships between operational and capability requirements and the various projects being implemented; Details dependencies between capability management and the Defense Acquisition System process
Services Viewpoint	Articulate the performers, activities, services, ant their exchanges providing for, or supporting, DoD functions
Standard Viewpoint	Articulate applicable Operational, Business, Technical, and Industry policy, standards, guidance, constraints, and forecasts
Systems Viewpoint	Articulate the legacy systems or independent systems, their composition, interconnectivity, and context providing for, or supporting, DoD functions

Table 2 provides the list and description of all models in DoDAF v2.0.

Table 2. DoDAF v2.0 Models

Models	Descriptions
AV-1: Overview and Summary Information	Describes a Project's Visions, Goals, Objectives, Plans, Activities, Events, Conditions, Measures, Effects (Outcomes), and produced objects.
AV-2: Integrated Dictionary	An architectural data repository with definitions of all terms used throughout the architectural data and presentations.
CV-1: Vision	The overall vision for transformational endeavors, which provides a strategic context for the capabilities described and a high-level scope.
CV-2: Capability Taxonomy	A hierarchy of capabilities which specifies all the capabilities that are referenced throughout one or more Architectural Descriptions.
CV-3: Capability Phasing	The planned achievement of capability at different points in time or during specific periods of time. The CV-3 shows the capability phasing in terms of the activities, conditions, desired effects, rules complied with, resource consumption and production, and measures, without regard to the performer and location solutions.
CV-4: Capability Dependencies	The dependencies between planned capabilities and the definition of logical groupings of capabilities.

CV-5: Capability to Organizational Development Mapping	The fulfillment of capability requirements shows the planned capability deployment and interconnection for a particular Capability Phase. The CV-5 shows the planned solution for the phase in terms of performers and locations and their associated concepts.
CV-6: Capability to Operational Activities Mapping	A mapping between the capabilities required and the operational activities that those capabilities support.
CV-7: Capability to Services Mapping	A mapping between the capabilities and the services that these capabilities enable.
DIV-1: Conceptual Data Model	The required high-level data concepts and their relationships.
DIV-2: Logical Data Model	The documentation of the data requirements and structural business process (activity) rules. In DoDAF V1.5, this was the OV-7.
DIV-3: Physical Data Model	The physical implementation format of the Logical Data Model entities, e.g., message formats, file structures, physical schema. In DoDAF V1.5, this was the SV-11.
OV-1: High-Level Operational Concept Graphic	The high-level graphical/textual description of the operational concept.
OV-2: Operational Resource Flow Description	A description of the Resource Flows exchanged between operational activities.
OV-3: Operational Resource Flow Matrix	A description of the resources exchanged and the relevant attributes of the exchanges.
OV-4: Organizational Relationships Chart	The organizational context, role or other relationships among organizations.
OV-5a: Operational Activity Decomposition Tree	The capabilities and activities (operational activities) organized in a hierarchal structure.
OV-5b: Operational Activity Model	The context of capabilities and activities (operational activities) and their relationships among activities, inputs, and outputs; Additional data can show cost, performers, or other pertinent information.
OV-6a: Operational Rules Model	One of three models used to describe activity (operational activity). It identifies business rules that constrain operations.
OV-6b: State Transition Description	One of three models used to describe operational activity (activity). It identifies business process (activity) responses to events (usually, very short activities).
OV-6c: Event-Trace Description	One of three models used to describe activity (operational activity). It traces actions in a scenario or sequence of events.
PV-1: Project Portfolio Relationships	It describes the dependency relationships between the organizations and projects and the organizational structures needed to manage a portfolio of projects.
PV-2: Project Timelines	A timeline perspective on programs or projects, with the key milestones and interdependencies.
PV-3: Project to Capability Mapping	A mapping of programs and projects to capabilities to show how the specific projects and program elements help to achieve a capability.
SvcV-1: Services Context Description	The identification of services, service items, and their interconnections.
SvcV-2: Services Resource Flow Description	A description of Resource Flows exchanged between services.
SvcV-3a: Systems-Services Matrix	The relationships among or between systems and services in a given Architectural Description.
SvcV-3b: Services-Services Matrix	The relationships among services in a given Architectural Description. It can be designed to show relationships of interest, (e.g., service-type interfaces, planned vs. existing interfaces).
SvcV-4: Services Functionality Description	The functions performed by services and the service data flows among service functions (activities).

SvcV-5: Operational Activity to Services Traceability Matrix	A mapping of services (activities) back to operational activities (activities).
SvcV-6: Services Resource Flow Matrix	It provides details of service Resource Flow elements being exchanged between services and the attributes of that exchange.
SvcV-7: Services Measures Matrix	The measures (metrics) of Services Model elements for the appropriate time frame(s).
SvcV-8: Services Evolution Description	The planned incremental steps toward migrating a suite of services to a more efficient suite or toward evolving current services to a future implementation.
SvcV-9: Services Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future service development.
SvcV-10a: Services Rules Model	One of three models used to describe service functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SvcV-10b: Services State Transition Description	One of three models used to describe service functionality. It identifies responses of services to events.
SvcV-10c: Services Event-Trace Description	One of three models used to describe service functionality. It identifies service-specific refinements of critical sequences of events described in the Operational Viewpoint.
StdV-1: Standards Profile	The listing of standards that apply to solution elements.
StdV-2: Standards Forecast	The description of emerging standards and potential impact on current solution elements, within a set of time frames.
SV-1: Systems Interface Description	The identification of systems, system items, and their interconnections.
SV-2: Systems Resource Flow Description	A description of Resource Flows exchanged between systems.
SV-3: Systems-Systems Matrix	The relationships among systems in a given Architectural Description. It can be designed to show relationships of interest, (e.g., system-type interfaces, planned vs. existing interfaces).
SV-4: Systems Functionality Description	The functions (activities) performed by systems and the system data flows among system functions (activities).
SV-5a: Operational Activity to Systems Function Traceability Matrix	A mapping of system functions (activities) back to operational activities (activities).
SV-5b: Operational Activity to Systems Traceability Matrix	A mapping of systems back to capabilities or operational activities (activities).
SV-6: Systems Resource Flow Matrix	Provides details of system resource flow elements being exchanged between systems and the attributes of that exchange.
SV-7: Systems Measures Matrix	The measures (metrics) of Systems Model elements for the appropriate timeframe(s).
SV-8: Systems Evolution Description	The planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation.
SV-9: Systems Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future system development.
SV-10a: Systems Rules Model	One of three models used to describe system functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SV-10b: Systems State Transition Description	One of three models used to describe system functionality. It identifies responses of systems to events.
SV-10c: Systems Event-Trace Description	One of three models used to describe system functionality. It identifies system-specific refinements of critical sequences of events described in the Operational Viewpoint.

2.1.3 The Open Group Architecture Framework (TOGAF)

The Open Group Architecture Framework (TOGAF) is an open source Enterprise Architecture Framework and represents a detailed method and a set of supporting tools for developing enterprise architecture [THE OPEN GROUP 2009a]. It is continuously evolved since 1995 and initially was based on Technical Architecture Framework for Information Management (TAFIM).

2.1.3.1 Architecture Framework

TOGAF is designed to support four architecture domains. A brief description of all of them is given below [THE OPEN GROUP 2009b]:

1. ***Business Architecture:*** Defines the business strategy, governance, organization and key business processes.

2. ***Data Architecture:*** Describes the structure of an organization's local and physical data assets and data management resources.

3. ***Application Architecture:*** Provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization.

4. ***Technology Architecture:*** Describes the logical software and hardware capabilities that are required to support the deployment of business, data, and application services. This includes IT infrastructure, middleware, networks, communications, processing, standards, etc.

TOGAF consists of the following components:

- Architecture Development Model (ADM)
- Architecture Content Framework
- Enterprise Continuum
- Architecture Capability Framework

2.1.3.1.1 Architecture Development Model

The ADM is a methodology for developing an organization-specific Enterprise Architecture that addresses business requirements and form the core of TOGAF.

The ADM consists of nine phases [THE OPEN GROUP 2009c]:

1. ***Preliminary Phase:*** Prepares and initiates activities required to meet the business directive for new enterprise architecture.

2. ***Phase A: Architecture Vision:*** Defines the scope, identifies the stakeholders and creates a comprehensive plan that addresses scheduling, resourcing, financing, communication, risks, constraints, assumptions and dependencies.

3. ***Phase B: Business Architecture:*** Describes the Baseline and develops Target Business Architectures, analyzes the gaps between them.

4. ***Phase C: Information System Architectures:*** Develops Target Architecture of the data and application systems domain.

5. ***Phase D: Technology Architecture:*** Create an overall target architecture which then will be implemented.

6. ***Phase E: Opportunities and Solutions:*** Develops an overall strategy, determining what will be bought, built and reused and how architecture will be implemented.

7. ***Phase F: Migration Planning:*** Creates a viable implementation and migration plan.

8. ***Phase G: Implementation Governance:*** Provides an architectural oversight of the implementation.

9. ***Phase H: Architecture Change Management:*** Manages changes to the new architecture and ensures that the architecture achieves its original target business value.

2.1.3.1.2 Architecture Content Framework

The Architecture Content Framework describes the outputs which are produced during executing of the ADM. “The content framework provides a structural model for architectural content that allows the major work products that an architect creates to be consistently defined, structured, and presented” [THE OPEN GROUP 2009e].

The Architecture Content Framework uses the following three categories for the description of the architectural work products [THE OPEN GROUP 2009e]:

1. ***Deliverable:*** A work product that is specified and reviewed by the stakeholders. Represents the output of the project or “those deliverables that are in documentation form will typically be archived at completion of a project, or transitioned into an Architecture Repository as a reference model” [THE OPEN GROUP 2009e].

2. ***Artifact:*** Describes the architecture from a specific viewpoint. Can be classified as a catalog, matrix or diagram.

3. ***Building Block:*** “Represents a (potentially re-usable) component of business, IT, or architectural capability that can be combined with other building blocks to deliver architectures and solutions” [THE OPEN GROUP 2009e].

2.1.3.1.3 Enterprise Continuum

"The simplest way of thinking of the Enterprise Continuum is as a view of the repository of all the architecture assets. It can contain architecture descriptions, models, building blocks, patterns, viewpoints, and other artifacts - that exist both within the enterprise and in the IT industry at large, which the enterprise considers to have available for the development of architectures for the enterprise" [THE OPEN GROUP 2009d].

The Enterprise continuum is the repository for all artifacts which are created while using ADM. "The enterprise architecture determines which architecture and solution artifacts an organization includes in its Architecture Repository. Re-use is a major consideration in this decision" [THE OPEN GROUP 2009d]. Therefore not only artifacts which were developed just for current enterprise architecture can be placed in and taken from the Architecture Repository, but the solution artifacts of previous architecture work can also be used.

Enterprise Continuum by itself also includes:

1. ***Architecture Continuum:*** "Offers a consistent way to define and understand the generic rules, representations, and relationships in an architecture, including traceability and derivation relationships (e.g., to show that an Organization-Specific Architecture is based on an industry or generic standard). Shows the relationships among foundational frameworks (such as TOGAF), common system architectures, industry architectures, and enterprise architectures. The Architecture Continuum is a useful tool to discover commonality and eliminate unnecessary redundancy" [THE OPEN GROUP 2009d].

2. ***Solutions Continuum:*** "Provides a consistent way to describe and understand the implementation of the assets defined in the Architecture Continuum. Defines what is available in the organizational environment as re-usable Solution Building Blocks. The Solutions Continuum addresses the commonalities and differences among the products, systems, and services of implemented systems" [THE OPEN GROUP 2009d].

2.1.3.1.4 Architecture Capability Framework

The Architecture Capability framework describes the appropriate organization structures, processes, roles, responsibilities and skills to realize the architecture capability. But it does not position itself as a "comprehensive template for operating an enterprise architecture capability" [THE OPEN GROUP 2009f].

Within the Architecture Capability Framework TOGAF requires the design of four domain architectures [THE OPEN GROUP 2009f]:

1. ***Business Architecture:*** Highlights the architecture governance, architecture processes, architecture organizational structure, architecture information requirements, architecture products, etc.

2. ***Data Architecture:*** Defines the structure of the organization's Enterprise Continuum and Architecture Repository.

3. ***Application Architecture:*** Specifies the functionality and/or applications services required to enable the architecture practice.

4. ***Technology Architecture:*** Depicts the architecture practice's infrastructure requirements and deployment in support of the architecture applications and Enterprise Continuum.

2.1.4 Institute of Electrical and Electronics Engineers (IEEE) 1471 Standard

IEEE 1471 is an IEEE Standard for specifying the architecture of software-intensive systems. It is also known as ANSI/IEEE 1471-200, Recommended Practice for Architectural Description of Software-Intensive Systems. It was developed to provide a foundation for architecture specification of software-intensive systems. The standard is focused on “any system in which software development and/or integration are dominant considerations (i.e., most complex systems nowadays). This includes computer-based systems ranging from individual software applications, information systems, embedded systems, software product lines and product families and systems-of-systems” [IEEE 2000]. In March 2006 IEEE 1471 was adopted by International Organization for Standardization (ISO) as an international standard and now it is known as ISO/IEC 42010:2007.

The next section describes the conceptual framework of IEEE 1471 Architectural Description (AD).

2.1.4.1 Conceptual Framework

2.1.4.1.1 Architectural Description

Each system has one or more stakeholders. And each stakeholder has his/her interests and concerns which are relative to the system. “Concerns are those interests which pertain to the system’s development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders” [IEEE 2000].

The standard states that “every system has an architecture, in terms of its recommended practice” [IEEE 2000]. And that architecture can be recorded by an AD which is provided in Figure 11.

By IEEE 1471, AD is “a collection of products to document an architecture”. It is organized into one or more *views* of a system. “Each view addresses one or more of the concerns of the system stakeholders” [IEEE 2000]. In other words a view is “a representation of a whole system from the perspective of a related set of concerns” [Hilliard 2007].

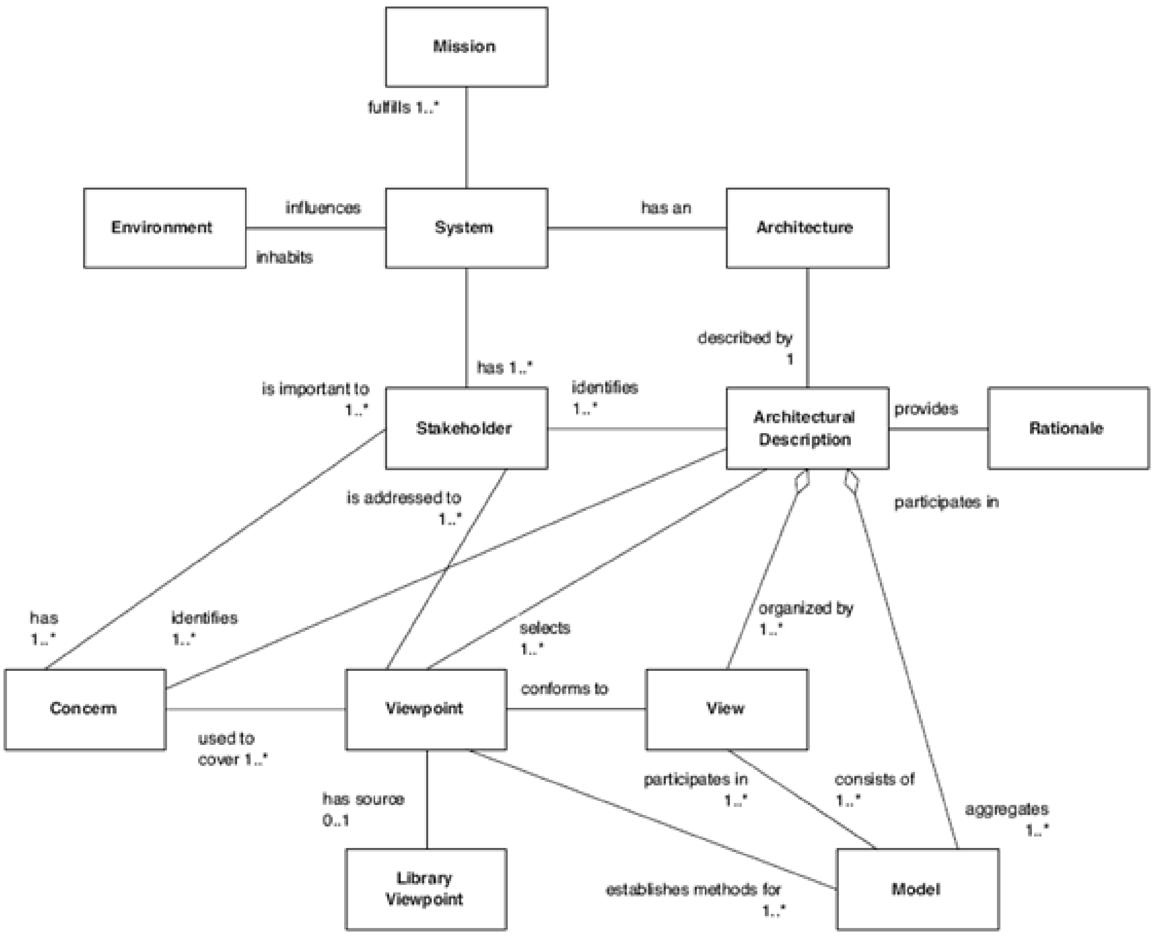


Figure 11. Conceptual Model of IEEE 1471 architectural description [IEEE 2000] [fair use]

View represents a system's architecture "with respect to a particular viewpoint" [IEEE 2000]. Where *viewpoint* – is "a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis" [IEEE 2000]. Therefore viewpoints dictate how and what corresponding views represent. Also, each view is associated with exactly one viewpoint. The reason for such mapping is in a conceptual consistency: "two views for the same viewpoint would cover the same concerns, raising consistency issues between the two views" [Hilliard 2007].

While specifying AD stakeholders should decide what concerns they have and to whom AD is addressed. After this "an architectural description selects one or more viewpoints for use" [IEEE 2000]. In such way concerns select what particular viewpoints should be used.

Each view may contain more than one architectural model which in their turn may participate in more than one view.

2.1.4.1.2 Stakeholders

Stakeholders can have different roles depending on the use of architectural descriptions. But standard evidently specifies two key roles: the acquirer (or client) and the architect. The architect develops and maintains an architecture for a system to satisfy the acquirer.

2.1.4.2 IEEE 1471 Requirements and Unresponsiveness

The AD must contain at least the following [Hilliard 2007]:

- Stakeholders and concerns
- Selection and declaration of the used viewpoints
- Architectural views, where each relates to a particular viewpoint
- All AD inconsistencies
- Architectural explanation

But the standard doesn't care about [Hilliard 2007]:

- What format is used for AD
- The language which is used for AD
- The process of creation and evaluation of AD

2.1.5 Software Engineering Institute (SEI) Views and Beyond Approach

In 2003 Software Engineering Institute of the Carnegie Mellon School of Computer Science presented their answer to the question: "How should you document an architecture so that others can successfully use it, maintain it, and build a system from it?" – approach which was called "Views and Beyond" (V&B) [Clements *et al* 2003].

2.1.5.1 Core Concepts

View is one of the core concepts of V&B approach. View is "a representation of a set of system elements and relationships among them" [Clements *et al* 2003]. It is not stated how many views should be specified. On the contrary, the authors advocate that you can create as many views as you need. The number of views should depend on the goals.

This philosophy leads to the fundamental principle of V&B approach:

"*Documenting an architecture is a matter of documenting the relevant views, and then adding documentation that applies to more than one view*" [Clements *et al* 2003].

Clements *et al* suggest including the next documentation for each view:

- a primary representation that shows the primary elements
- an explanation and definition of each element, which is included in view, and their properties

- the elements interface specification and the behavior description
- rationale and design information

Viewtype is the second core concept. While there is no restriction on how many views should be created, V&B approach has only three viewtypes. And each view falls into one of the next categories:

- the *module* viewtype – describes how the system should be structured as a set of system's principal units
- the *component-and-connector* (C&C) viewtype – describes system's unit of execution, i.e. what is the structure of interacting runtime elements
- the *allocation* viewtype – specifies how relationships between a system's software and its development and execution environments should be documented

Style is the third and the last core concept of the V&B approach. "Style is described as a set of element and relation types, together with a set of constraints on how they can be used" [Clements *et al* 2003]. Style is a viewtype specialization that represents repeated patterns which are independent from any particular system.

2.1.5.2 Documenting Software Architecture

The second part of the V&B approach describes how all selected views should be specified and tied together to provide the complete picture of the architecture. Along with different suggestions, such as "how to document system behavior?", a short guideline of how views for the architecture description should be chosen is provided. The guideline consists of the following three steps:

1. Make a table with stakeholders and views and specify how much information each stakeholder needs for each view

2. Reduce the number of views by removing unnecessary view and combining those views which has similarities

3. Prioritize the order of views creation

The authors also provide the template for documenting software architecture using V&B approach.

2.2 Tools

This section describes some of the enterprise architecture tools that are used by major commercial companies and government agencies in the software based architecting processes.

2.2.1 Telelogic System Architect

Telelogic System Architect is a software based solution that works in five key domains [IBM 2010]:

- Strategy
- Business
- Information
- Systems
- Technology

It is very famous in enterprise architect community for its ability to analyze, visualize and communicate organization's enterprise architectures and business process analysis [IBM 2010]. System Architect supports all most important and commonly used Enterprise Architecture Frameworks: Zachman, TOGAF, DoDAF, etc.

System Architect is very useful when used with DoDAF. It provides comprehensive support for all DoDAF Views and Models. It also helps to automate the process and reuse the data as much as possible. For example, if you have already created one DoDAF view and some data of this view can be reused in another DoDAF view, System Architect will reuse it and build another view automatically.

All modeling notations of TOGAF are also supported. Moreover, System Architect helps to understand what will be affected if you make some changes by providing corresponding graphics and indicating the results on your diagrams and matrices [IBM 2010].

System Architect provides simulation capabilities but only for various business process diagrams. It does not provide such capabilities for system architecture diagrams.

But nevertheless it is one of the best solutions which should be used when you are working with enterprise architecture frameworks which were mentioned above.

2.2.2 Metastorm ProVision

Metastorm ProVision is another software based solution for enterprise architecting that "enables you to model, visualize and improve your entire enterprise – resulting in a thorough understanding of the cause-effect relationships between business strategy, business processes and the system and technology that support them" [Metastorm 2008]. It positions itself as an end-to-end solution for Enterprise Architecture and Business Process Analysis.

Metastorm ProVision also supports the main Enterprise Architecture frameworks and business modeling languages, such as Zachman, TOGAF, DoDAF, UML, BPMN-BPEL. The same as System Architect, Metastorm ProVision helps you to simplify the knowledge reuse and exchange by providing online collaboration servers and repositories.

It also provides the possibility of simulation, but only for business process scenarios too. It provides Monte Carlo and discrete event simulators [Metastorm 2008].

The distinct feature of Metastorm Provision is that it is very customizable. Along with enterprise frameworks and modeling languages that are already included, it allows creating your own personalized framework or modeling language.

CHAPTER 3: THE PROPOSED CONCEPTUAL FRAMEWORK

In this chapter we describe the important concepts and elements of the proposed conceptual framework. The elements can be grouped in two categories: static structure and dynamic structure. Static structure presents static view of the architecture, whereas dynamic structure presents dynamic constituent of the conceptual framework.

3.1 Static Structure

Elements of static structure represent the static view of system architecture. By one of the definitions the architecture is [IEEE 2000]:

"The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution".

In our conceptual framework we propose to use the following components, i.e. a set of building blocks which can be used for developing network-centric architectures:

- Component
- Connection
- Service

These elements are described below.

3.1.1 Component

3.1.1.1 Overview

Typical network-centric system consists of a set of elements that are:

- Independent, and
- Communicated over the network.

These properties are the main difference between monolithic and network-centric software systems. Moreover, most often the elements of the network-centric systems are developed independently and systems emerge only through the interaction of their elements.

3.1.1.2 Defining a component

Components are making up the system being architected. The question is that: how to identify those components and what attributes do the components have.

In our conceptual framework we propose to use top-down approach. By this approach, first, top components should be identified. Further, these components can be decomposed into other components. If component is decomposed, it is called deep, otherwise it is called shallow.

Such component decomposition approach is very similar to modular programming where each module is independent and represents a single unit of functionality. In other words modular and our approaches possess loose coupling and high cohesion properties.

The component should satisfy the following four criteria [Dhanji 2009]:

- Whole – a component is a complete unit of responsibility. With respect to an architecting system, this means that components can be picked up and dropped in as needed.
- Independence – component does not have dependencies on other components to perform its core function. Apart from some common components, a component can be architected, designed, developed and tested independently.
- Contractually – a component conforms to well-defined behavior and can be relied on to behave as expected under all circumstances.
- Separate – a component is not invasive of collaborators, and thus it is a discrete unit of functionality.

If any component violates any of these criteria, such component should be divided into two or more components to satisfy them. After components on a particular level of decomposition have been defined, each of them can be decomposed further. Figure 12 shows the example of such decomposition.

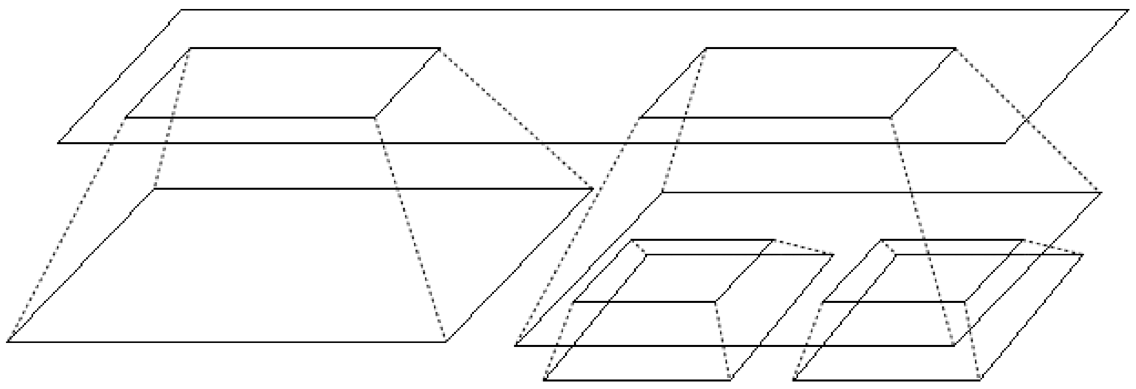


Figure 12. Components decomposition

We believe that such approach enforces an architect to maintain the important fundamentals which should "never go out of style" [Booch 2009]:

- Crisp abstraction
- Clear separation of concerns
- Balanced distribution of responsibilities
- Simplicity

3.1.1.3 Component properties

Table 3 contains the summary of component properties.

Table 3. Summary of component properties

Property Name	**Property Description**
Name	Component’s name which reflects its functionality.
Description	Component’s description.
Services provided	List of services which current component provides.
Services requested	List of services other components provide and which current component uses.

If component is deep, i.e. it is decomposed, then for each subcomponent of this decomposed component, “services requested” property can contain only services which are defined in the “services requested” property of the decomposed component or which are defined in the “services provided” property of the decomposed component or its subcomponents.

At the same time “services provided” property of each subcomponent of the decomposed component can contain only services which are defined in the “services provided” property of the decomposed component.

And you can see that the decomposition stops when you come to the point when each component provides only one service. Figure 13 shows the example of components properties.

On the top level there are two components, “component 1” and “component 2”. “Component 1” provides “service 3” and “service 4” services and requests “service 1” and “service 2” services. “Component 1” is decomposed to “component 3” and “component 4”. “Component 3” can only provide services which are listed in the “services provided” property of “component 1”. Therefore, “component 3” provides, for example, “service 3” service. The same situation is with “services requested” property, thus “component 3” requests “service 1” and “service 2” services. “Component 4” provides “service 4” and requests “service 1”. As each of the components: “component 3” and “component 4”, provides only one service, then decomposition of these components is finished.

“Component 2” provides “service 1” and “service 2” services and does not request any services. As “component 2” is decomposed to “component 5” and “component 6”, then “component 5” provides “service 1” service and “component 6” provides “service 2” service. As “component 2” does not request any services, then “component 5” and “component 6” also cannot request any services. “Component 5” and “component 6” provide only one service each, therefore decomposition for these components is also finished.

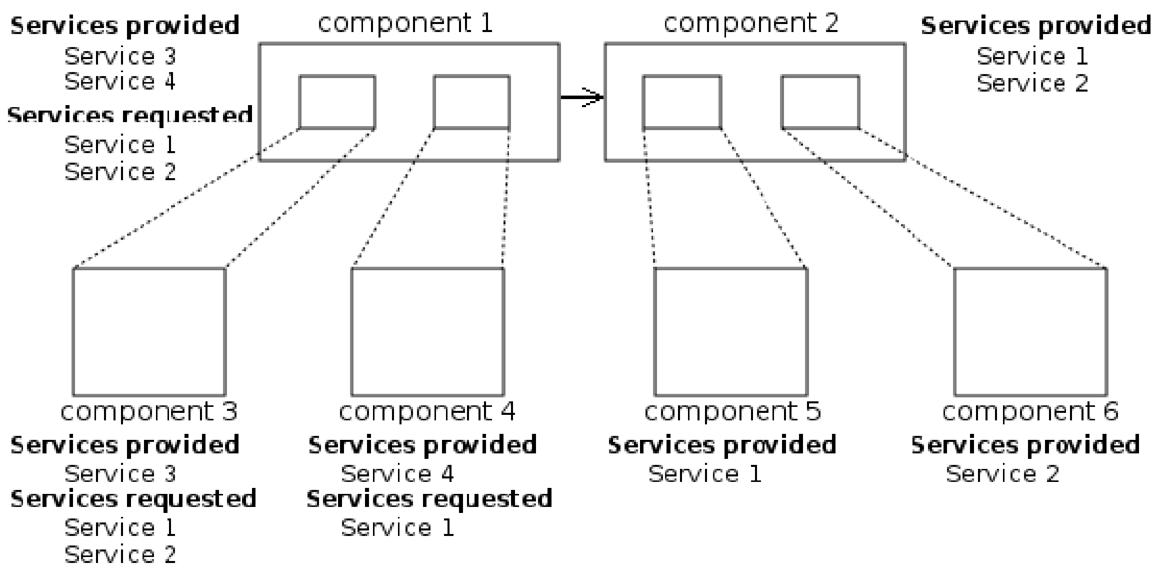


Figure 13. Example of components properties

3.1.2 Connection

3.1.2.1 Defining a connection

Network-centric software systems are distributed by their nature and very intensively rely on network communication. In our conceptual framework “connection” element represents interaction between components. Figure 14 shows the example of using “connection” element.

In Figure 14 “component 1” interacts with “component 2” and “component 3”. Therefore there are connection elements between “component 1” and “component 2”, and “component 1” and “component 3”. “Component 2” also interacts with “component 1”. Thus there is another one connection element which connects them.

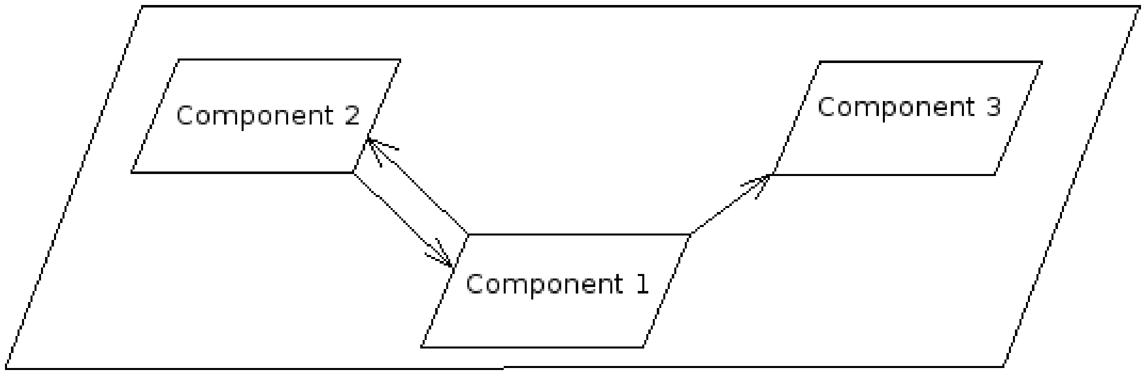


Figure 14. Example of connected components

3.1.2.2 Connection properties

"While interactions may be as simple as procedure calls, often they represent more complex forms of communication and coordination" [Monroe 1997]. We believe that the following properties of components interactions are the most important and should be presented in software architecture specifications:

- protocol name
- types of communication

Table 4 describes all connection properties in detail.

Table 4. Summary of connection properties

Property Name	**Property Description**	**Property Values**
Protocol Name	Name of a protocol to be used for components interaction	HTTP, SOAP, etc.
Types of Communication	Types of communication which is used to communicate between components	A combination of Request, Response or Notification values.

"Types of communication" property defines how two components interact with each other. When one component interacts with another one it can send request. And after request has been sent, it would or would not expect response. In the first case there will be request-response interaction and component which sends request expects to get response. This is the most common way of components interaction. In the second case component just sends notification and does not expect any response from the components. Figure 15 shows example which explains such behavior.

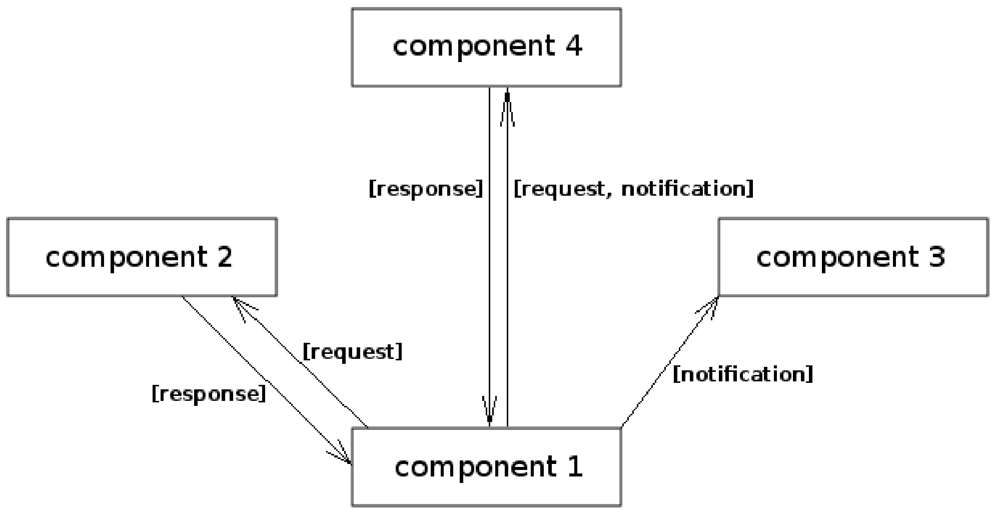


Figure 15. Example of components connection properties

In Figure 15 “component 1” interacts with “component 2”. And we can see that there is one connection between “component 1” and “component 2” with “types of communication” property which contains “request” and another connection from “component 2” to “component 1” with “types of communication” property which contains “response”. We can also see the connection between “component 1” and “component 3” with “types of communication” property which contains “notification”, i.e. “component 1” sends only notifications to “component 3” and does not expect any responses. But we also can see that “component 1” interacts with “component 4”. And one connection has “types of communication” property which contains “request” and “notification”, and another connection has “types of communication” property which contains “response”.

As it was said, request-response type of interaction is the most common way of components interaction. And if you start to show such type of interactions as two different interactions (one for request and one for response), it will make lots of excessiveness. Therefore we can state that if one component interacts with another component and sends requests to it, it will expect to receive responses. That is why we can show only interactions between the components which have such “type of interaction” as “request”, request and notification or just notification. Moreover, we can also skip the label of connection element if its type is only “request”. With such modifications representation of components interaction properties which are described in Figure 15 can be changed.

Figure 16 represents the updated view of components connection properties.

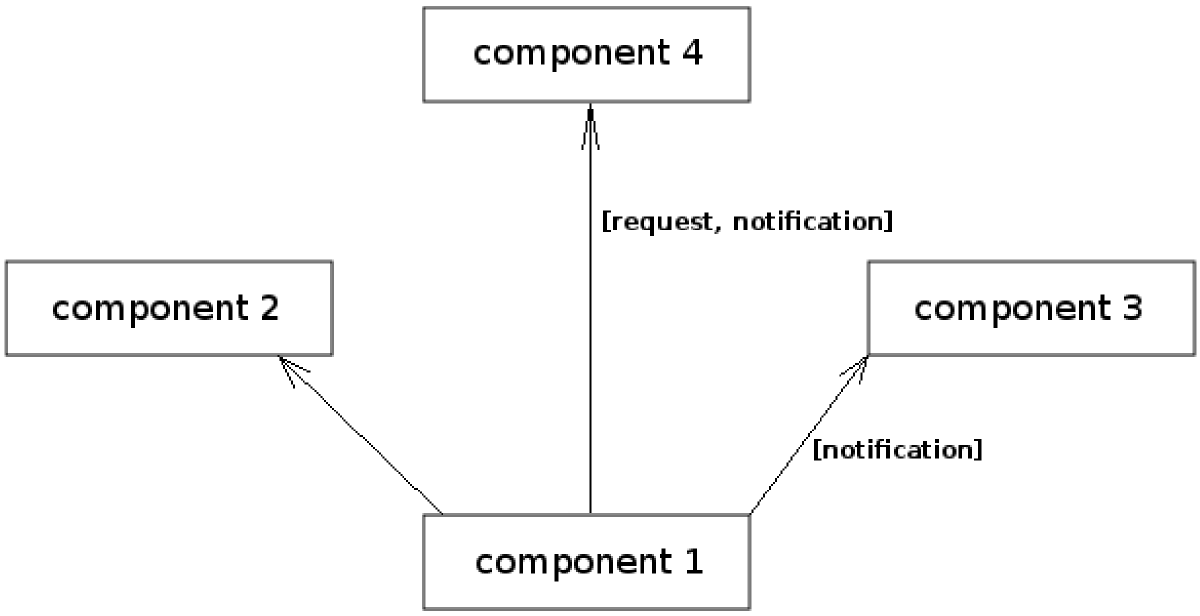


Figure 16. Modified example of components connection properties

3.1.3 Service

3.1.3.1 Defining a service

Similar to components, services are fundamental "building blocks that allow architects to organize capabilities in ways familiar to them" [Microsoft 2004].

Capabilities are another very broad and important research topic. There is still no consensus about its definition and how capabilities should be defined. [Ravichandar 2008] defines capabilities in the next way:

"Capabilities embody the software engineering characteristics of high cohesion, minimal coupling and balanced abstraction levels. By incorporating such desirable attributes in a phase that even precedes the formalization of requirements, the capability engineering process influences the structure of the system to be developed."

In network-centric systems component capabilities are delivered and consumed as services. Services should not be associated with web-services in SOA. The primary goal of a service is to represent a "natural" step of business functionality. That is, according to the domain for which it's provided, a service should represent a self-contained functionality that corresponds to a real-world business activity [Josuttis 2007].

Although services are presented as properties of components, not as separate elements, they are still an essential part of the conceptual framework.

3.1.3.2 Open system paradigm

In our conceptual framework we propose to use open system paradigm [Balci 2009a]. It means that if any service is defined, i.e. listed in the "services provided" property of a component, then it is assumed that this service is "visible" and accessible for any other components – can be listed in the "services requested" property of a component which uses this service. Figure 17 presents example of components and services which use open system paradigm.

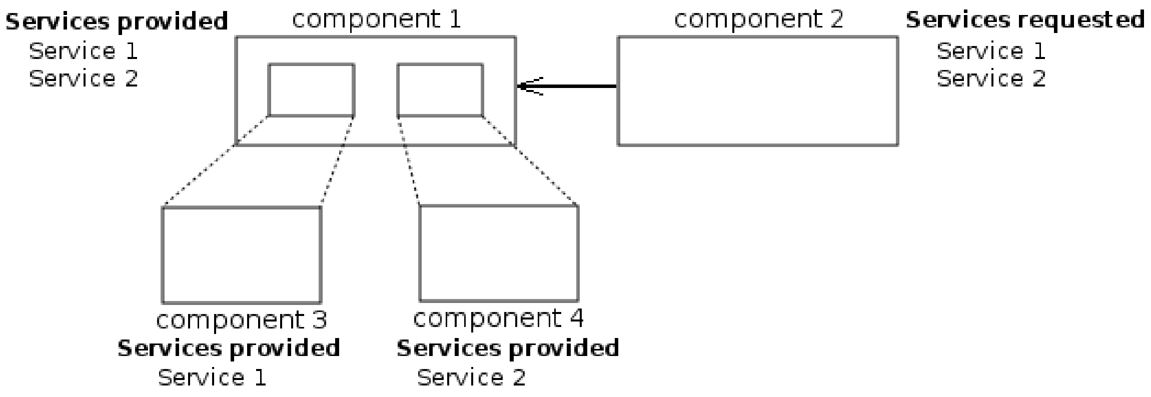


Figure 17. Components and services are defined using open system paradigm

In Figure 17 there are two components which interact with each other: “component 1” and “component 2”. “Component 1” is decomposed to “component 3” and “component 4”. “Component 3” provides one service: “service 1”. “Component 4” also provides one service: “service 2”. As “component 1” is decomposed to “component 3” and “component 4” and they correspondingly provide “service 1” and “service 2”, “component 1” contains those services in “services provided” property. Therefore “component 2” can use and uses these services and that is why they are listed in the “services requested” property of “component 2” component.

3.2 Dynamic Structure

Elements of dynamic structure represent the dynamic behavior of system architecture. Network-centric software systems “focus substantially on their communication elements” [Chigani, Arthur, and Bohner 2007] therefore dynamic structure elements of the conceptual framework show how these communication elements interact with each other.

3.2.1 Message

3.2.1.1 Defining a message

Message element is the only element of the dynamic structure group. It represents a single action of communication between connected components. By single action of communication we mean a remote procedure call or invocation, message passing, etc.

Message element is not used in the architecture specification directly. It can be used to visualize the interaction between the components. After message has been generated it travels from one component to another, until it reaches final component or service. Besides, if component is decomposed, i.e. deep, moving message drops into it and travels inside decomposed component, until it reaches its destination.

It can happen that when message is sent and then reached its destination, for example “service 1”, “service 1” needs to communicate with another service to accomplish its goals, for example, “service 2”. In that case another message is generated and sent from “service 1” to “service 2”.

3.2.1.2 Message properties

Message element has a *type* property which can be one of the next values:

- request
- response
- notification

There can be two ways of communication between components:

- component which has sent request and expects response
- component which has sent request and does not expect response

In the first case message can be sent as request or response, i.e. its type property is correspondingly “request” or “response”. Message is sent as request when one service uses one of the services of another component and expects response from it. After message has been received, receiving service needs to send response. In that case, message with type property “response” is sent.

In the second case message can be sent as notification. It happens when service sends message to another component and does not expect response from it.

We can notice that message element is tightly related to connection element, i.e. messages “type” property is the same as connections “types of communication” property. And it is reasonable, because message elements can travel from one component to another component only if messages type property value is contained in the “types of communication” property value of connection element which connects these two components. For example, message with type “notification” cannot use connection which does not contain “notification” in its “types of communication” property to travel from one component to another one.

CHAPTER 4: EVALUATION OF THE CONCEPTUAL FRAMEWORK

This chapter presents the evaluation of the conceptual framework which was described in Chapter 3. Here we evaluate the applicability of the described Conceptual Framework and provide half a dozen examples of using Conceptual Framework for specifying network-centric system architectures. Also, for each example, we provide simulation models which were built based on the described system architecture specifications.

4.1 The Process of Creating System Architecture Specification and Simulation Model

Before we present examples of network-centric system architecture specifications, we need to describe the process of creating system architecture specification and its transformation to the simulation model.

Figure 18 shows such process.

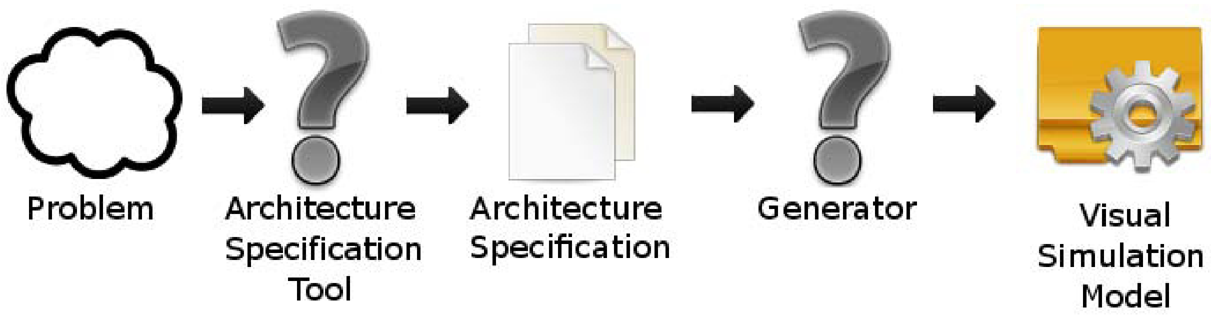


Figure 18. The process of creating system architecture specification and simulation model

At the beginning we have a problem. Then, using some architecture specification tool we create architecture specification. After that, some generator uses this specification to automatically build simulation model. But nowadays, there is no such generator which can get an architecture specification and build a simulation model from it.

In this chapter, to evaluate the applicability of the proposed Conceptual Framework, we have built architecture specification using Conceptual Framework manually. Moreover, we have built architecture specification using HTML [W3C 1999] and JavaScript programming language. Then, for each example we manually created the simulation model, also using proposed Conceptual Framework.

Simulation model is built in the Visual Simulation Environment (VSE) [Orca Computer 2002] program which allows to build manually simulation models of different complexity and to specify programmatically its behavior.

4.2 Examples

In this section we present examples of using Conceptual Framework for creating system architecture specifications and simulation models of network-centric system architectures. The process of creating architecture specifications is the same for each example and the figures of all examples are detailed and self-explanatory. Therefore only the first example is described in detail, while for other examples only figures and short explanations are provided.

4.2.1 Online Banking System

4.2.1.1 Overview

This example describes Online Banking System which provides web access for its users and uses services to communicate with other external devices, for example, ATMs. This example describes SOA of online banking system which allows bank customers to have access to their bank accounts from any place in the world where they have internet connections and allows employees to serve clients using web access.

To sum up, online banking system:

- provides web access to bank customers
- provides web access to bank employees
- supports ATM access to the system

4.2.1.2 System Architecture Specification

Figure 19 presents the top view of the Online Banking System architecture specification. You can see four components there:

- Main Bank
- ATM
- Client
- Clerk

Client and Clerk communicate with Main Bank and each of them sends only requests. In the same time ATM communicates with Main Bank and sends requests and notifications.

Each component has a description which contains name, description, services provided and services requested fields. And each of these fields provides corresponding information.

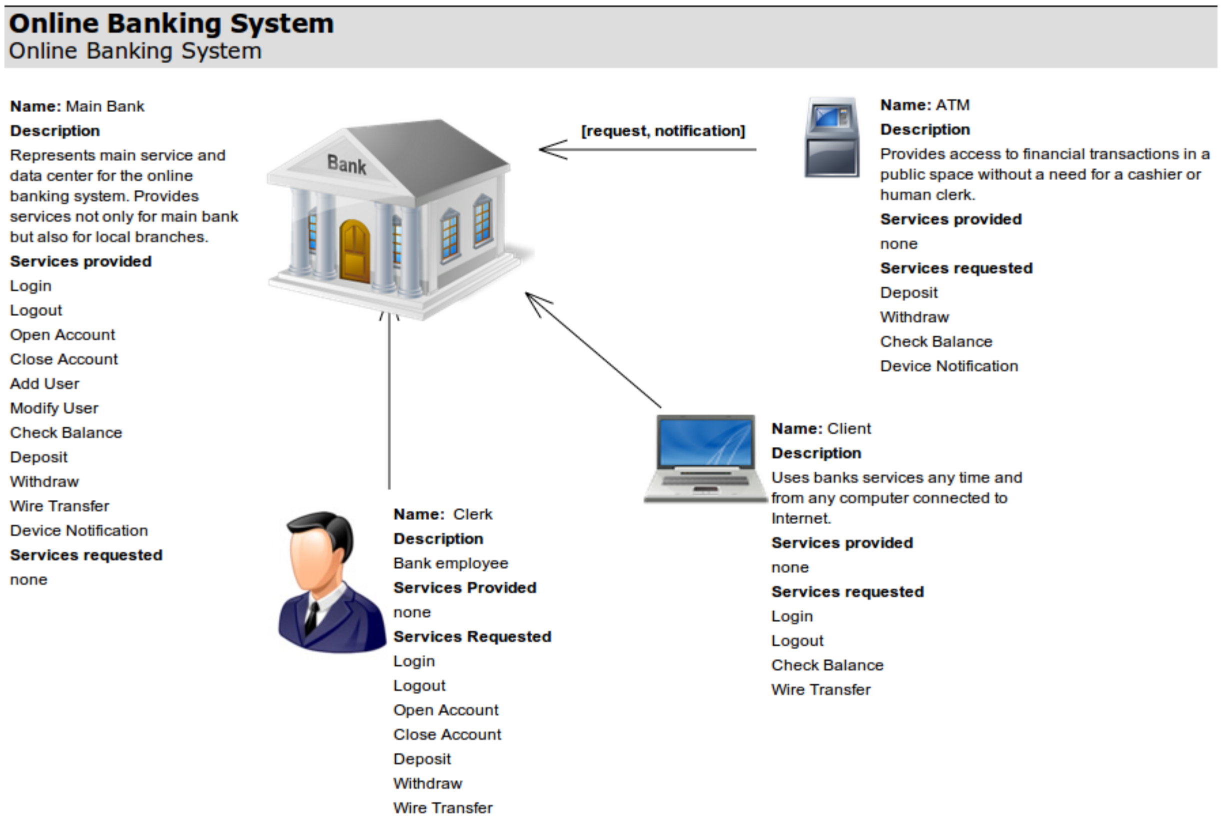
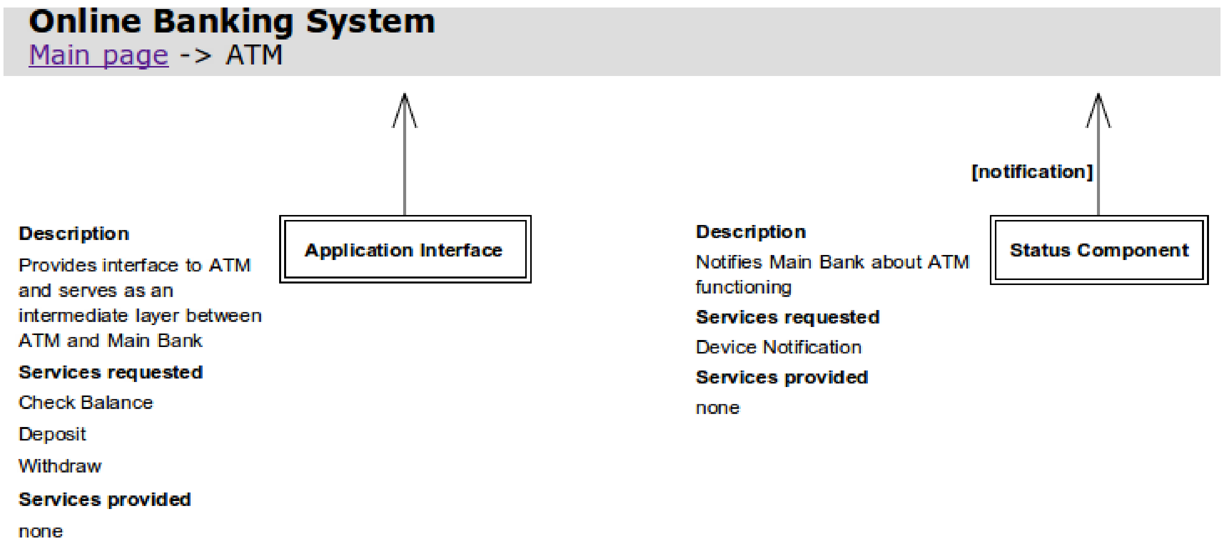


Figure 19. Top view of the Online Banking System architecture specification

Figure 20 shows specification of the ATM component. We can see that “Application Interface” component sends only requests and “Status Component” component sends only notifications. We can also see that both components do not provide any services and request services of the Main Bank component.



Online Banking System
Main page -> ATM

[notification]

Application Interface

Description
Provides interface to ATM and serves as an intermediate layer between ATM and Main Bank

Services requested
Check Balance
Deposit
Withdraw

Services provided
none

Status Component

Description
Notifies Main Bank about ATM functioning

Services requested
Device Notification

Services provided
none

Figure 20. Specification of the ATM component

Figure 21 provides specification of the Main Bank component. In some cases it is very inconvenient to show a lot of information in one place. And specification of the Main Bank component is exactly such case. It is decomposed further on four components and such decomposition is represented as a mind map. Using mind maps not only in standalone applications but also in a web environment can simplify the information perception.

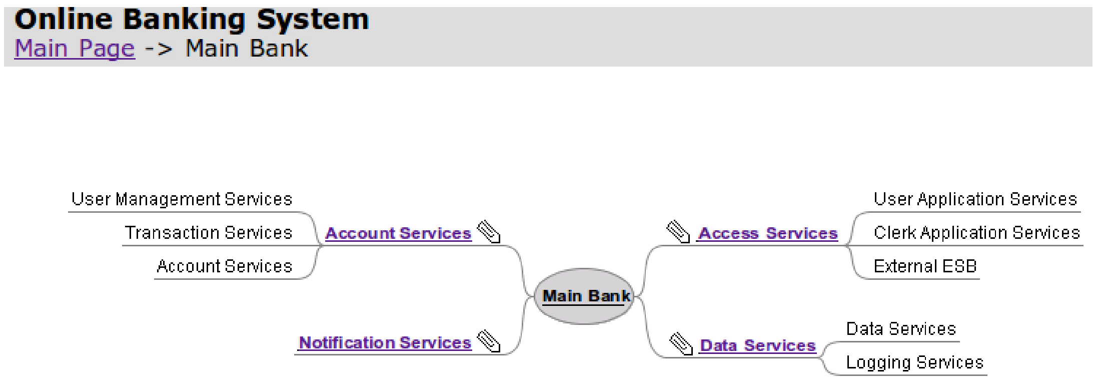
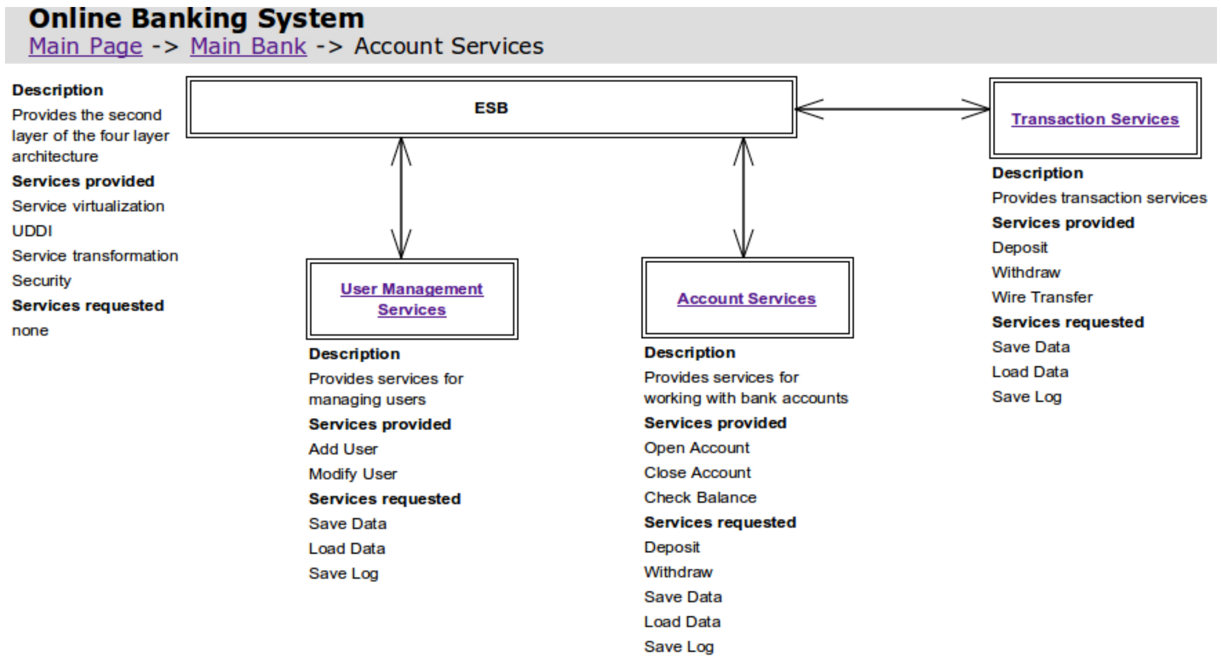


Figure 21. Specification of the Main Bank component

Main Bank component specification shows that Main Bank is decomposed on four components:

- Account Services
- Notification Services
- Access Services
- Data Services

Figure 22 provides specification of the Account Services component.


Online Banking System

Main Page -> Main Bank -> Account Services

ESB

Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security
Services requested
none

User Management Services

Description
Provides services for managing users
Services provided
Add User
Modify User
Services requested
Save Data
Load Data
Save Log

Account Services

Description
Provides services for working with bank accounts
Services provided
Open Account
Close Account
Check Balance
Services requested
Deposit
Withdraw
Save Data
Load Data
Save Log

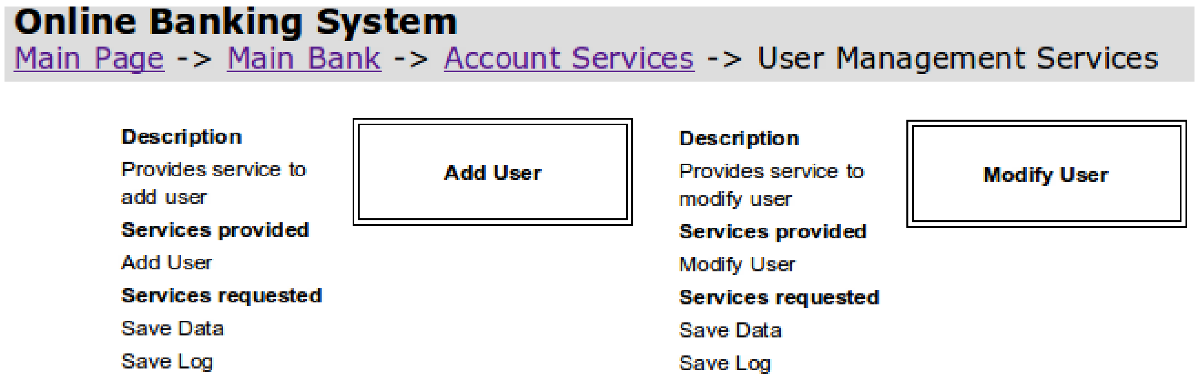
Transaction Services

Description
Provides transaction services
Services provided
Deposit
Withdraw
Wire Transfer
Services requested
Save Data
Load Data
Save Log

Figure 22. Specification of the Account Services component

We can see that some components of the Account Services component provides more than one service, therefore these components are decomposed further.

Next three figures provide specification of the User Management Services, Account Services and Transaction Services components.


Online Banking System

Main Page -> Main Bank -> Account Services -> User Management Services

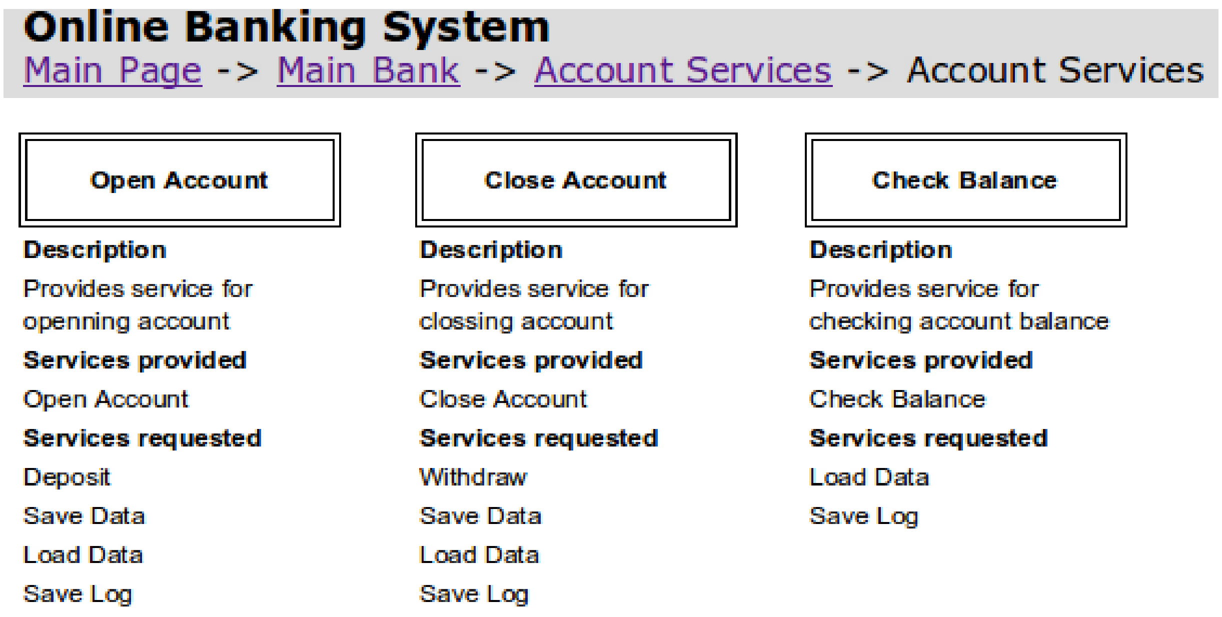
Add User

Description
Provides service to add user
Services provided
Add User
Services requested
Save Data
Save Log

Modify User

Description
Provides service to modify user
Services provided
Modify User
Services requested
Save Data
Save Log

Figure 23. Specification of the User Management Services component


Online Banking System
Main Page -> Main Bank -> Account Services -> Account Services

Open Account

Description
Provides service for openning account
Services provided
Open Account
Services requested
Deposit
Save Data
Load Data
Save Log

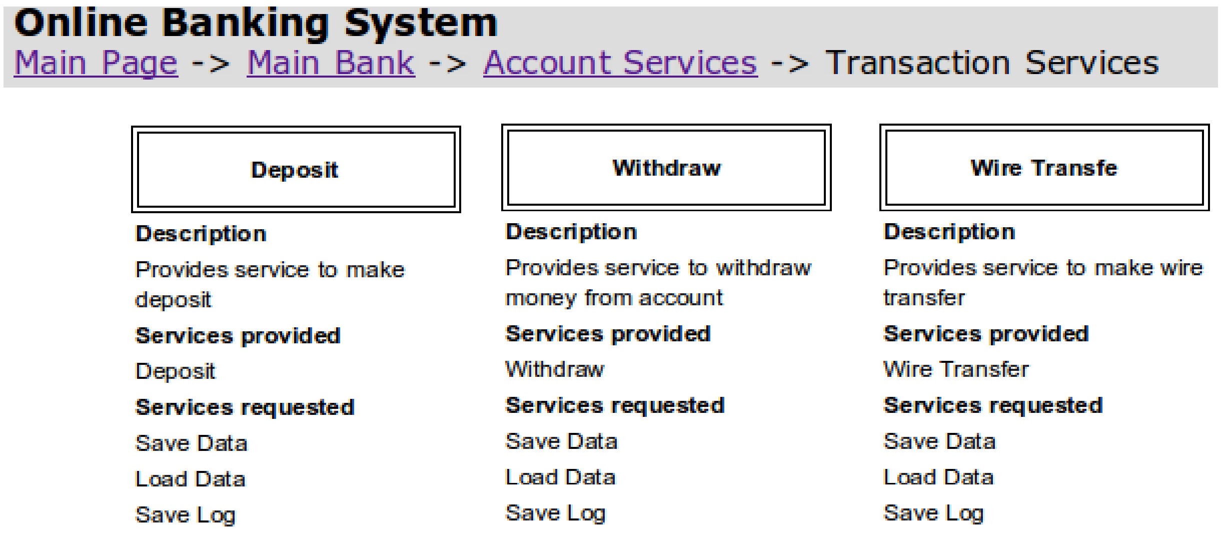
Close Account

Description
Provides service for closing account
Services provided
Close Account
Services requested
Withdraw
Save Data
Load Data
Save Log

Check Balance

Description
Provides service for checking account balance
Services provided
Check Balance
Services requested
Load Data
Save Log

Figure 24. Specification of the Account Services component


Online Banking System
Main Page -> Main Bank -> Account Services -> Transaction Services

Deposit

Description
Provides service to make deposit
Services provided
Deposit
Services requested
Save Data
Load Data
Save Log

Withdraw

Description
Provides service to withdraw money from account
Services provided
Withdraw
Services requested
Save Data
Load Data
Save Log

Wire Transfe

Description
Provides service to make wire transfer
Services provided
Wire Transfer
Services requested
Save Data
Load Data
Save Log

Figure 25. Specification of the Transaction Services component

Figure 26 provides specification of the Notification Services component which contains only one subcomponent.

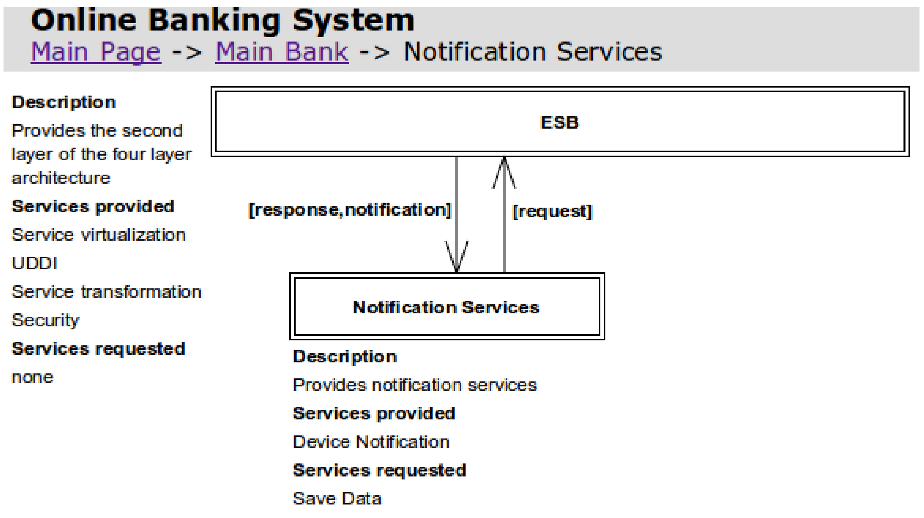


Figure 26. Specification of the Notification Services component

Figure 27 provides specification of the Access Services component.

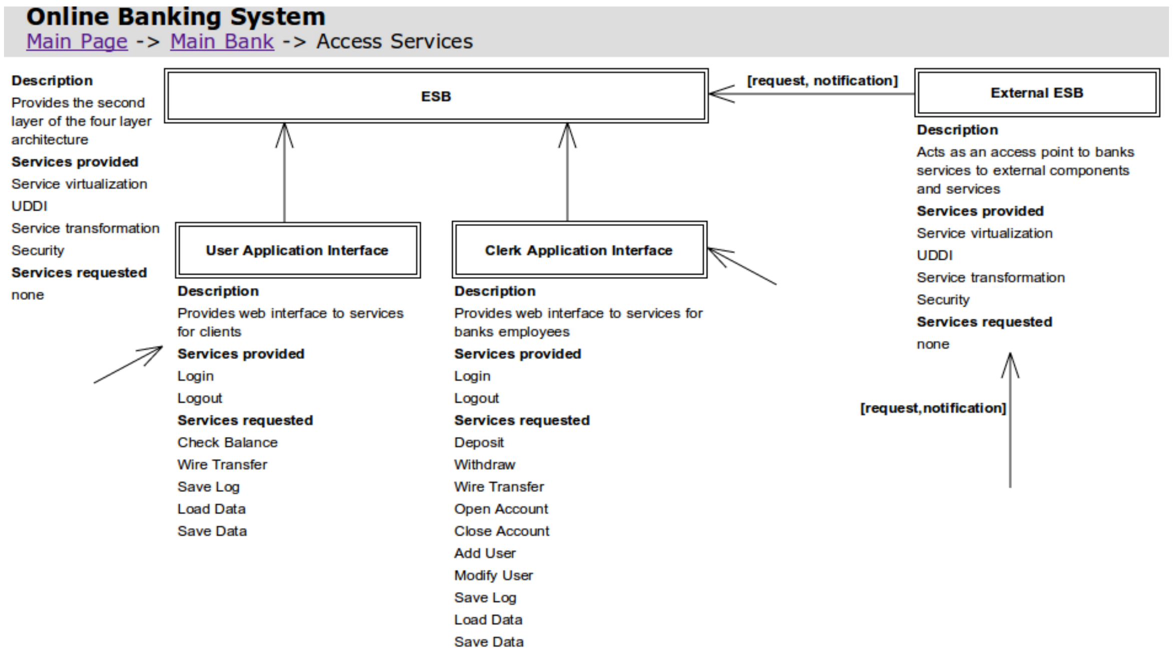
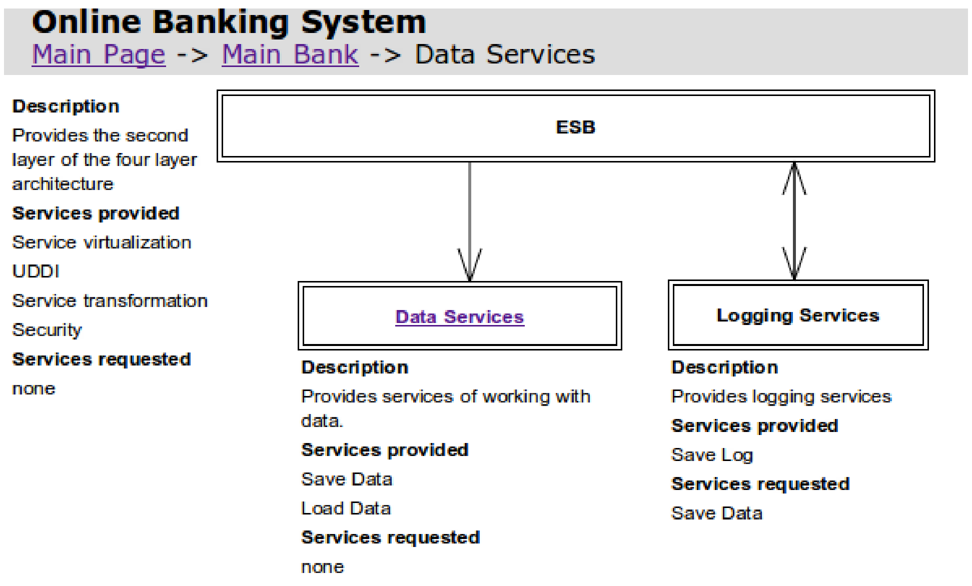


Figure 27. Specification of the Access Services component

We can see that User Application Interface, Clerk Application Interface and External ESB components are accessed outside. You can see it by arrows which come to them. Moreover, External ESB gets not only requests, but also it gets notifications. These three components are the access points for users of the Online Banking System (User Application Interface and Clerk Application Interface components) and external devices (External ESB component).

Figure 28 provides specification of the Data Services component. One of the components is also decomposed and its specification is presented in Figure 29.


Online Banking System

Main Page -> Main Bank -> Data Services

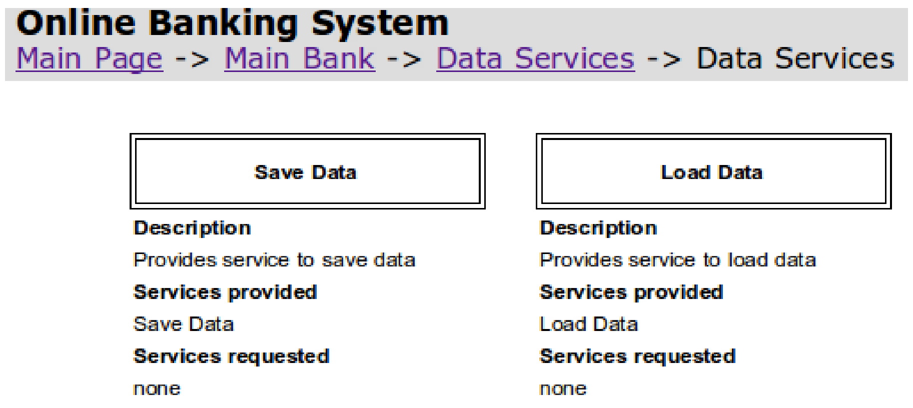
Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security
Services requested
none

ESB

Data Services
Description
Provides services of working with data.
Services provided
Save Data
Load Data
Services requested
none

Logging Services
Description
Provides logging services
Services provided
Save Log
Services requested
Save Data

Figure 28. Specification of the Data Services component


Online Banking System

Main Page -> Main Bank -> Data Services -> Data Services

Save Data
Description
Provides service to save data
Services provided
Save Data
Services requested
none

Load Data
Description
Provides service to load data
Services provided
Load Data
Services requested
none

Figure 29. Specification of the Data Services component of Data Services component group

4.2.1.3 Simulation Model

After architecture specification has been described, we can present the simulation model. As it was said before, simulation model was made manually and was prepared in VSE program. You can find many similarities between simulation model and architecture specification.

Figure 30 provides top view of the simulation model of Online Banking System. You can see four main components there:

- Main Bank
- ATM
- Clerk
- User

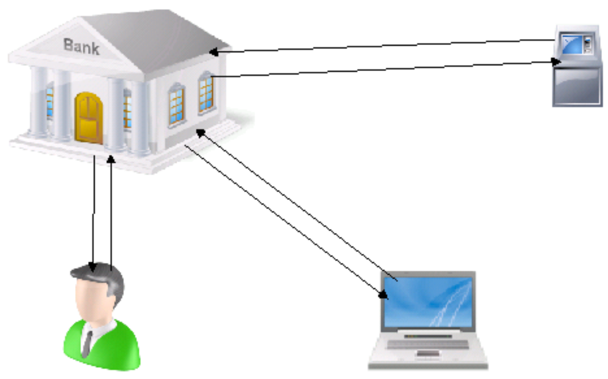


Figure 30. Top view of the simulation model of Online Banking System in VSE

Figure 31 shows ATM component decomposition.

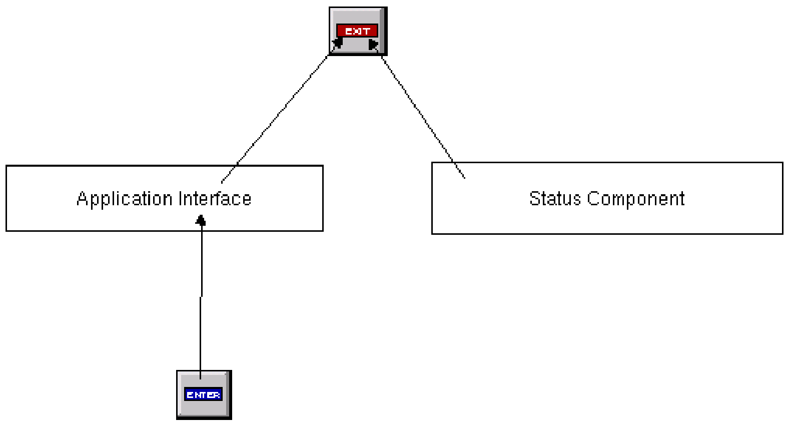


Figure 31. ATM component view in VSE

Figure 32 presents Main Bank in VSE.

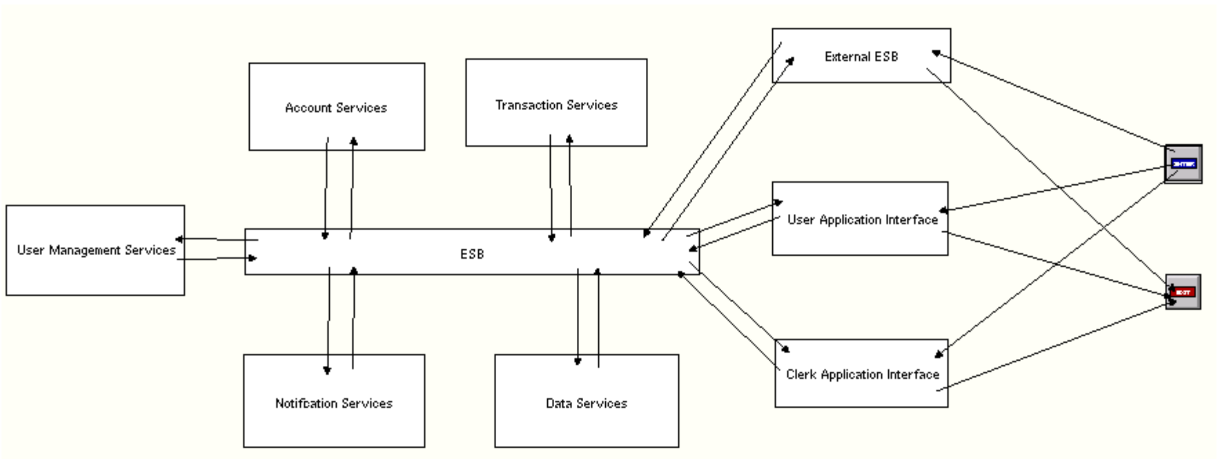


Figure 32. Main Bank view in VSE

In Figure 32 you can see all sub components of the Main Bank component and communications between them. Figure 33 and Figure 34 present correspondingly views of decomposed User Application Interface and Clerk Application Interface components in VSE.

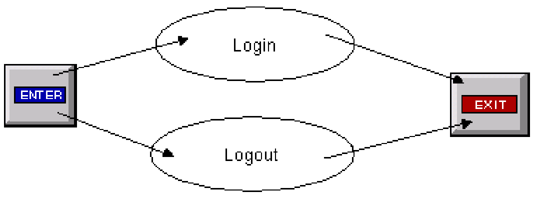


Figure 33. User Application Interface component view in VSE

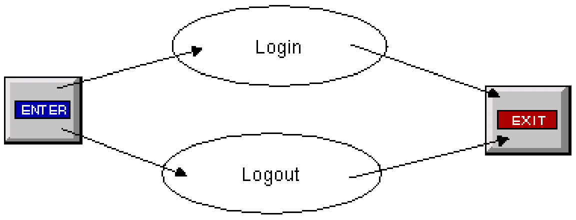


Figure 34. Clerk Application Interface component view in VSE

Account Services component view is presented in Figure 35. You can see there all services of Account Services component which were specified in the system architecture specification described earlier.

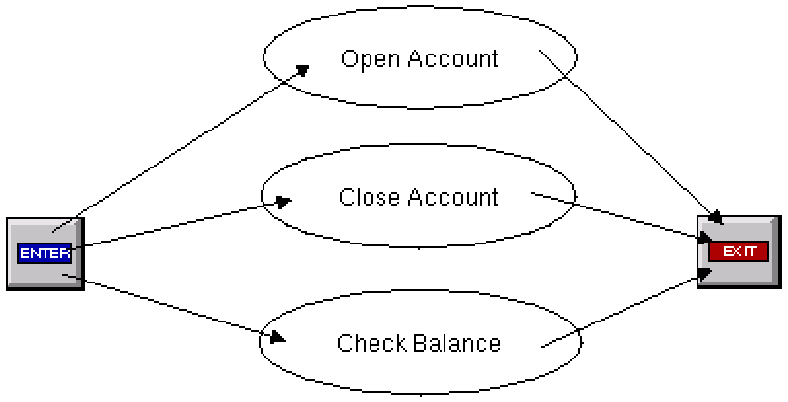


Figure 35. Account Services component view in VSE

Figure 36 shows Transaction Services component view in VSE.

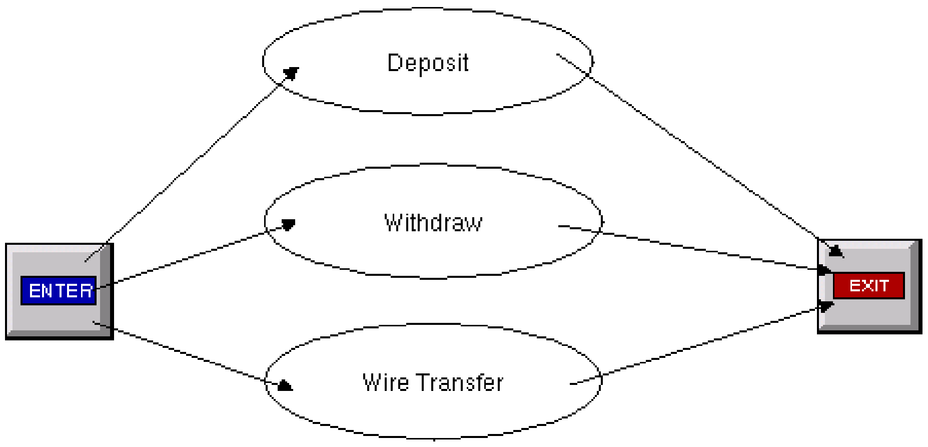


Figure 36. Transaction Services component view in VSE

The following three figures present Data Services view in VSE.

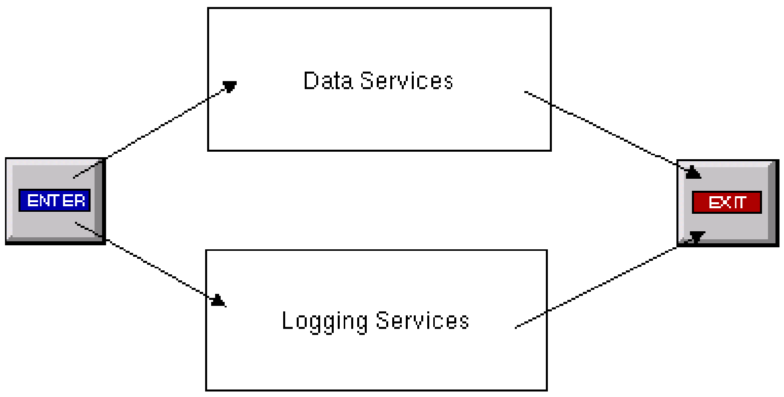


Figure 37. Data Services component view in VSE

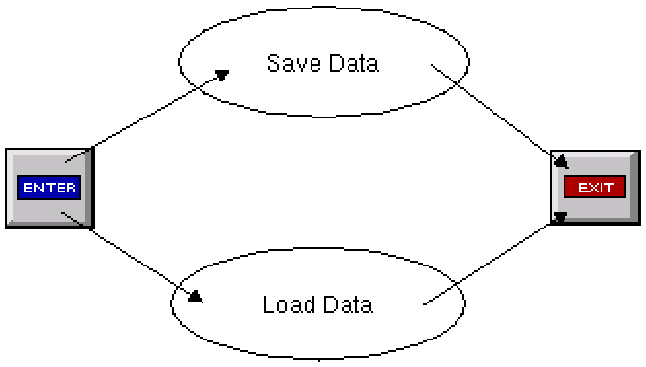


Figure 38. Data Services sub component view in VSE

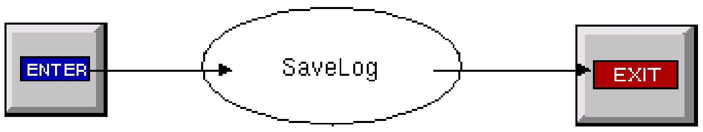


Figure 39. Logging Services component view in VSE

The last two figures show the Notification Services and User Management Services components views.

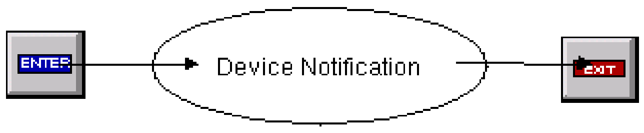


Figure 40. Notification Services component view in VSE

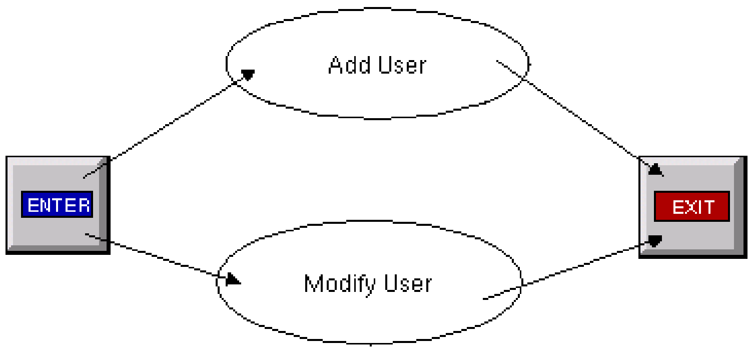


Figure 41. User Management Services component view in VSE

4.2.1.4 Simulation Model in Action

Here we want to present some figures which show how simulation model works. Figure 42 presents top view of the Online Banking System while running simulation model in VSE. You can see three types of messages there:

- Requests
- Responses
- Notifications

Requests are highlighted in blue, responses - in purple and notifications – in green.

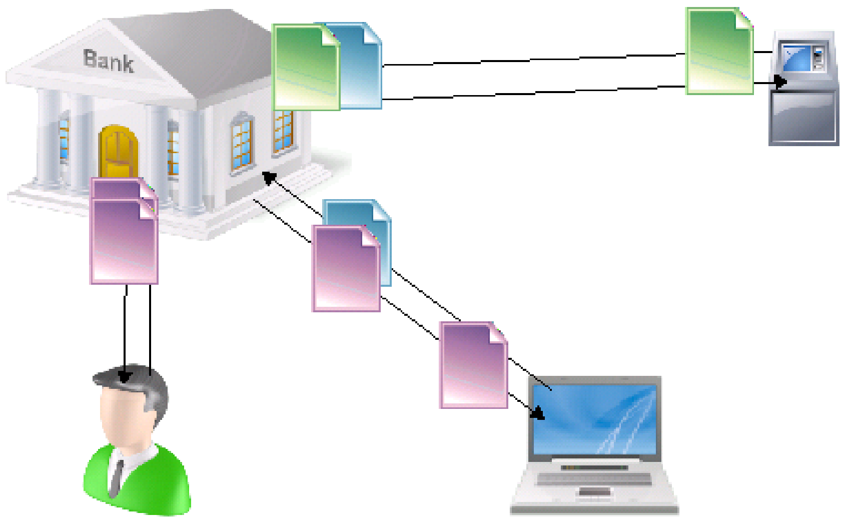


Figure 42. Top view of Online Banking System during running simulation model in VSE

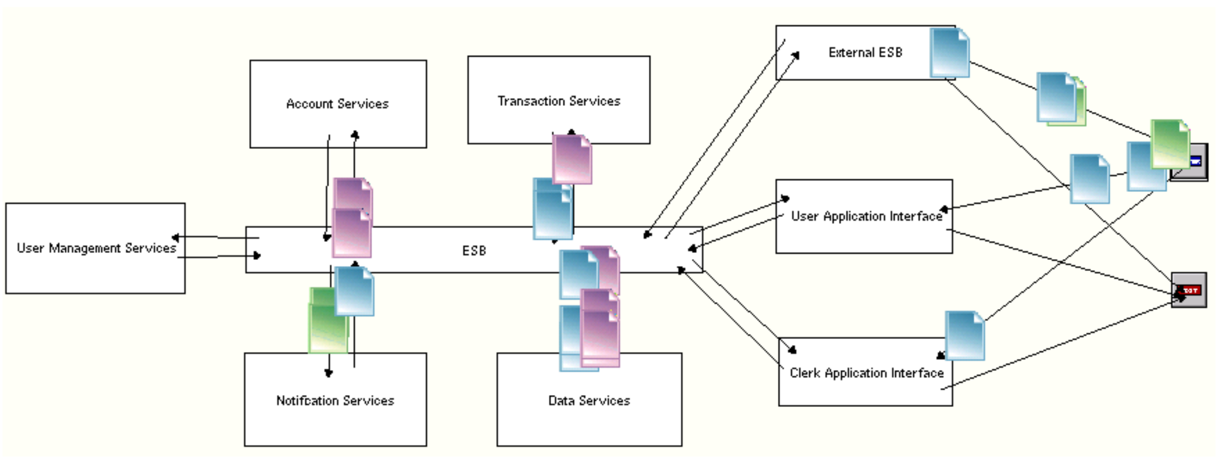


Figure 43. Main Bank view of Online Banking System during running simulation model in VSE

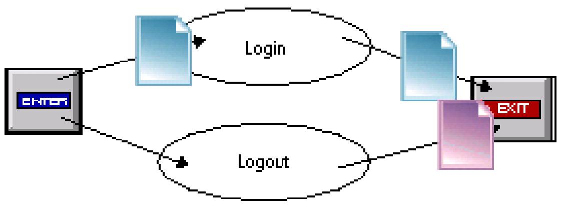


Figure 44. User Application Interface view of Online Banking System during running simulation model in VSE

4.2.2 *Online Learning Environment*

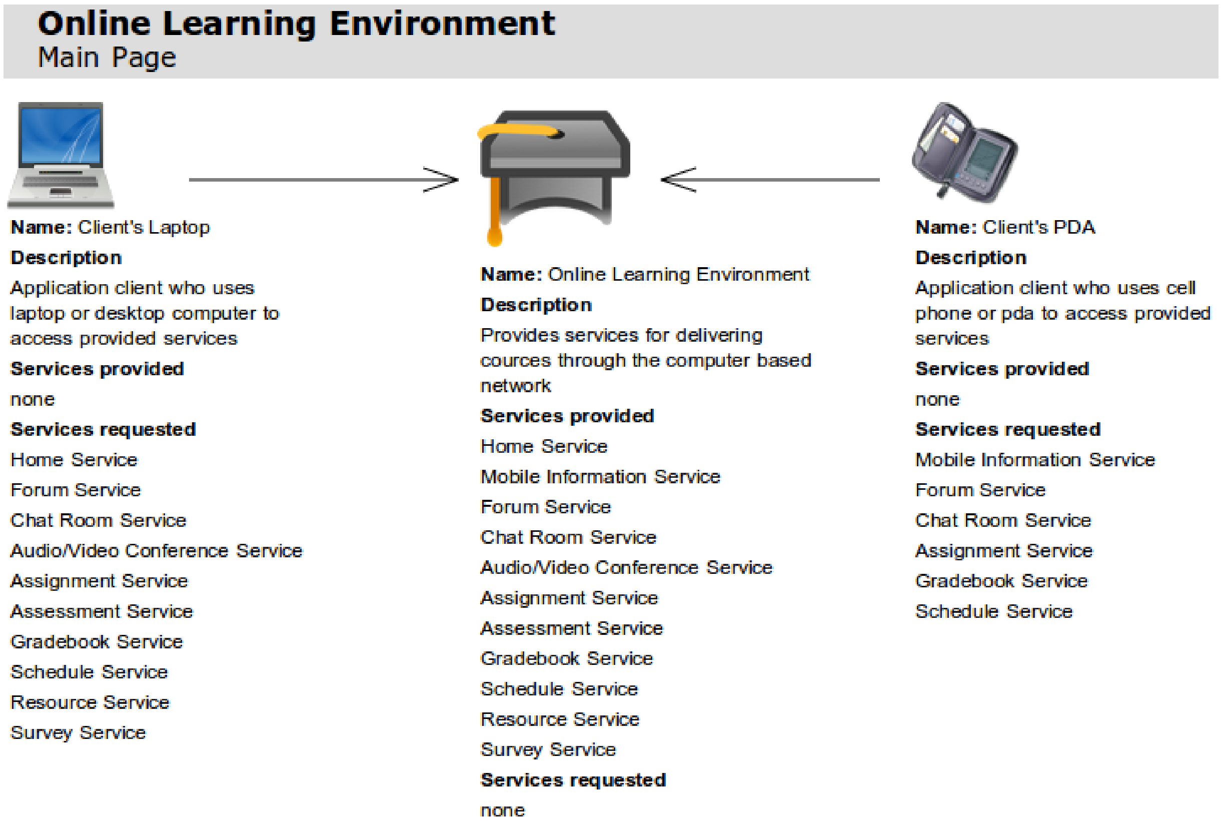
4.2.2.1 Overview

“An Online Learning Environment (OLE) is an integrated set of software tools that are accessible through a computer-based network and is used for delivering courses entirely on the world wide web (online), or enhancing traditional face-to-face (i.e. classroom based) courses” [Alpergin 2007].

The OLE system architecture specification, which is presented here, is based on the description of [Alpergin 2007].

4.2.2.2 System Architecture Specification

Figures of this section present OLE system architecture specification. Figure 45 shows the top view of the OLE architecture specification.



Online Learning Environment
Main Page

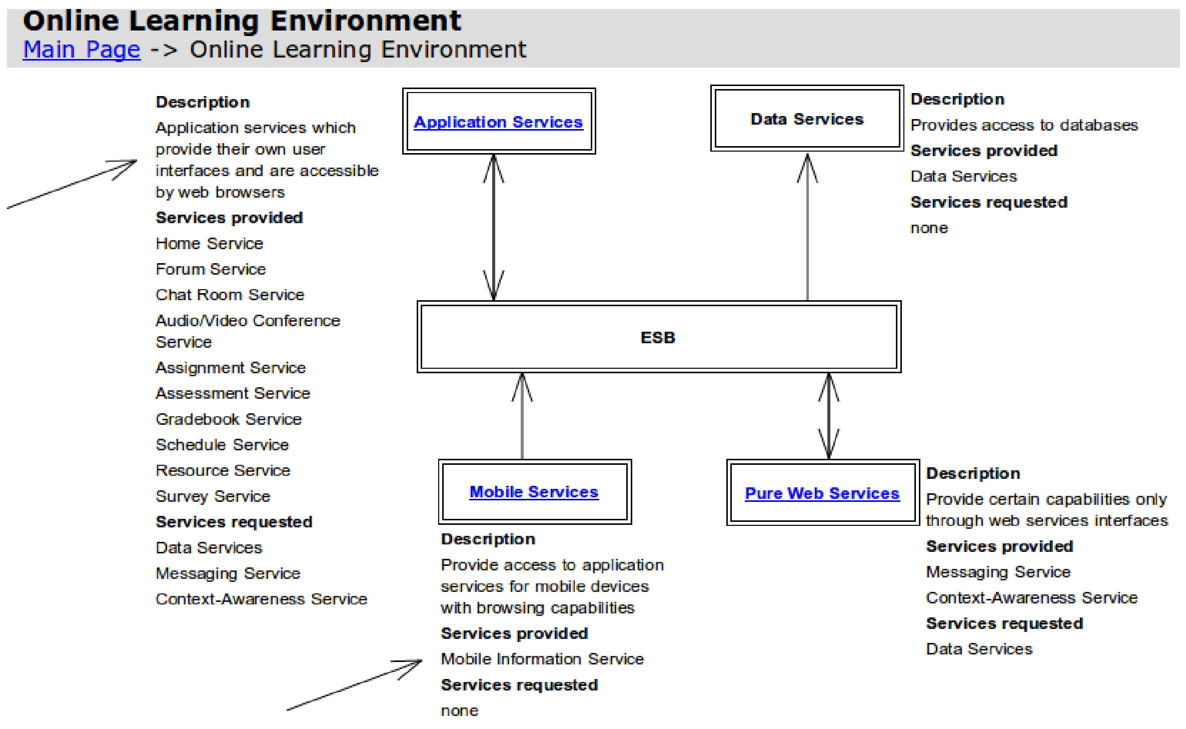
Name: Client's Laptop
Description
Application client who uses laptop or desktop computer to access provided services
Services provided
none
Services requested
Home Service
Forum Service
Chat Room Service
Audio/Video Conference Service
Assignment Service
Assessment Service
Gradebook Service
Schedule Service
Resource Service
Survey Service

Name: Online Learning Environment
Description
Provides services for delivering cources through the computer based network
Services provided
Home Service
Mobile Information Service
Forum Service
Chat Room Service
Audio/Video Conference Service
Assignment Service
Assessment Service
Gradebook Service
Schedule Service
Resource Service
Survey Service
Services requested
none

Name: Client's PDA
Description
Application client who uses cell phone or pda to access provided services
Services provided
none
Services requested
Mobile Information Service
Forum Service
Chat Room Service
Assignment Service
Gradebook Service
Schedule Service

Figure 45. Top view of the OLE architecture specification

Client's laptop and client's PDA components do not provide any services. Therefore they are not decomposed. Figure 46 provides the specification of the Online Learning Environment component.


Online Learning Environment
Main Page -> Online Learning Environment

Description
Application services which provide their own user interfaces and are accessible by web browsers
Services provided
Home Service
Forum Service
Chat Room Service
Audio/Video Conference Service
Assignment Service
Assessment Service
Gradebook Service
Schedule Service
Resource Service
Survey Service
Services requested
Data Services
Messaging Service
Context-Awareness Service

Description
Provides access to databases
Services provided
Data Services
Services requested
none

Description
Provide access to application services for mobile devices with browsing capabilities
Services provided
Mobile Information Service
Services requested
none

Description
Provide certain capabilities only through web services interfaces
Services provided
Messaging Service
Context-Awareness Service
Services requested
Data Services

Figure 46. Specification of the OLE component

Application Services component provides many services. That is why Figure 47 shows the decomposition of the Application Services component which is presented as a mind map.

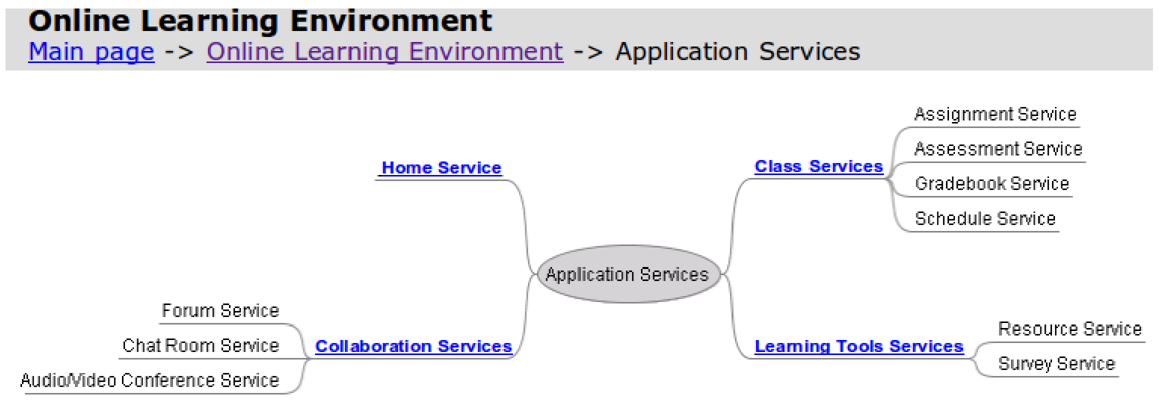
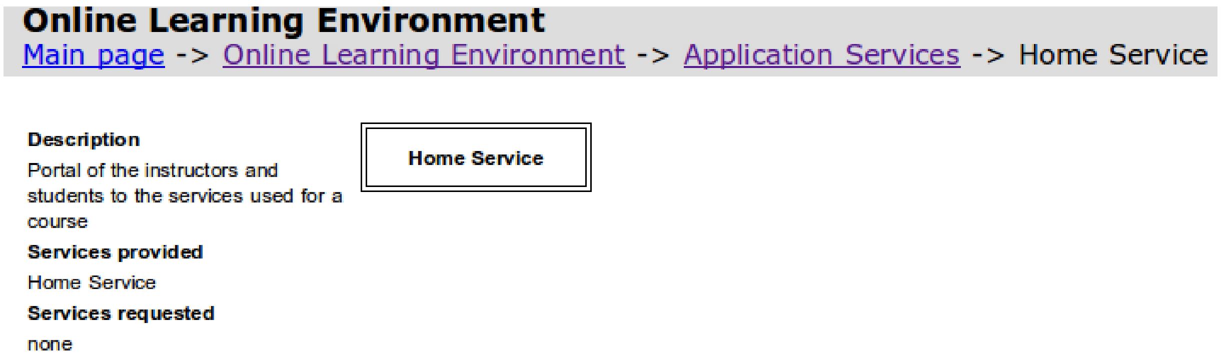

Online Learning Environment
Main page -> Online Learning Environment -> Application Services

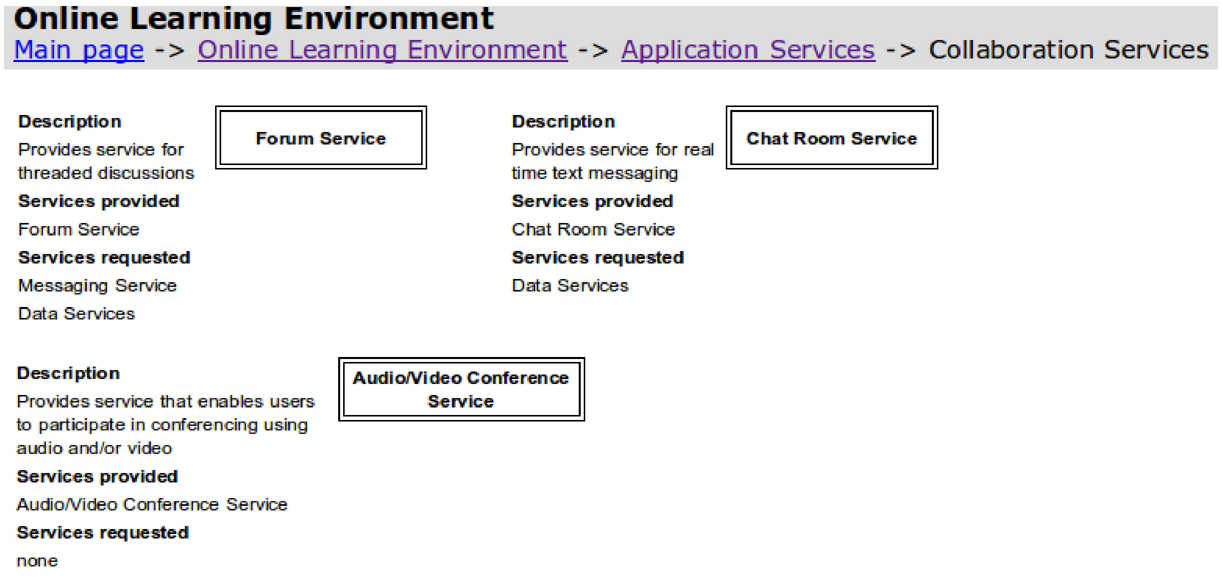
Figure 47. Specification of the Application Services component

The following figures provide specifications of all components of Application Services component.


Online Learning Environment
Main page -> Online Learning Environment -> Application Services -> Home Service

Description
Portal of the instructors and students to the services used for a course
Services provided
Home Service
Services requested
none

Figure 48. Specification of the Home Services component

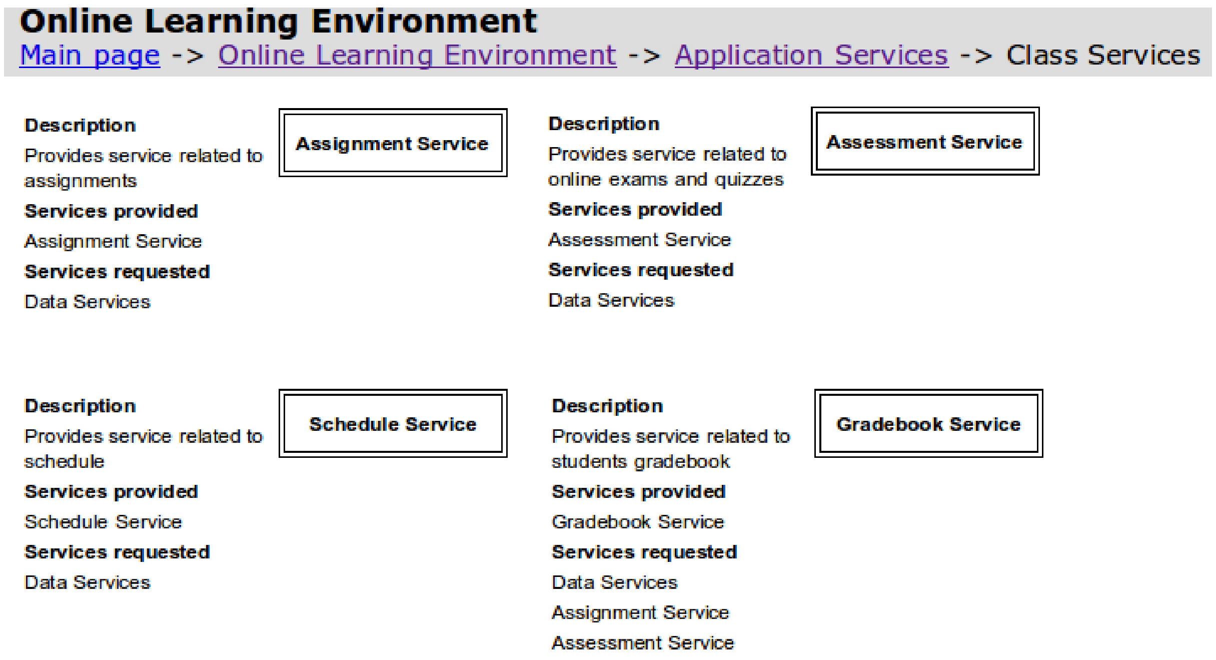

Online Learning Environment
Main page -> Online Learning Environment -> Application Services -> Collaboration Services

Description
Provides service for threaded discussions
Services provided
Forum Service
Services requested
Messaging Service
Data Services

Description
Provides service for real time text messaging
Services provided
Chat Room Service
Services requested
Data Services

Description
Provides service that enables users to participate in conferencing using audio and/or video
Services provided
Audio/Video Conference Service
Services requested
none

Figure 49. Specification of the Collaboration Services component


Online Learning Environment
Main page -> Online Learning Environment -> Application Services -> Class Services
Assignment Service
Assessment Service
Schedule Service
Gradebook Service

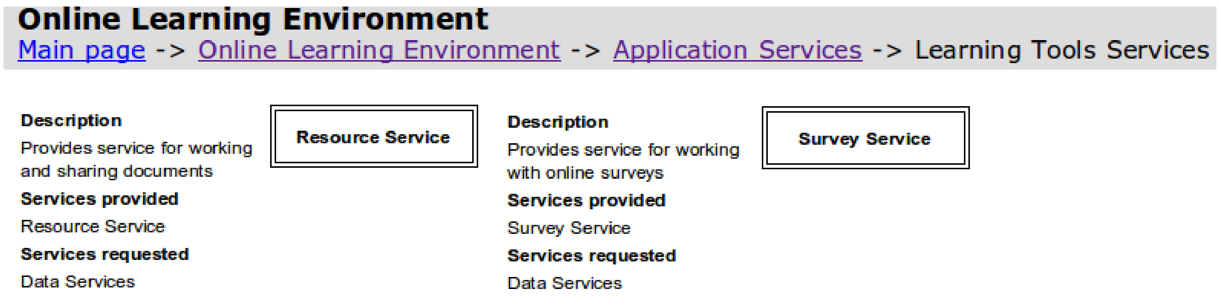
Description
Provides service related to assignments
Services provided
Assignment Service
Services requested
Data Services

Description
Provides service related to online exams and quizzes
Services provided
Assessment Service
Services requested
Data Services

Description
Provides service related to schedule
Services provided
Schedule Service
Services requested
Data Services

Description
Provides service related to students gradebook
Services provided
Gradebook Service
Services requested
Data Services
Assignment Service
Assessment Service

Figure 50. Specification of the Class Services component

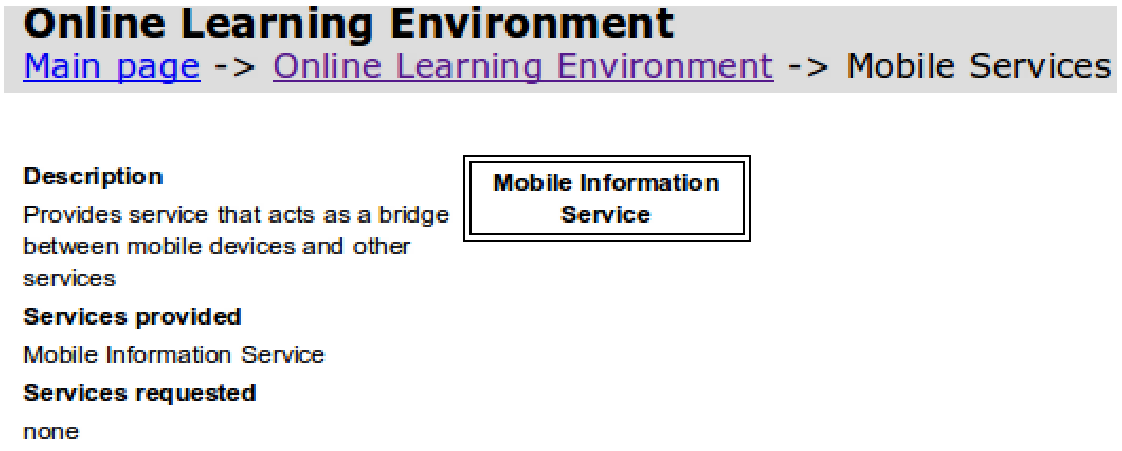

Online Learning Environment
Main page -> Online Learning Environment -> Application Services -> Learning Tools Services
Resource Service
Survey Service

Description
Provides service for working and sharing documents
Services provided
Resource Service
Services requested
Data Services

Description
Provides service for working with online surveys
Services provided
Survey Service
Services requested
Data Services

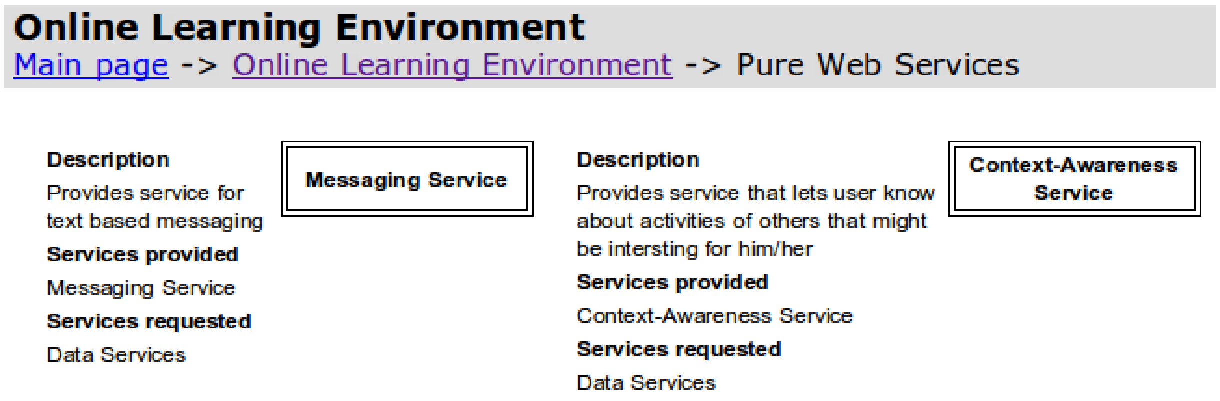
Figure 51. Specification of the Learning Tools Services component

The last two figures provide specifications of Mobile Services and Pure Web Services components.


Online Learning Environment
Main page -> Online Learning Environment -> Mobile Services

Description
Provides service that acts as a bridge between mobile devices and other services
Services provided
Mobile Information Service
Services requested
none

Figure 52. Specification of the Mobile Services component


Online Learning Environment
Main page -> Online Learning Environment -> Pure Web Services

Description
Provides service for text based messaging
Services provided
Messaging Service
Services requested
Data Services

Description
Provides service that lets user know about activities of others that might be intersting for him/her
Services provided
Context-Awareness Service
Services requested
Data Services

Figure 53. Specification of the Pure Web Services component

4.2.2.3 Simulation Model

Figure 54 shows the top view of the simulation model of OLE in VSE.

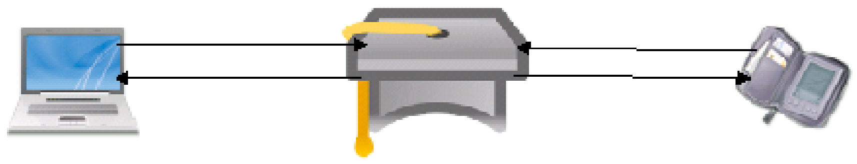


Figure 54. Top view of the simulation model of OLE in VSE

The next figure shows the view of the decomposed Online Learning Environment component.

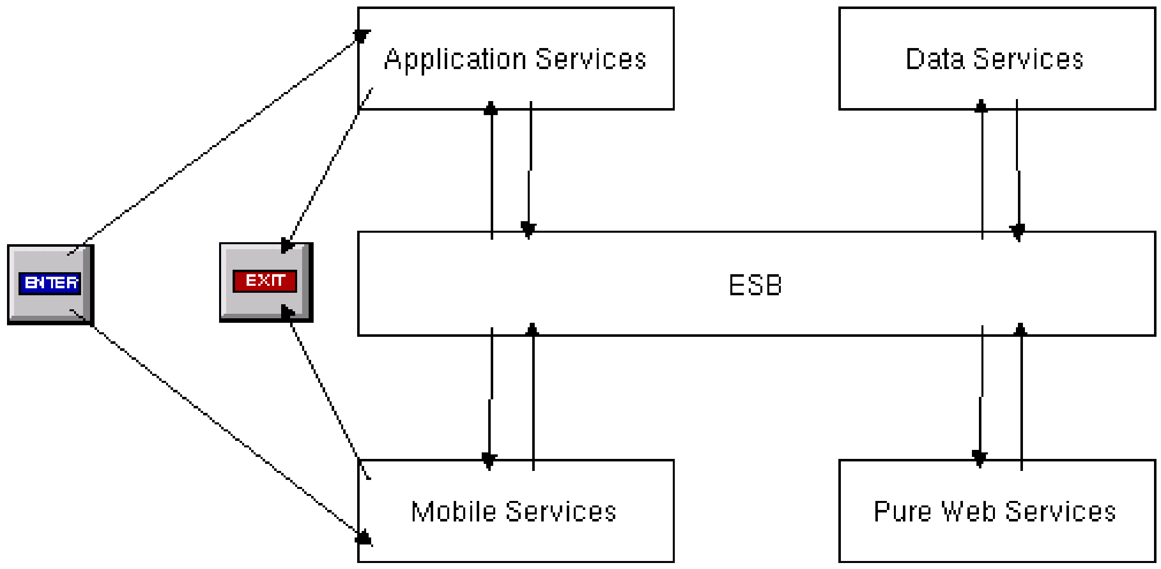


Figure 55. Online Learning Environment view in VSE

Application Services component view and all its subcomponents views are shown in the following four figures.

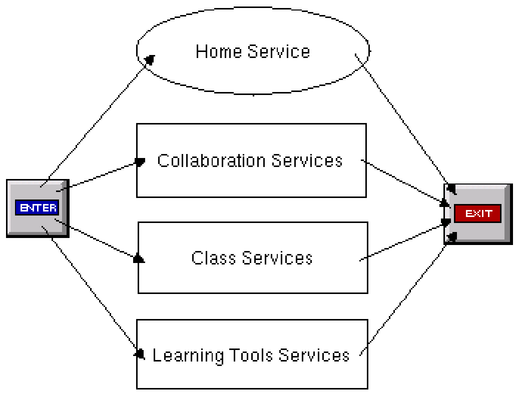


Figure 56. Application Services component view in VSE

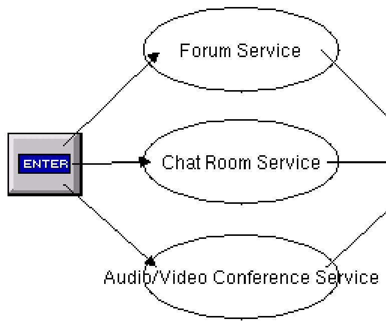


Figure 57. Collaboration Services component view in VSE

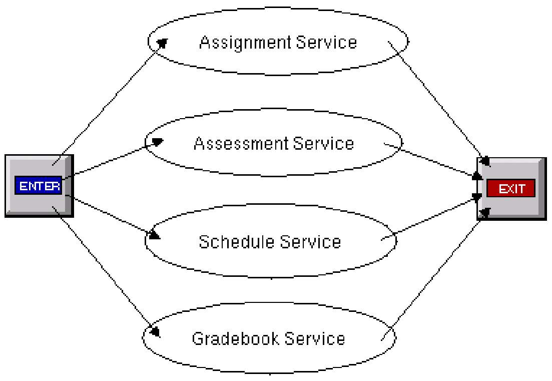


Figure 58. Class Services component view in VSE

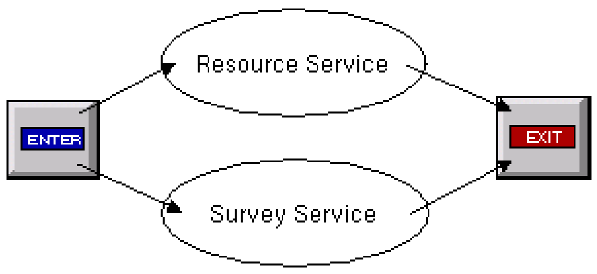


Figure 59. Learning Tools Services component view in VSE

The last two figures present views of Mobile Services and Pure Web Services components.

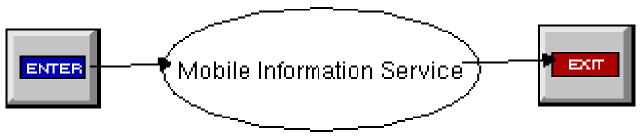


Figure 60. Mobile Services component view in VSE

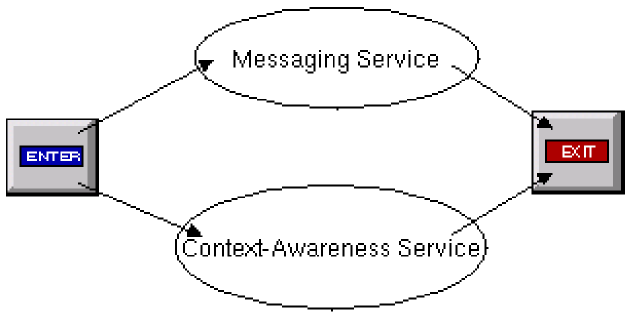


Figure 61. Pure Web Services component view in VSE

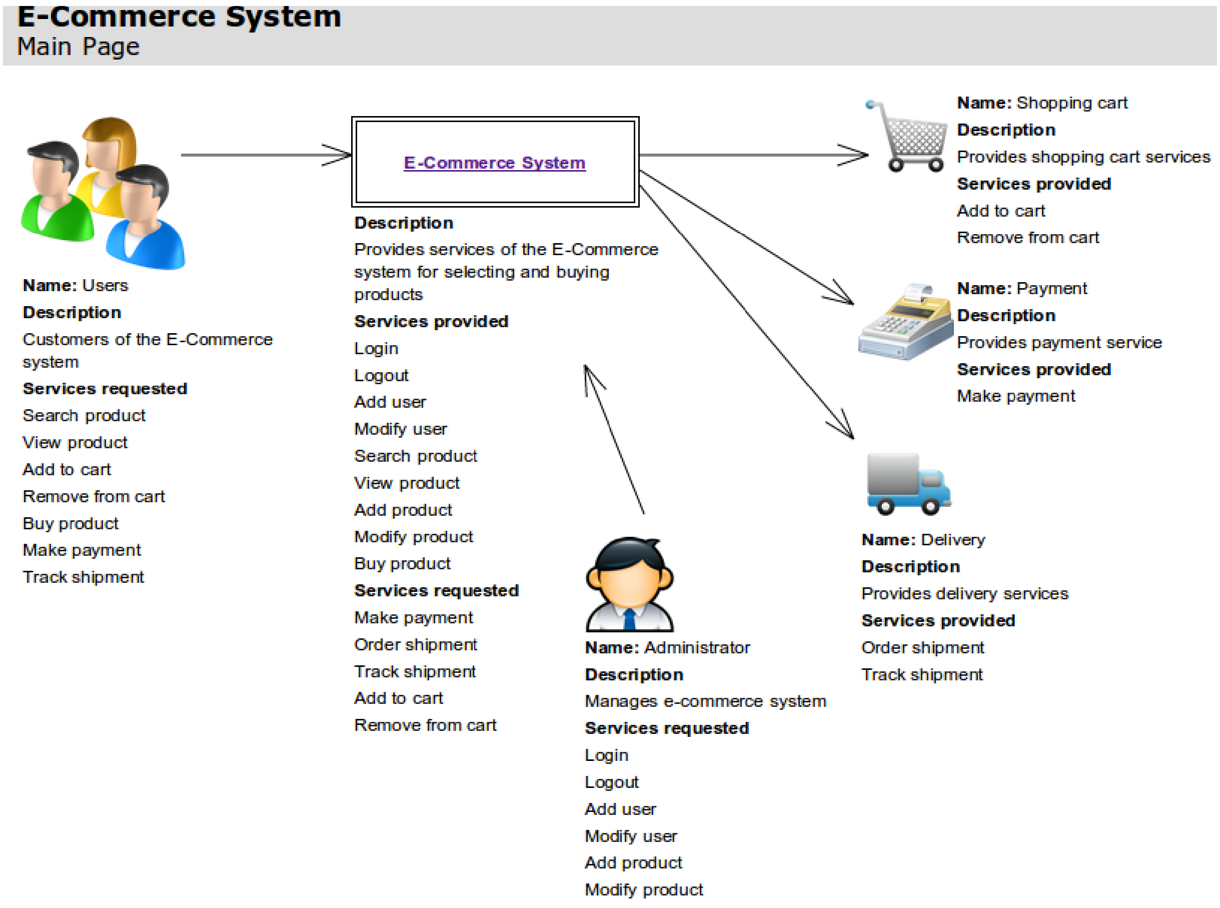
4.2.3 E-Commerce System

4.2.3.1 Overview

The example in this section describes E-Commerce System which represents a network-centric application and allows users to buy products online. The described system uses external providers for shopping cart, payment and delivery services.

The following two sections provide figures of the system architecture specification and the simulation model of the E-Commerce System.

4.2.3.2 System Architecture Specification



E-Commerce System
Main Page

E-Commerce System

Description
Provides services of the E-Commerce system for selecting and buying products
Services provided
Login
Logout
Add user
Modify user
Search product
View product
Add product
Modify product
Buy product
Services requested
Make payment
Order shipment
Track shipment
Add to cart
Remove from cart

Name: Users
Description
Customers of the E-Commerce system
Services requested
Search product
View product
Add to cart
Remove from cart
Buy product
Make payment
Track shipment

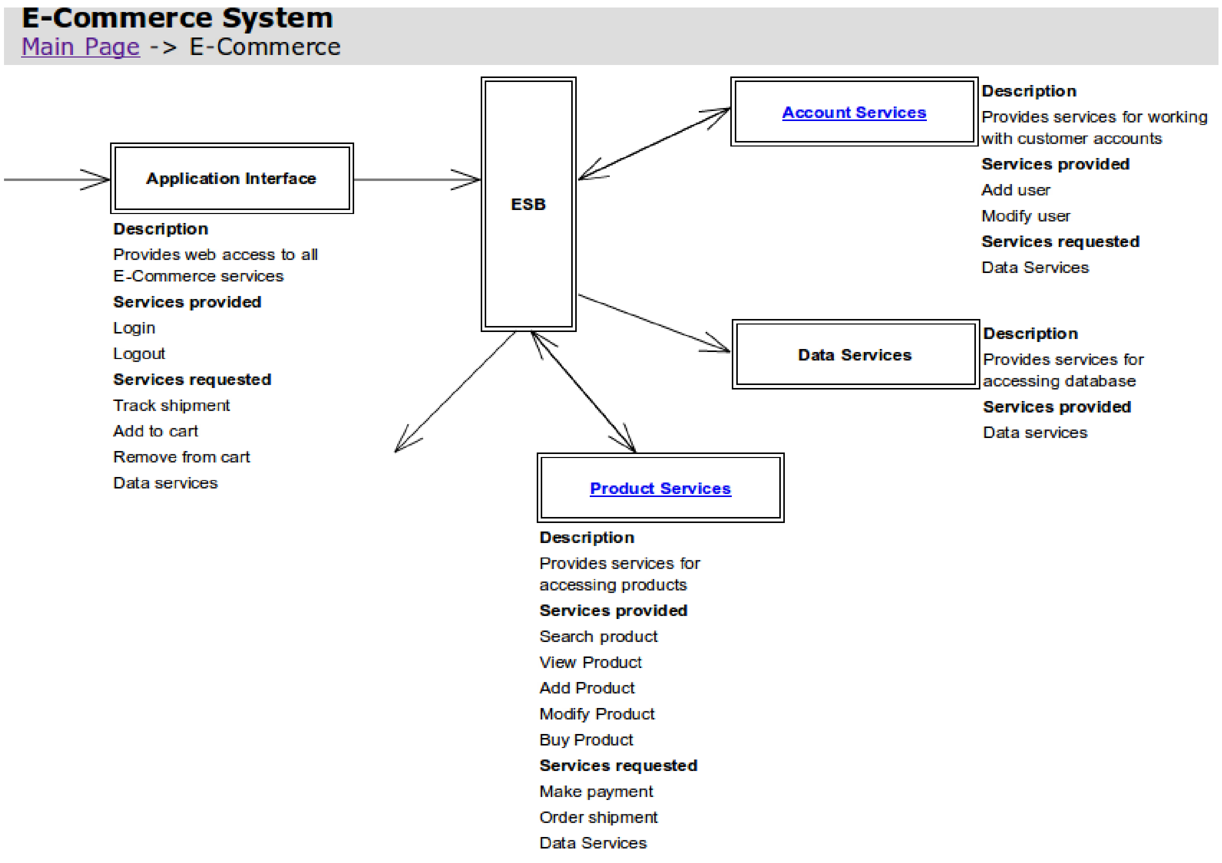
Name: Administrator
Description
Manages e-commerce system
Services requested
Login
Logout
Add user
Modify user
Add product
Modify product

Name: Shopping cart
Description
Provides shopping cart services
Services provided
Add to cart
Remove from cart

Name: Payment
Description
Provides payment service
Services provided
Make payment

Name: Delivery
Description
Provides delivery services
Services provided
Order shipment
Track shipment

Figure 62. Top view of the E-Commerce System architecture specification



Application Interface

Description
Provides web access to all E-Commerce services
Services provided
Login
Logout
Services requested
Track shipment
Add to cart
Remove from cart
Data services

Account Services

Description
Provides services for working with customer accounts
Services provided
Add user
Modify user
Services requested
Data Services

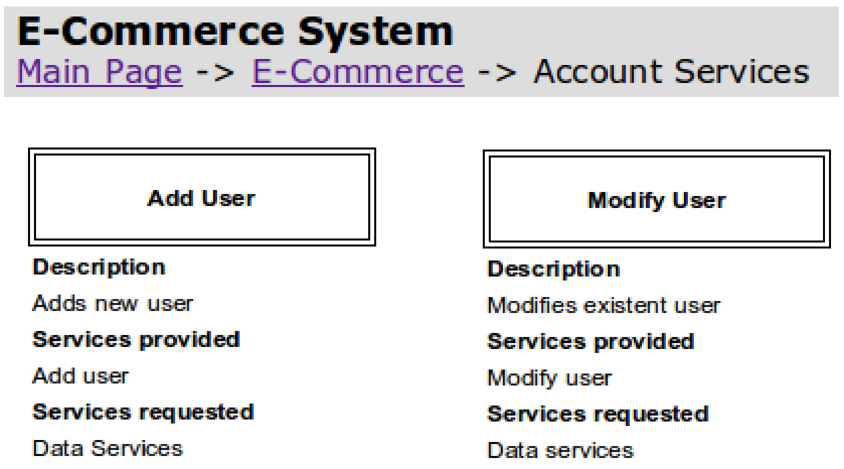
Data Services

Description
Provides services for accessing database
Services provided
Data services

Product Services

Description
Provides services for accessing products
Services provided
Search product
View Product
Add Product
Modify Product
Buy Product
Services requested
Make payment
Order shipment
Data Services

Figure 63. Specification of the E-Commerce System component



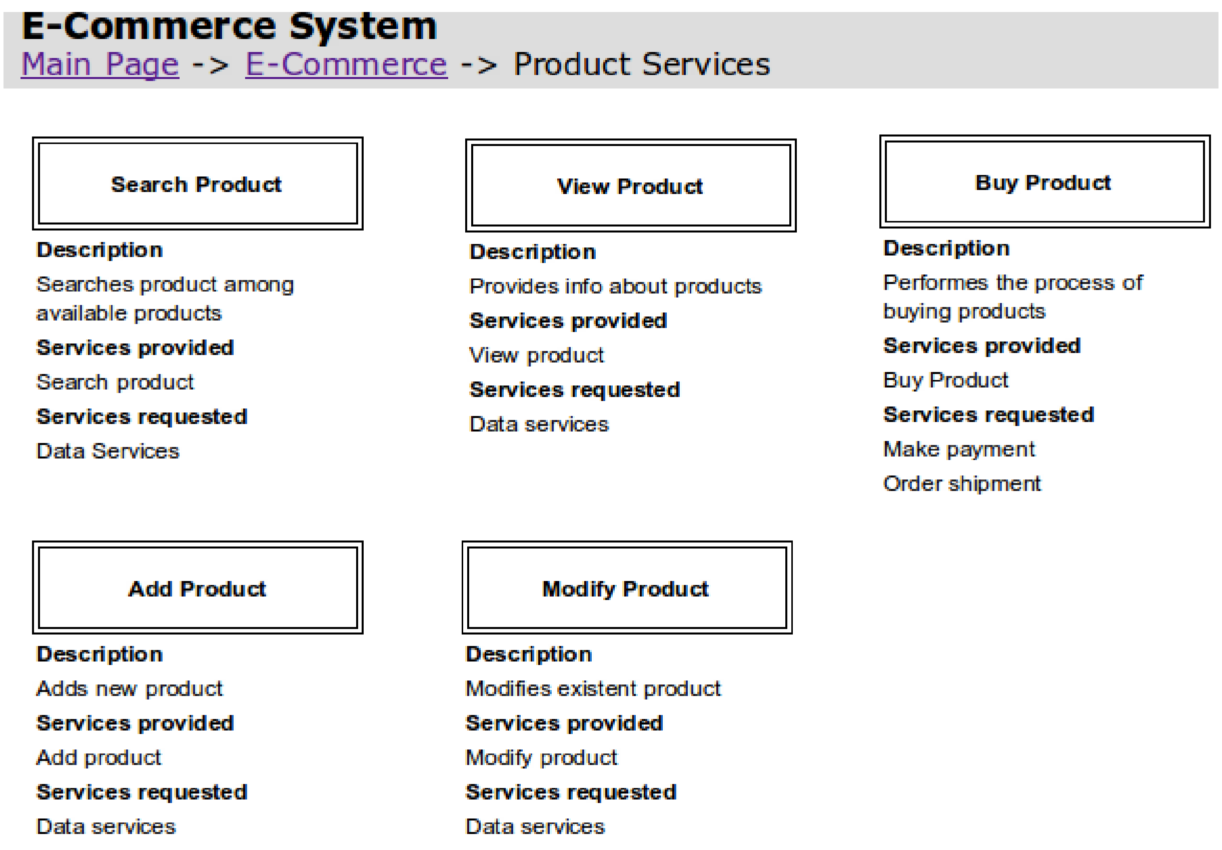
Add User

Description
Adds new user
Services provided
Add user
Services requested
Data Services

Modify User

Description
Modifies existent user
Services provided
Modify user
Services requested
Data services

Figure 64. Specification of the Account Services component



Search Product
Description
Searches product among available products
Services provided
Search product
Services requested
Data Services

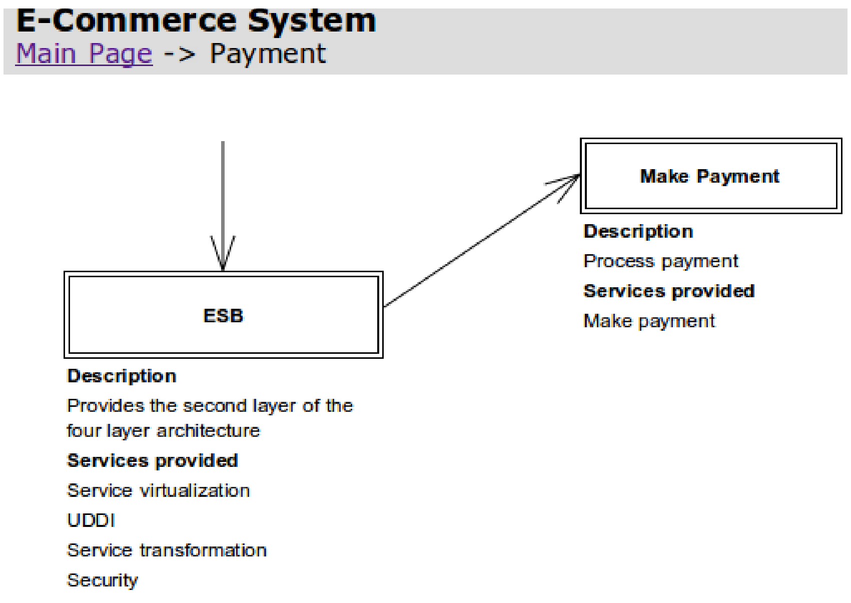
View Product
Description
Provides info about products
Services provided
View product
Services requested
Data services

Buy Product
Description
Performes the process of buying products
Services provided
Buy Product
Services requested
Make payment
Order shipment

Add Product
Description
Adds new product
Services provided
Add product
Services requested
Data services

Modify Product
Description
Modifies existent product
Services provided
Modify product
Services requested
Data services

Figure 65. Specification of the Product Services component



Make Payment
Description
Process payment
Services provided
Make payment

ESB
Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security

Figure 66. Specification of the Payment component

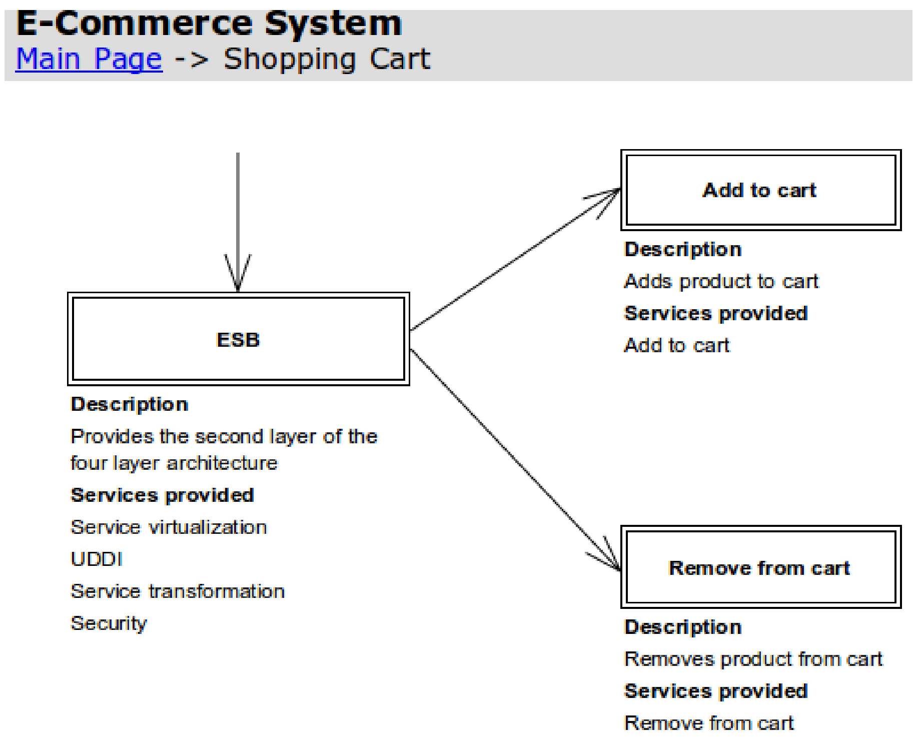


Figure 67. Specification of the Shopping Cart component

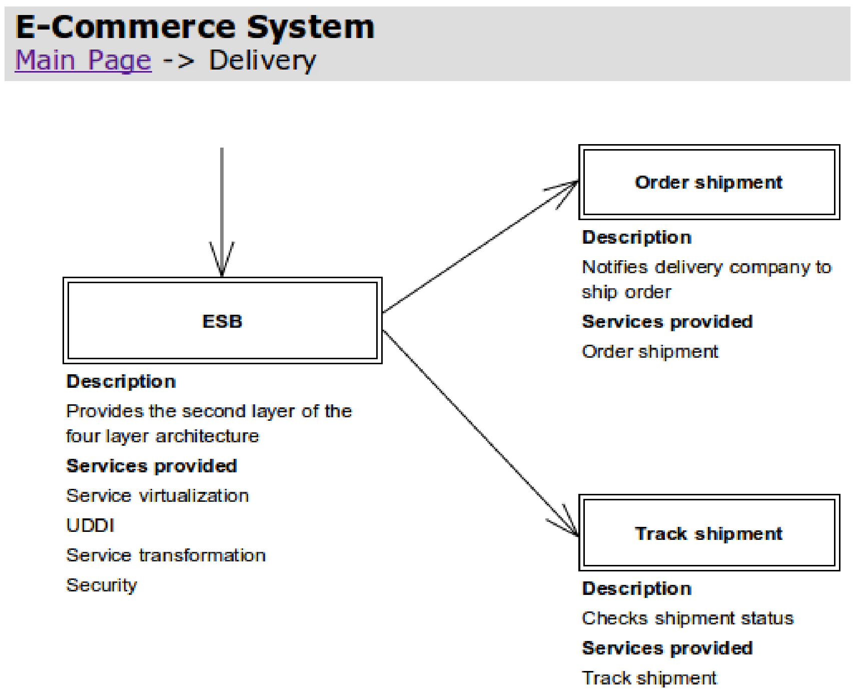


Figure 68. Specification of the Delivery component

4.2.3.3 Simulation Model

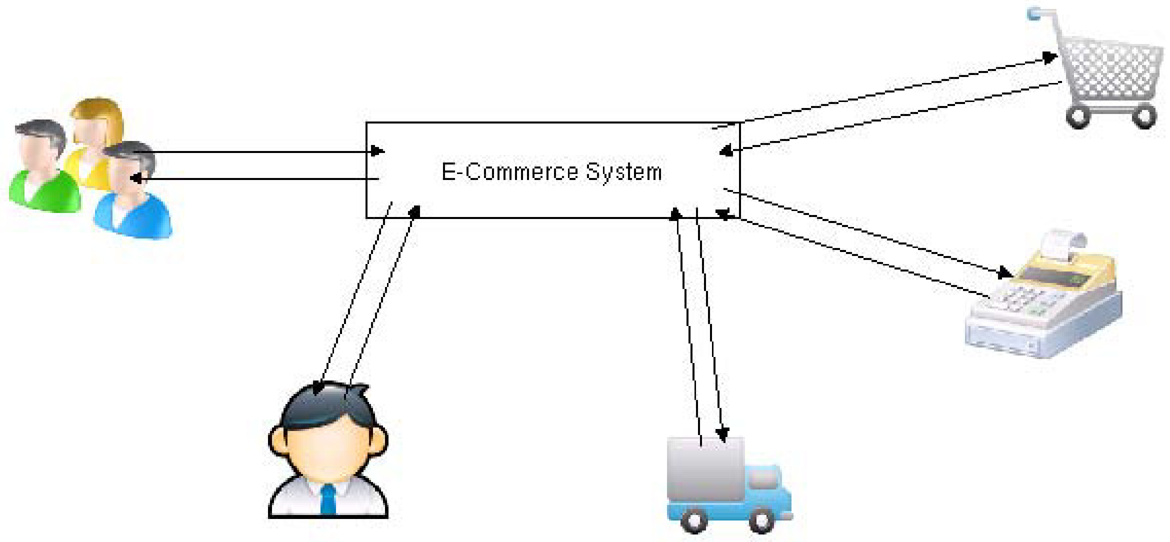


Figure 69. Top view of the simulation model of E-Commerce System in VSE

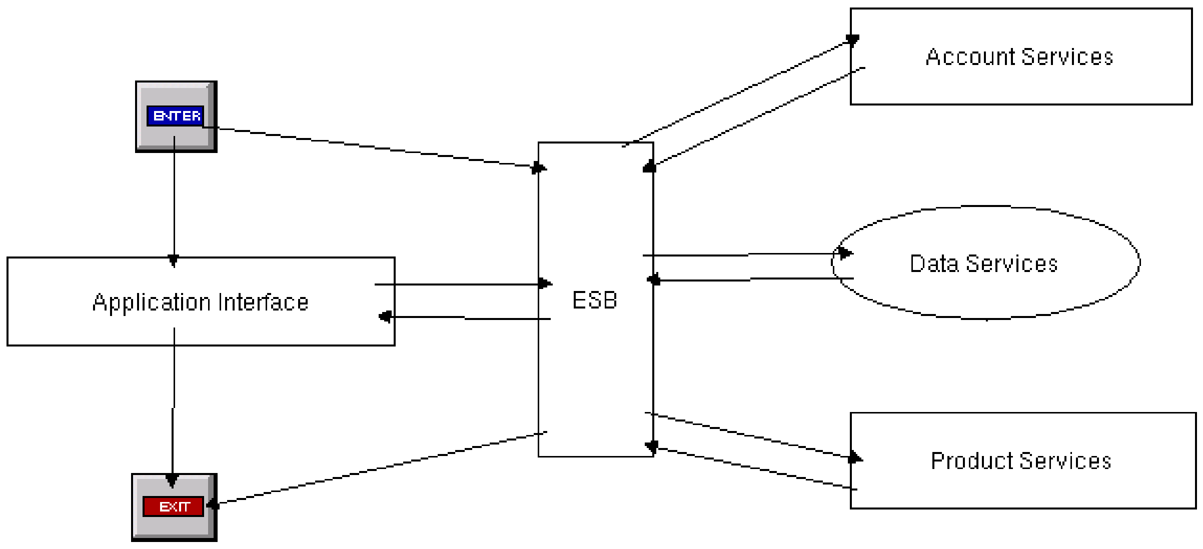


Figure 70. E-Commerce component view in VCE

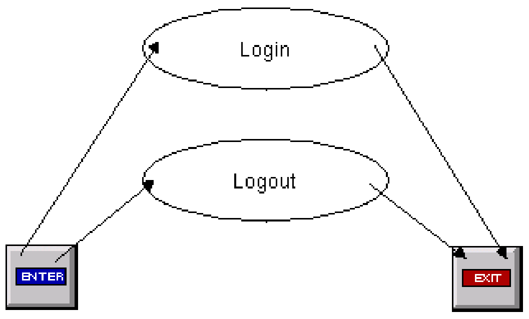


Figure 71. Application Interface component view in VCE

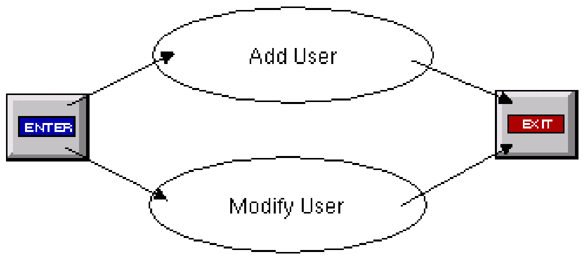


Figure 72. Account Services component view in VCE

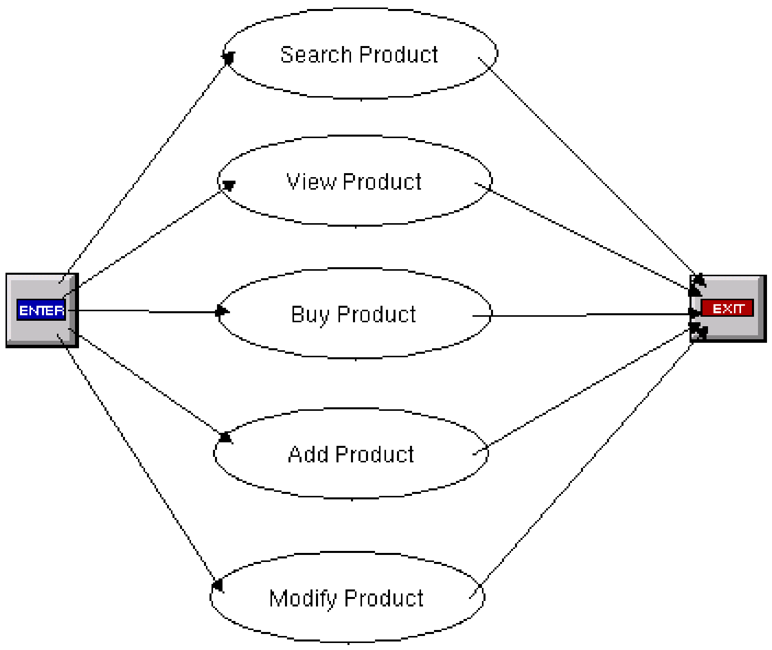


Figure 73. Product Services component view in VCE

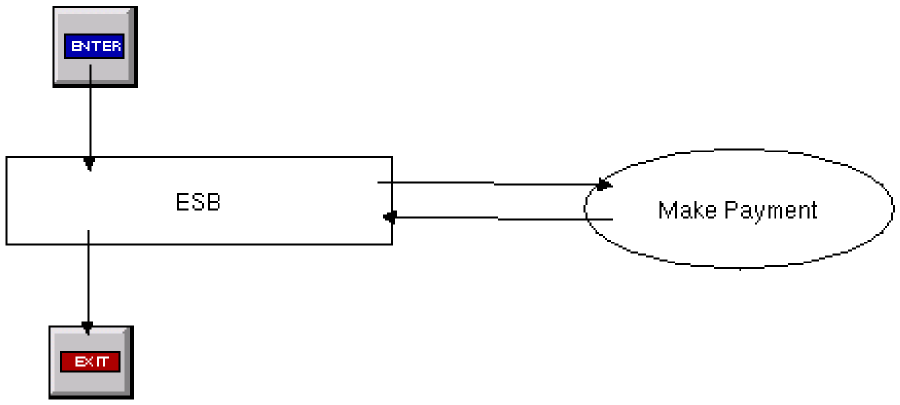


Figure 74. Payment component view in VCE

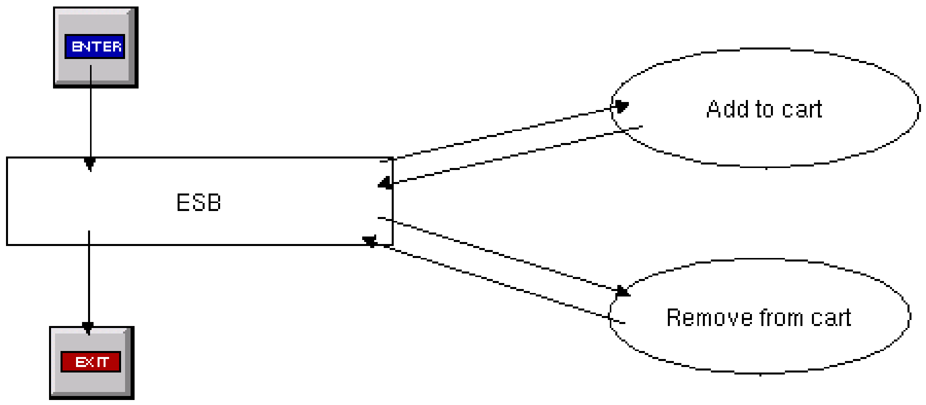


Figure 75. Shopping Cart component view in VCE

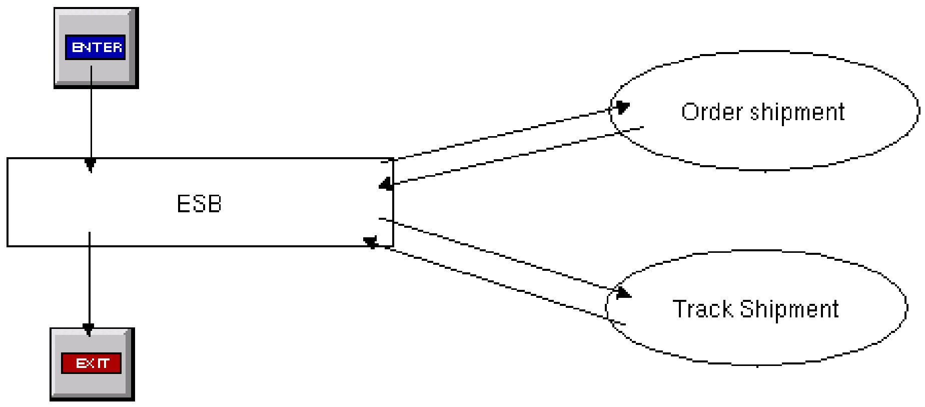


Figure 76. Delivery component view in VCE

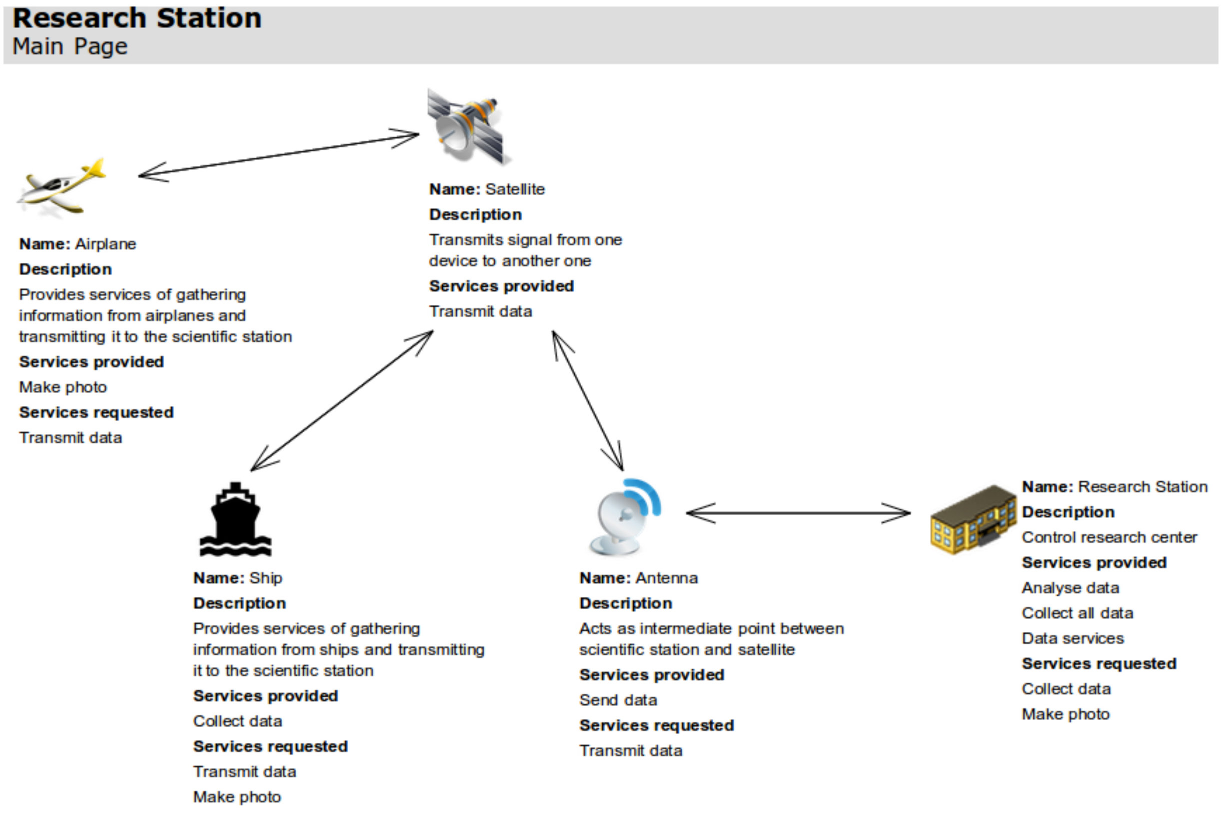
4.2.4 *Research station*

4.2.4.1 Overview

This example describes Research Station System. Research Station System is a family of systems which can be used to automate different procedures of collecting data for research labs and organizations. In this example information is collected from air, using airplanes, and from sea using ships. All communication goes with the help of satellites.

The next two sections provide figures of the system architecture specification and the simulation model of the Research Station System.

4.2.4.2 System Architecture Specification



Research Station
Main Page

Name: Satellite
Description
Transmits signal from one device to another one
Services provided
Transmit data

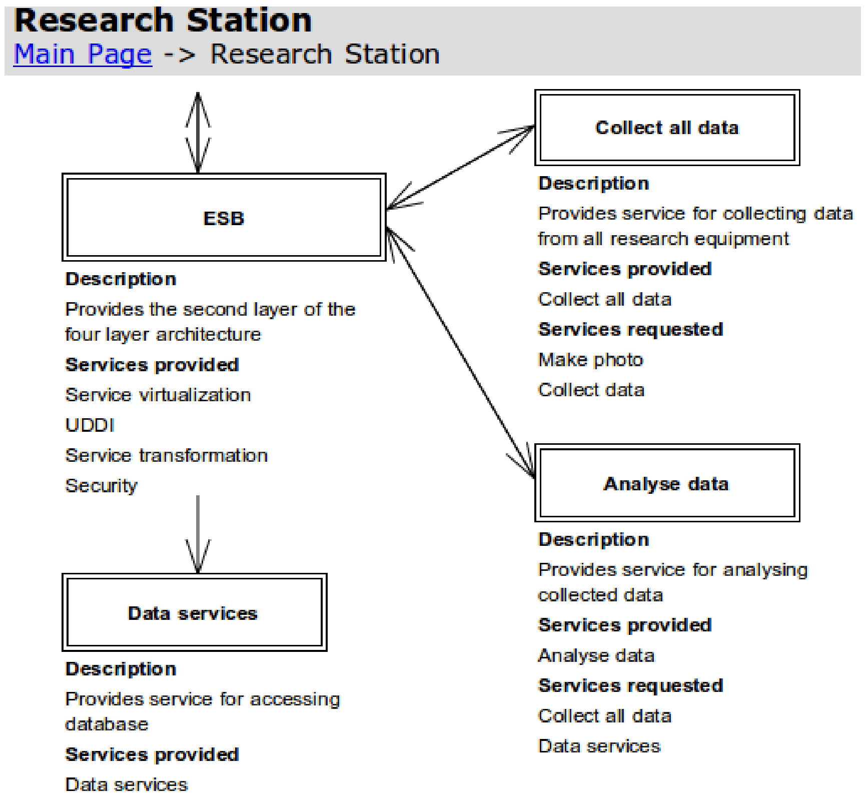
Name: Airplane
Description
Provides services of gathering information from airplanes and transmitting it to the scientific station
Services provided
Make photo
Services requested
Transmit data

Name: Ship
Description
Provides services of gathering information from ships and transmitting it to the scientific station
Services provided
Collect data
Services requested
Transmit data
Make photo

Name: Antenna
Description
Acts as intermediate point between scientific station and satellite
Services provided
Send data
Services requested
Transmit data

Name: Research Station
Description
Control research center
Services provided
Analyse data
Collect all data
Data services
Services requested
Collect data
Make photo

Figure 77. Top view of the Research Station architecture specification


Research Station

Main Page -> Research Station

ESB

Description
Provides the second layer of the four layer architecture

Services provided
Service virtualization
UDDI
Service transformation
Security

Collect all data

Description
Provides service for collecting data from all research equipment

Services provided
Collect all data

Services requested
Make photo
Collect data

Analyse data

Description
Provides service for analysing collected data

Services provided
Analyse data

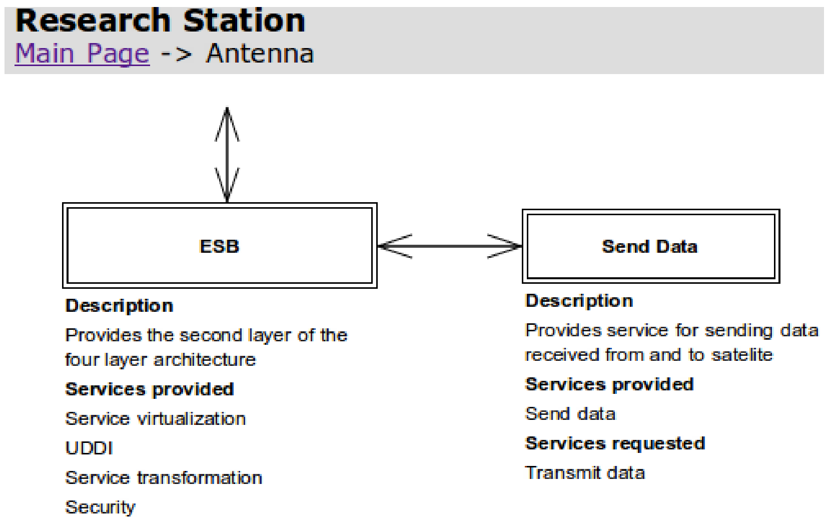
Services requested
Collect all data
Data services

Data services

Description
Provides service for accessing database

Services provided
Data services

Figure 78. Specification of the Research Station component


Research Station

Main Page -> Antenna

ESB

Description
Provides the second layer of the four layer architecture

Services provided
Service virtualization
UDDI
Service transformation
Security

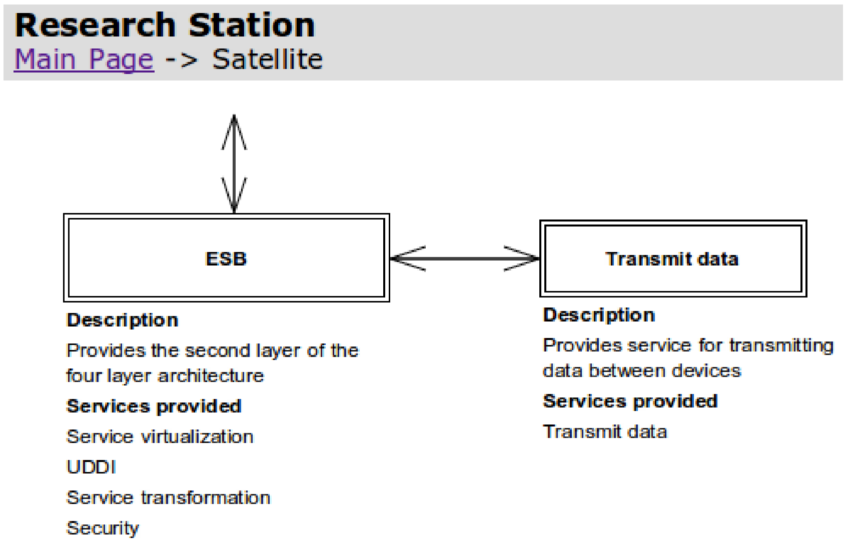
Send Data

Description
Provides service for sending data received from and to satelite

Services provided
Send data

Services requested
Transmit data

Figure 79. Specification of the Antenna component


Research Station

Main Page -> Satellite

Description

Provides the second layer of the four layer architecture

Services provided

Service virtualization

UDDI

Service transformation

Security

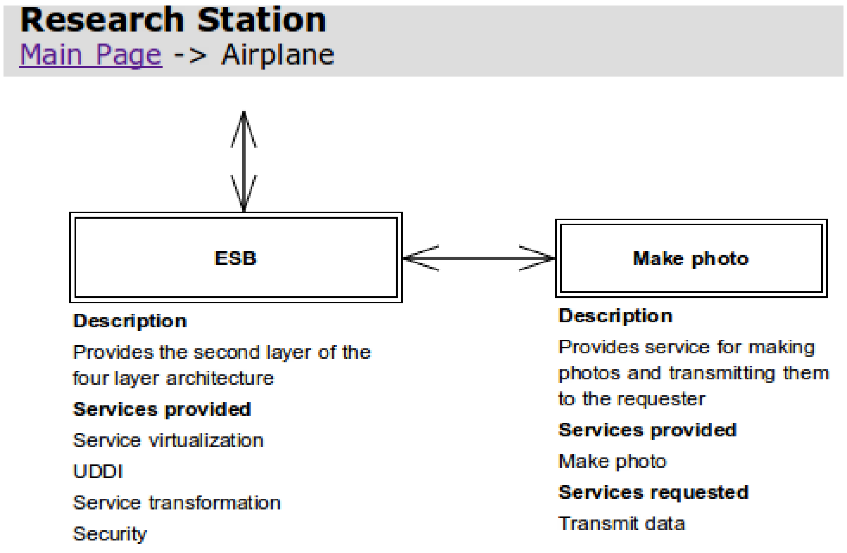
Description

Provides service for transmitting data between devices

Services provided

Transmit data

Figure 80. Specification of the Satellite component


Research Station

Main Page -> Airplane

Description

Provides the second layer of the four layer architecture

Services provided

Service virtualization

UDDI

Service transformation

Security

Description

Provides service for making photos and transmitting them to the requester

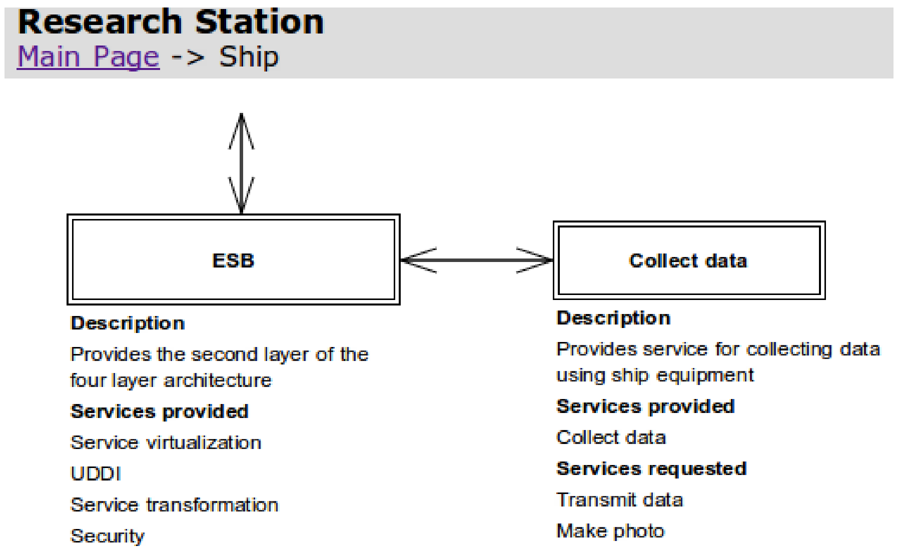
Services provided

Make photo

Services requested

Transmit data

Figure 81. Specification of the Airplane component



Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security

Description
Provides service for collecting data using ship equipment
Services provided
Collect data
Services requested
Transmit data
Make photo

Figure 82. Specification of the Ship component

4.2.4.3 Simulation Model

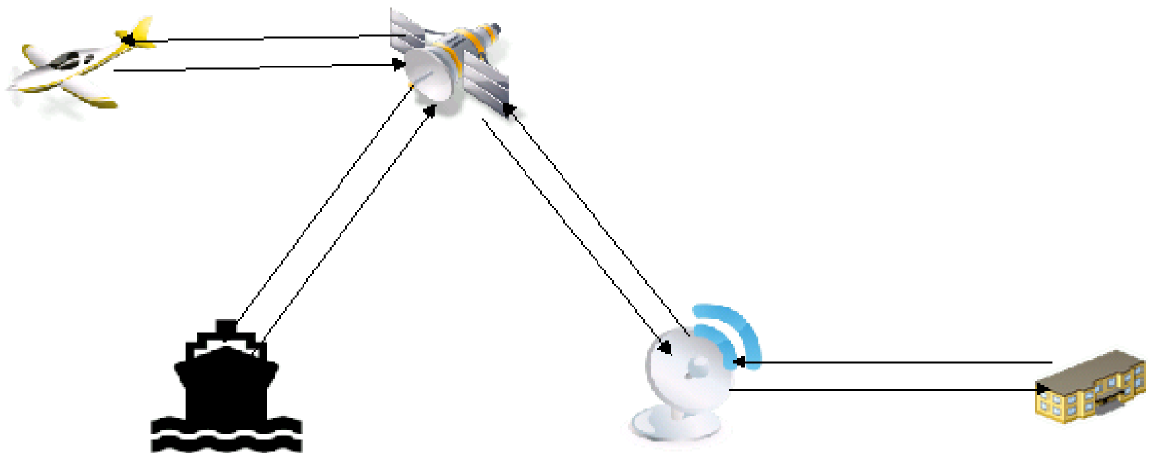


Figure 83. Top view of the simulation model of Research Station in VSE

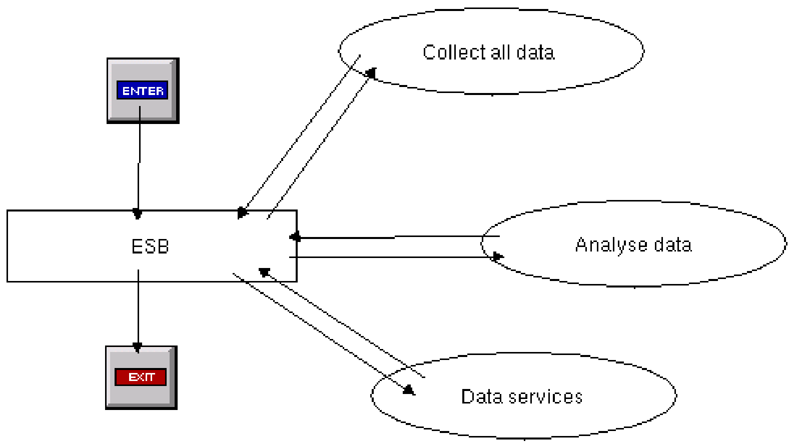


Figure 84. Research Station component view in VSE

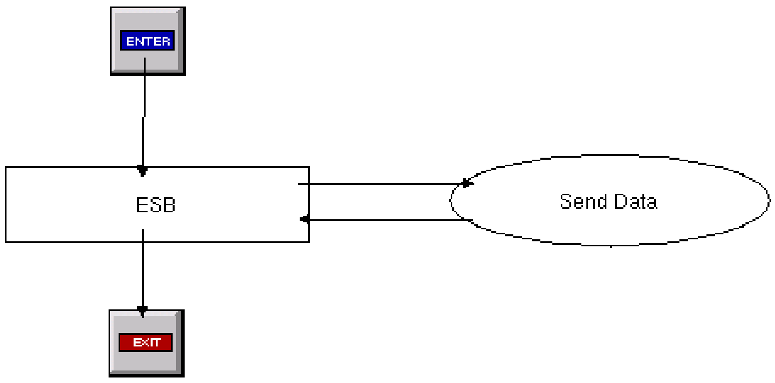


Figure 85. Antenna component view in VSE

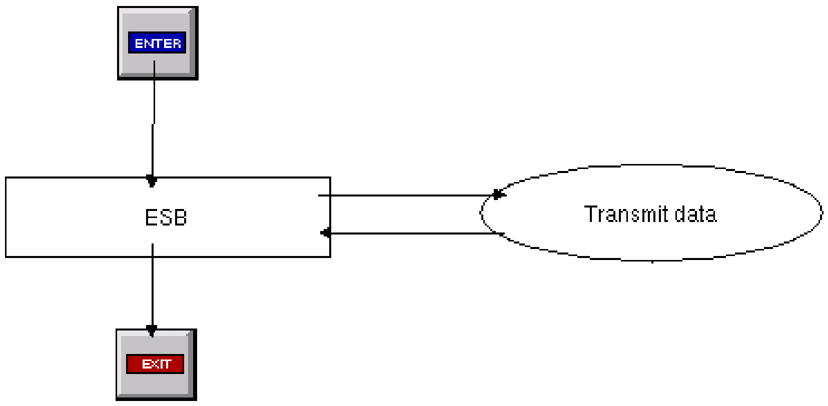


Figure 86. Satellite component view in VSE

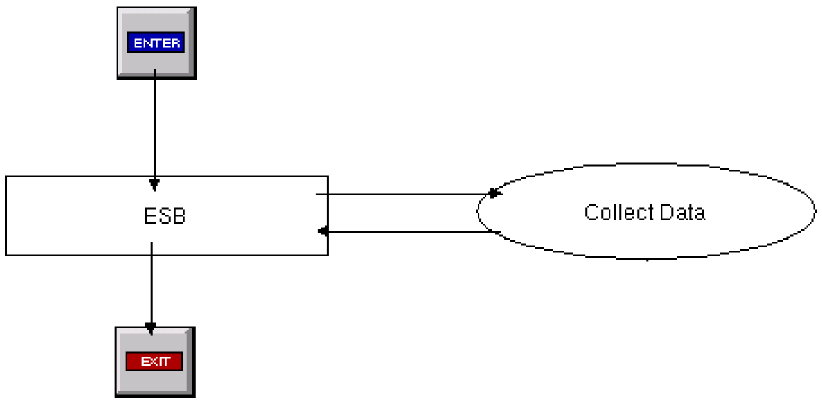


Figure 87. Ship component view in VSE

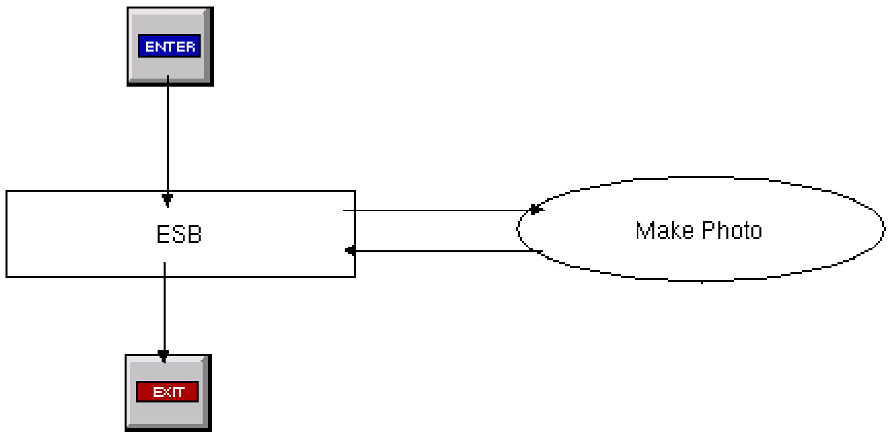


Figure 88. Airplane component view in VSE

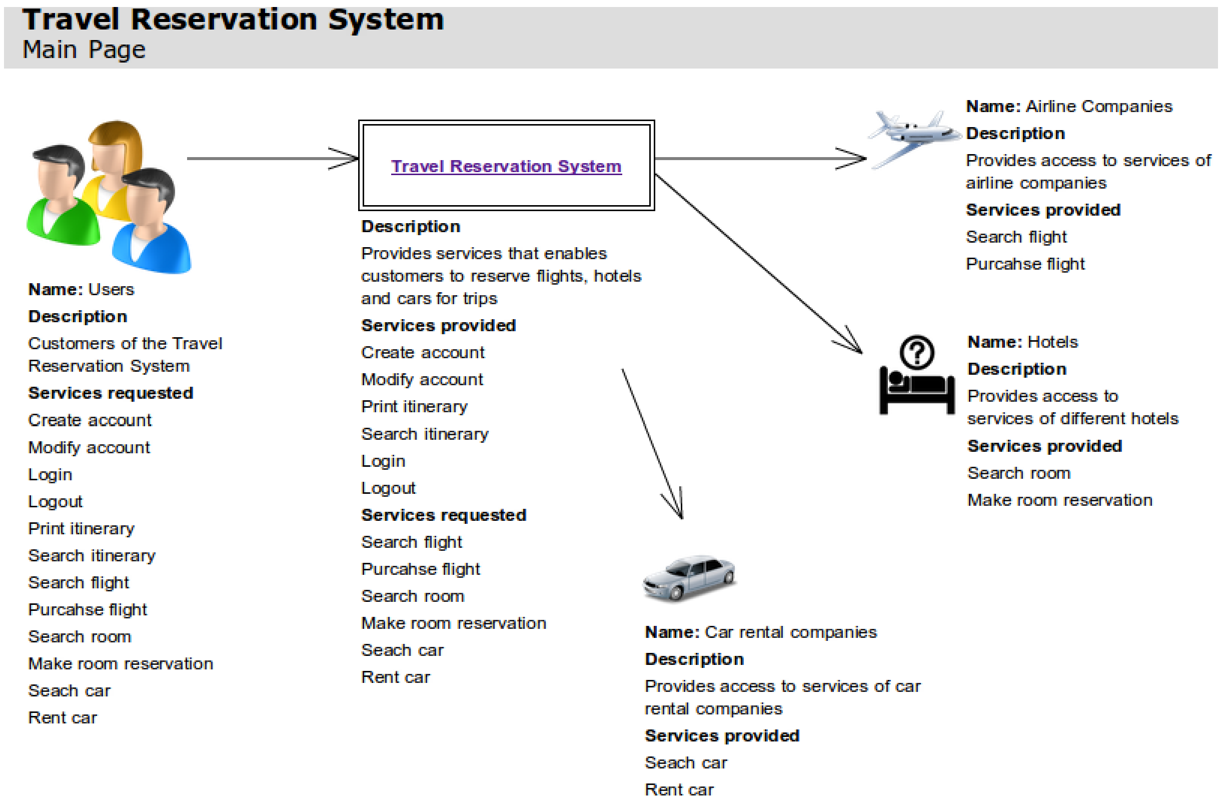
4.2.5 Travel Reservation System

4.2.5.1 Overview

Travel Reservation System is a family of systems which allows customers to plan their trips, purchase flights, make room reservations and rent cars. All these operations can be done online.

The next two sections provide figures of the system architecture specification and the simulation model of the Travel Reservation System.

4.2.5.2 System Architecture Specification



Name: Users
Description
Customers of the Travel Reservation System
Services requested
Create account
Modify account
Login
Logout
Print itinerary
Search itinerary
Search flight
Purcahse flight
Search room
Make room reservation
Seach car
Rent car

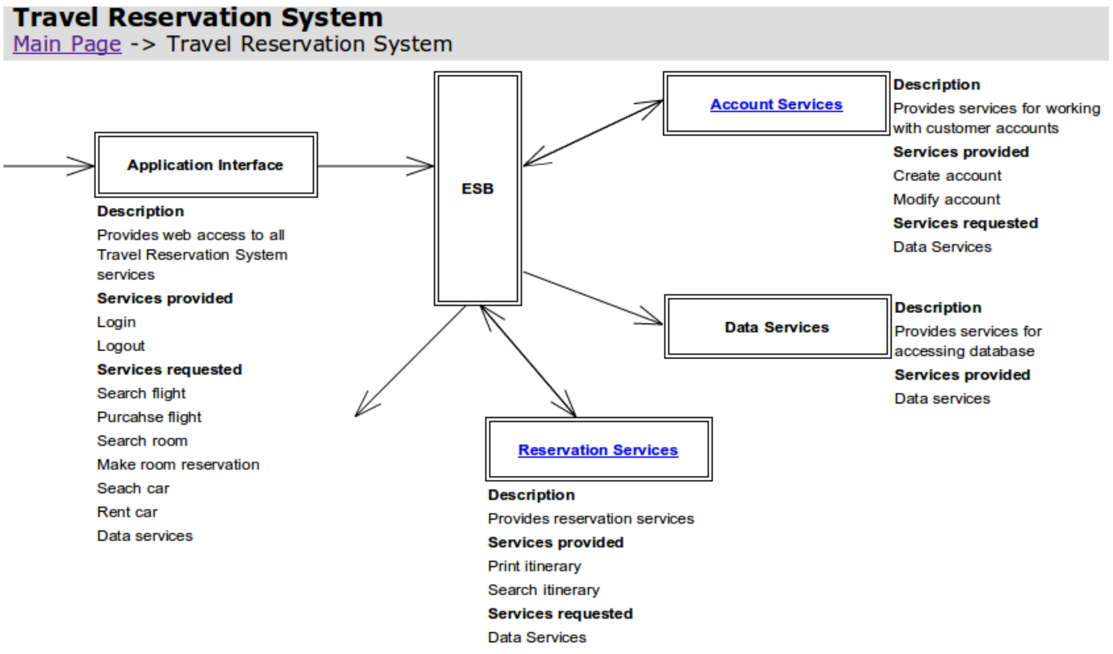
Description
Provides services that enables customers to reserve flights, hotels and cars for trips
Services provided
Create account
Modify account
Print itinerary
Search itinerary
Login
Logout
Services requested
Search flight
Purcahse flight
Search room
Make room reservation
Seach car
Rent car

Name: Airline Companies
Description
Provides access to services of airline companies
Services provided
Search flight
Purcahse flight

Name: Hotels
Description
Provides access to services of different hotels
Services provided
Search room
Make room reservation

Name: Car rental companies
Description
Provides access to services of car rental companies
Services provided
Seach car
Rent car

Figure 89. Top view of the Travel Reservation System architecture specification



Application Interface

Description
Provides web access to all Travel Reservation System services
Services provided
Login
Logout
Services requested
Search flight
Purcahse flight
Search room
Make room reservation
Seach car
Rent car
Data services

Account Services

Description
Provides services for working with customer accounts
Services provided
Create account
Modify account
Services requested
Data Services

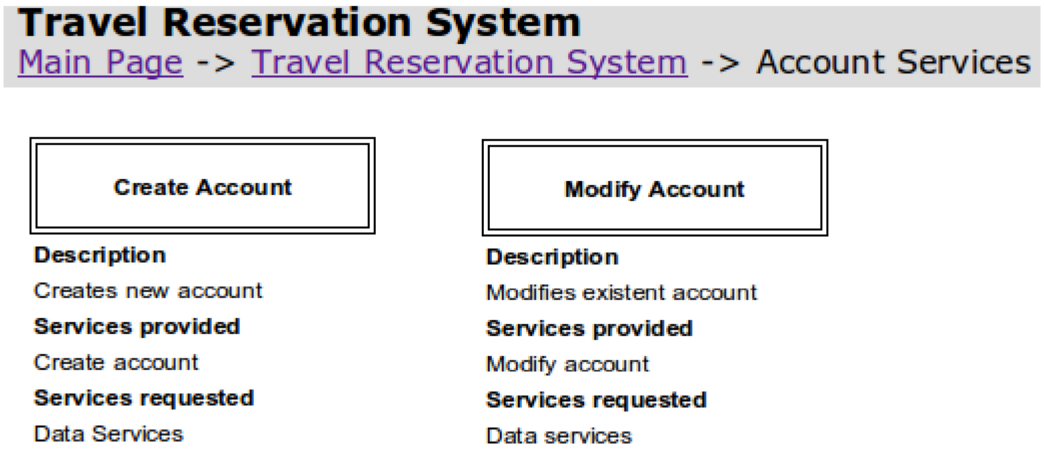
Data Services

Description
Provides services for accessing database
Services provided
Data services

Reservation Services

Description
Provides reservation services
Services provided
Print itinerary
Search itinerary
Services requested
Data Services

Figure 90. Specification of the Travel Reservation System component


Travel Reservation System
Main Page -> Travel Reservation System -> Account Services
Create Account
Modify Account

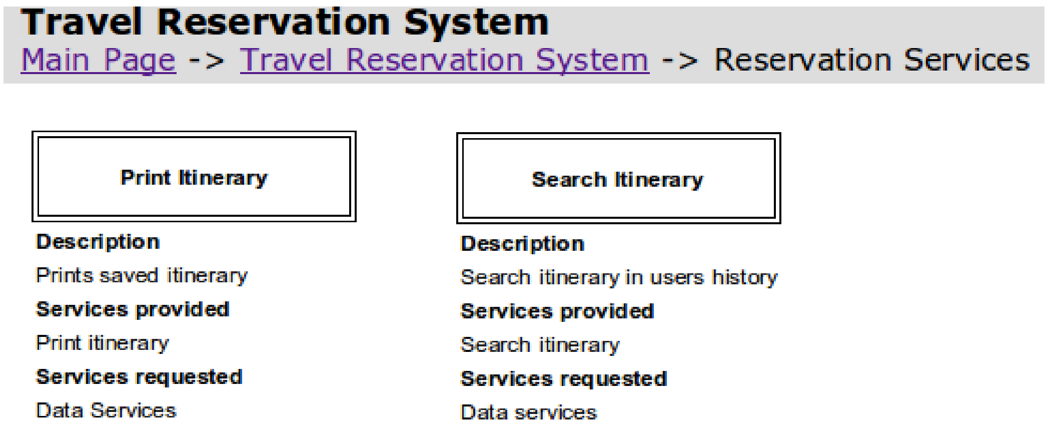
Create Account

Description
Creates new account
Services provided
Create account
Services requested
Data Services

Modify Account

Description
Modifies existent account
Services provided
Modify account
Services requested
Data services

Figure 91. Specification of the Account Services component


Travel Reservation System

Main Page -> Travel Reservation System -> Reservation Services

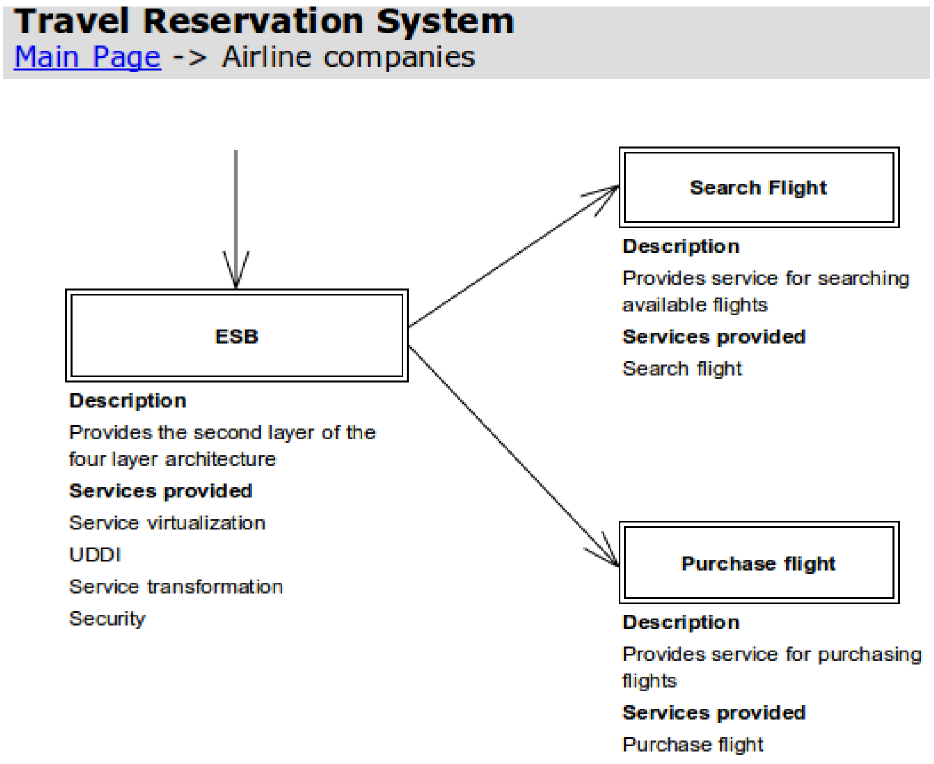
Print Itinerary

Description
Prints saved itinerary
Services provided
Print itinerary
Services requested
Data Services

Search Itinerary

Description
Search itinerary in users history
Services provided
Search itinerary
Services requested
Data services

Figure 92. Specification of the Reservation Services component


Travel Reservation System

Main Page -> Airline companies

ESB

Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security

Search Flight

Description
Provides service for searching available flights
Services provided
Search flight

Purchase flight

Description
Provides service for purchasing flights
Services provided
Purchase flight

Figure 93. Specification of the Airline Companies component

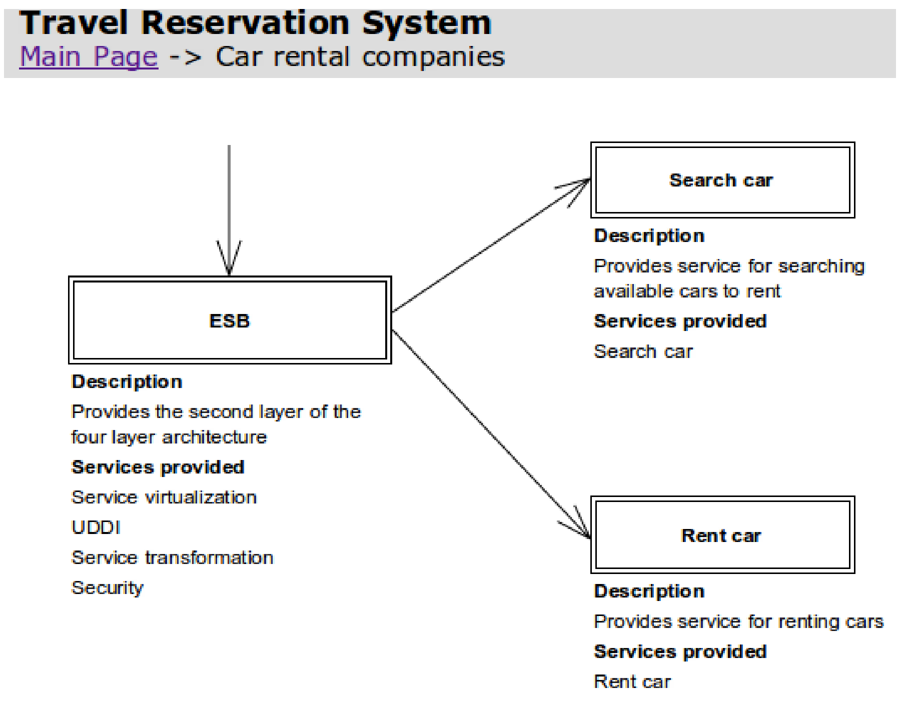

Travel Reservation System
Main Page -> Car rental companies
ESB
Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security
Search car
Description
Provides service for searching available cars to rent
Services provided
Search car
Rent car
Description
Provides service for renting cars
Services provided
Rent car

Figure 94. Specification of the Car Rental Companies component

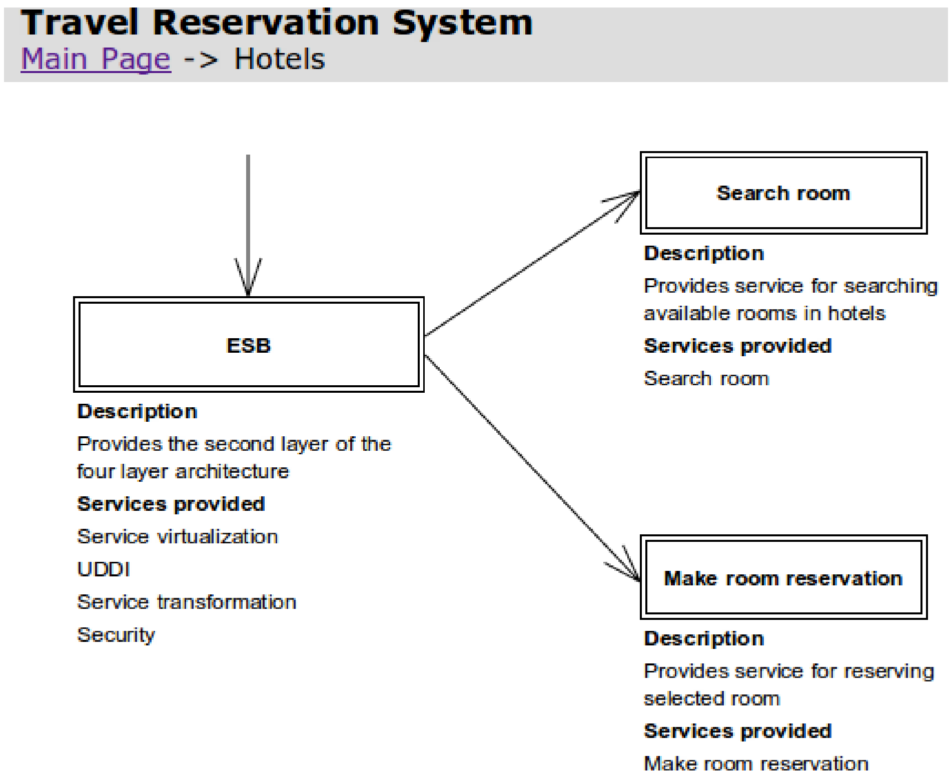


Figure 95. Specification of the Hotels component

4.2.5.3 Simulation Model

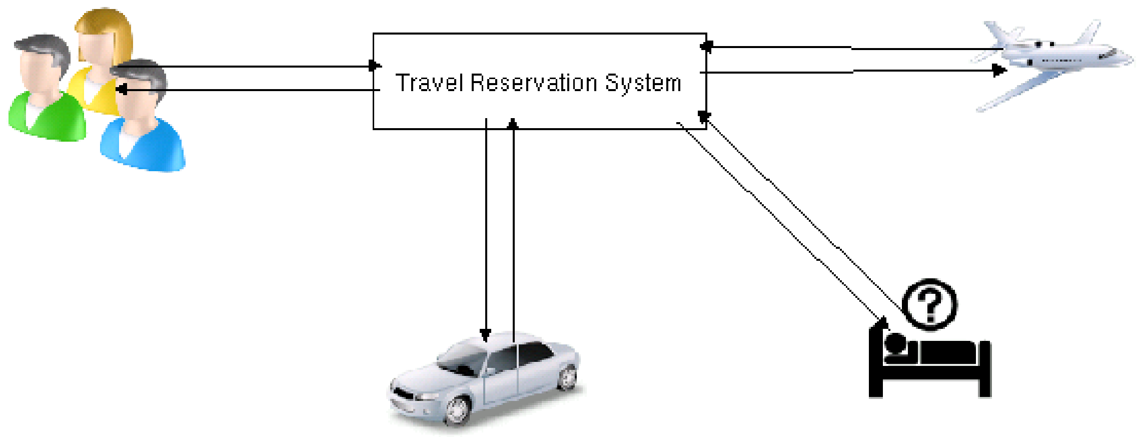


Figure 96. Top view of the simulation model of Travel Reservation System in VSE

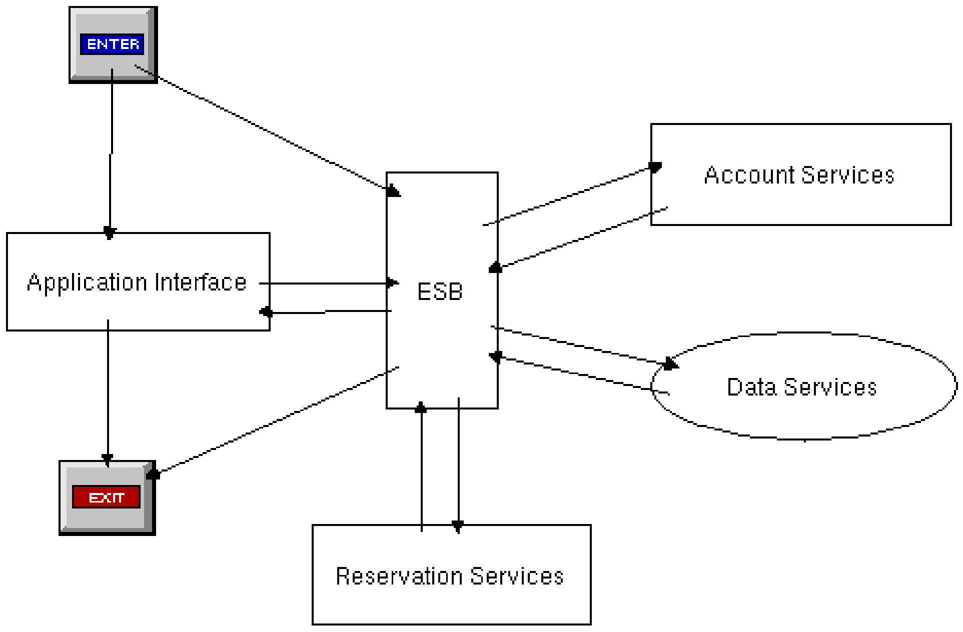


Figure 97. Travel Reservation System view in VSE

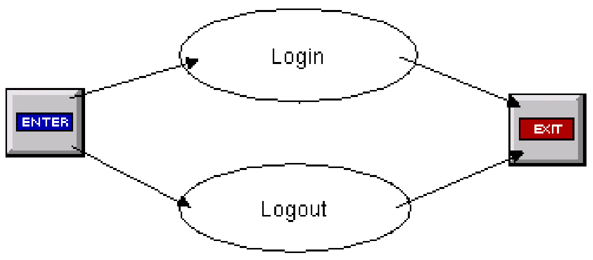


Figure 98. Application Interface component view in VSE

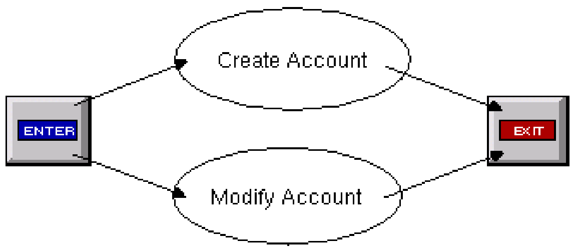


Figure 99. Account Services component view in VSE

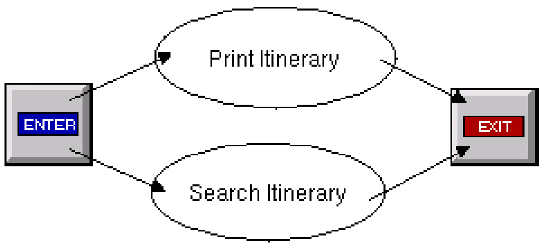


Figure 100. Reservation Services component view in VSE

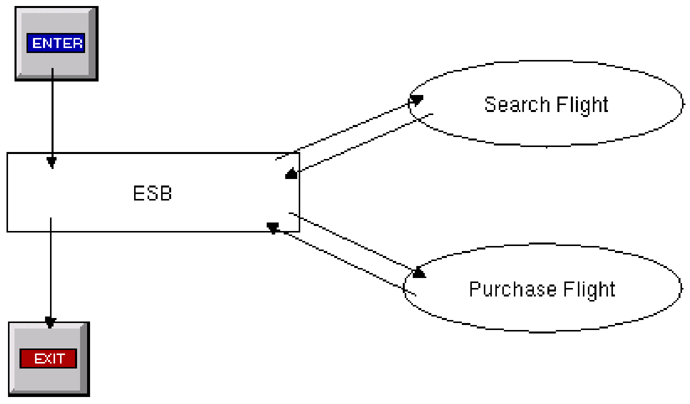


Figure 101. Airline Companies component view in VSE

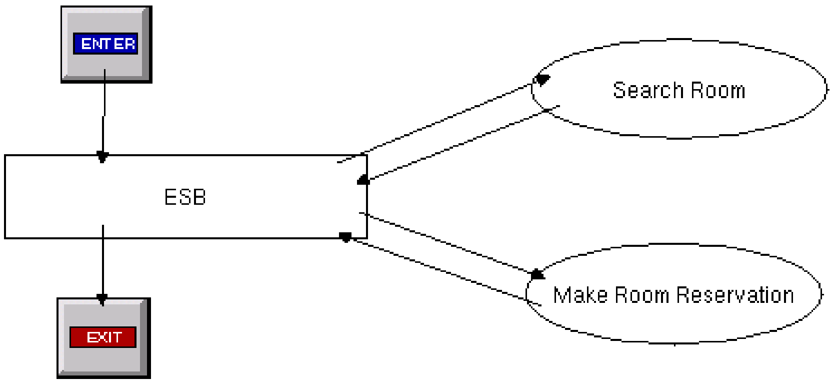


Figure 102. Hotels component view in VSE

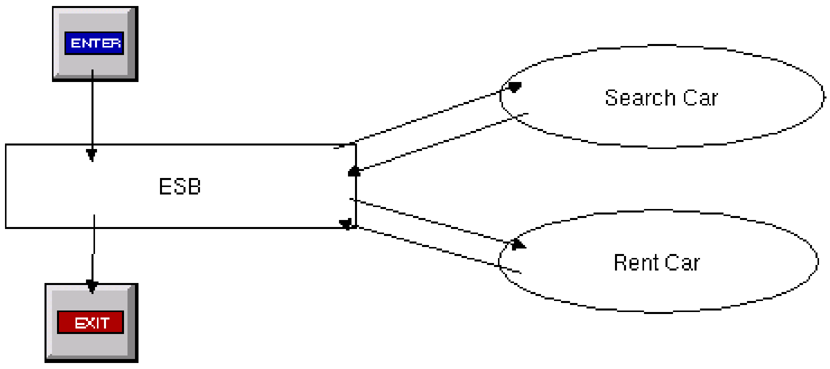


Figure 103. Car Rental Companies component view in VSE

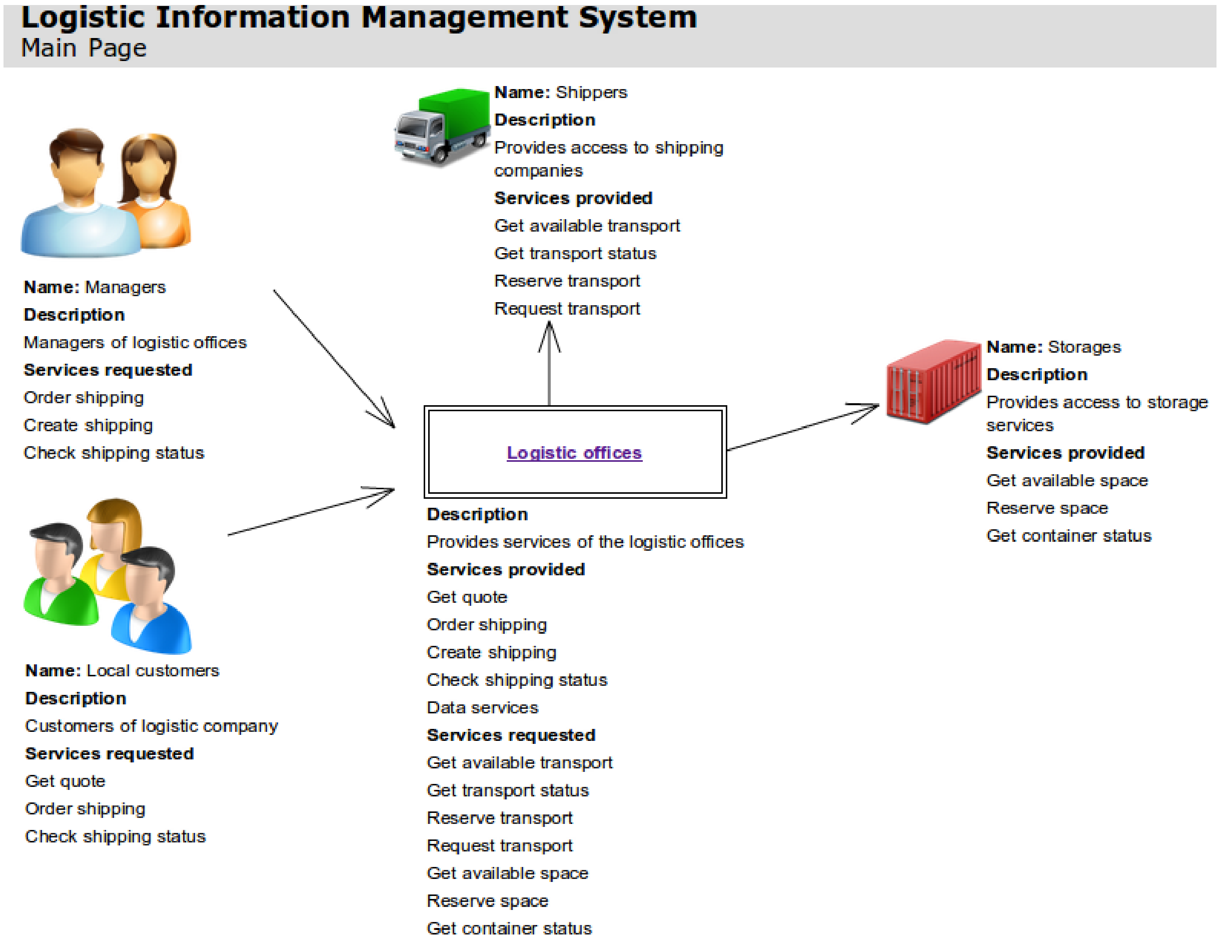
4.2.6 *Logistic Information Management System*

4.2.6.1 Overview

Logistic Information Management System is a family of systems which allows customers and managers to plan shipping operations using transportation companies and storages. Users and managers of the system can access it through the network.

The following two sections provide figures of the system architecture specification and the simulation model of the Logistic Information Management System.

4.2.6.2 System Architecture Specification



Logistic Information Management System
Main Page

Name: Shippers
Description
Provides access to shipping companies
Services provided
Get available transport
Get transport status
Reserve transport
Request transport

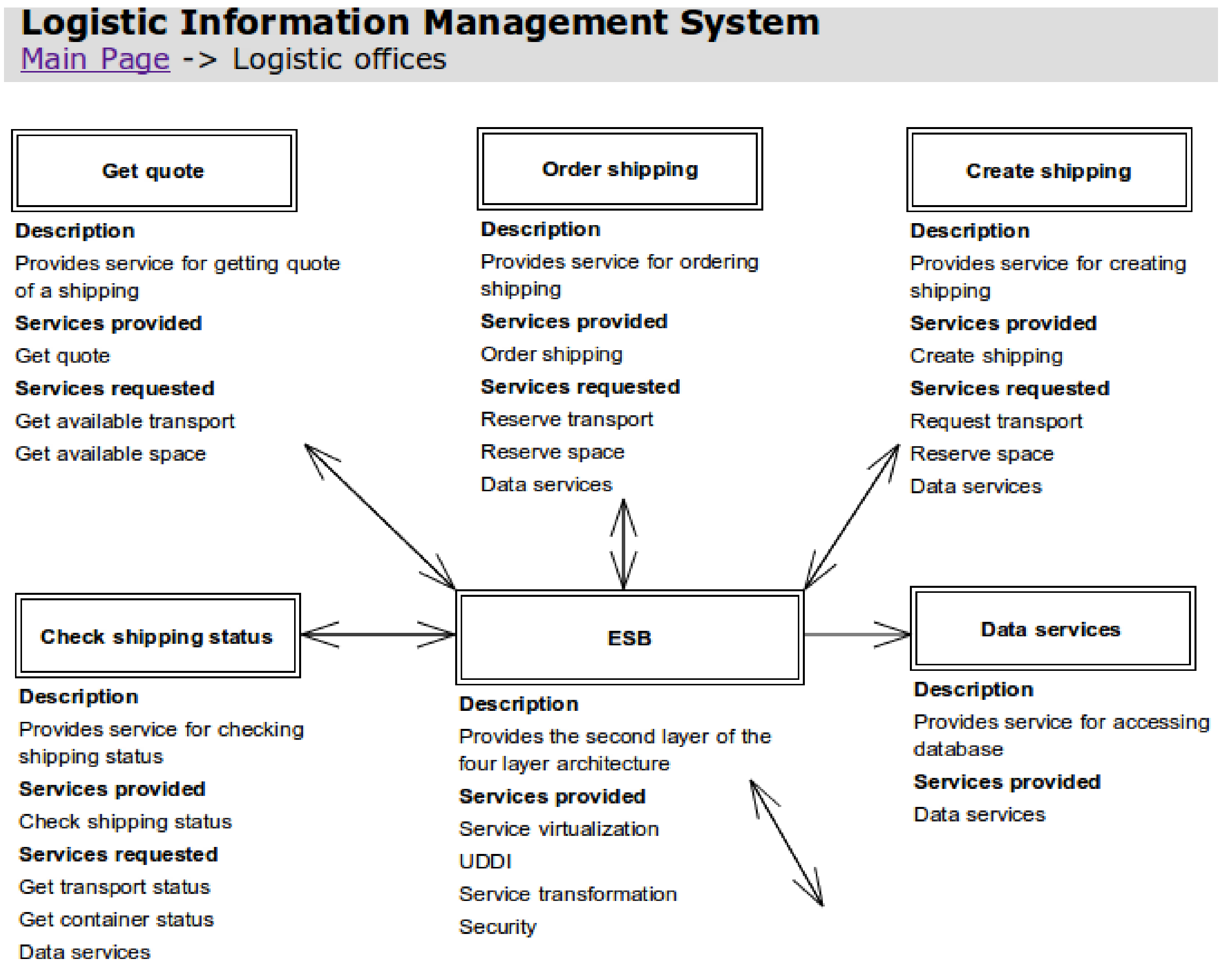
Name: Managers
Description
Managers of logistic offices
Services requested
Order shipping
Create shipping
Check shipping status

Name: Storages
Description
Provides access to storage services
Services provided
Get available space
Reserve space
Get container status

Logistic offices
Description
Provides services of the logistic offices
Services provided
Get quote
Order shipping
Create shipping
Check shipping status
Data services
Services requested
Get available transport
Get transport status
Reserve transport
Request transport
Get available space
Reserve space
Get container status

Name: Local customers
Description
Customers of logistic company
Services requested
Get quote
Order shipping
Check shipping status

Figure 104. Top view of the Logistic Information Management System architecture specification



Logistic Information Management System

Main Page -> Logistic offices

Get quote

Description

Provides service for getting quote of a shipping

Services provided

Get quote

Services requested

Get available transport

Get available space

Order shipping

Description

Provides service for ordering shipping

Services provided

Order shipping

Services requested

Reserve transport

Reserve space

Data services

Create shipping

Description

Provides service for creating shipping

Services provided

Create shipping

Services requested

Request transport

Reserve space

Data services

Check shipping status

Description

Provides service for checking shipping status

Services provided

Check shipping status

Services requested

Get transport status

Get container status

Data services

ESB

Description

Provides the second layer of the four layer architecture

Services provided

Service virtualization

UDDI

Service transformation

Security

Data services

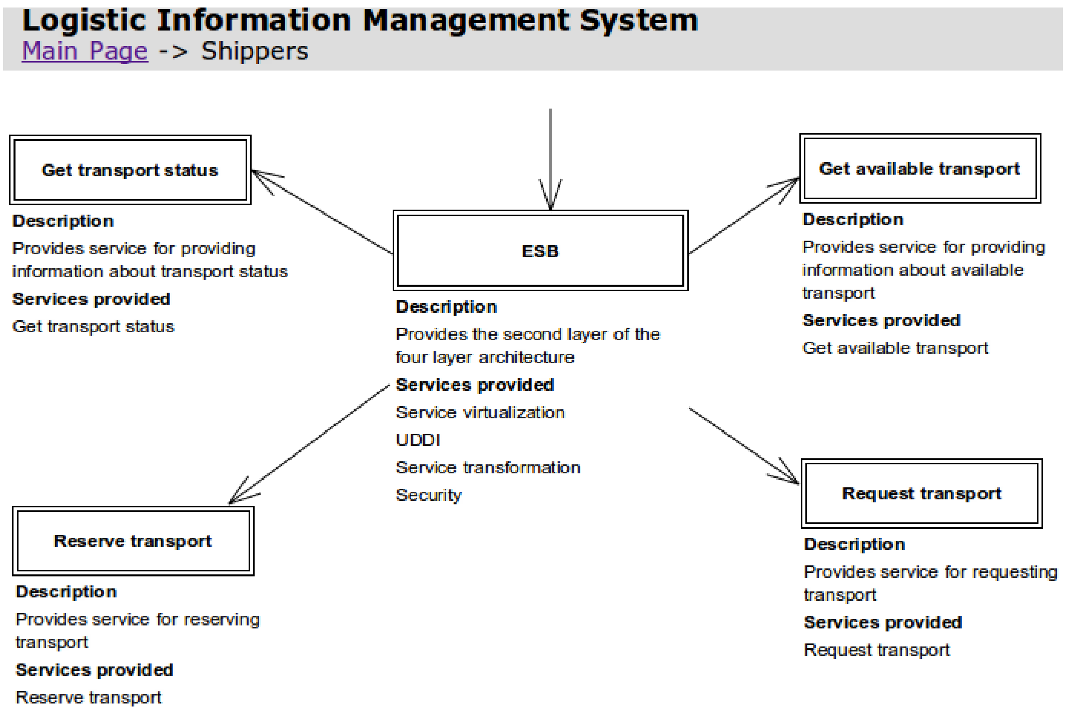
Description

Provides service for accessing database

Services provided

Data services

Figure 105. Specification of the Logistic Offices component


Logistic Information Management System
Main Page -> Shippers

ESB
Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security

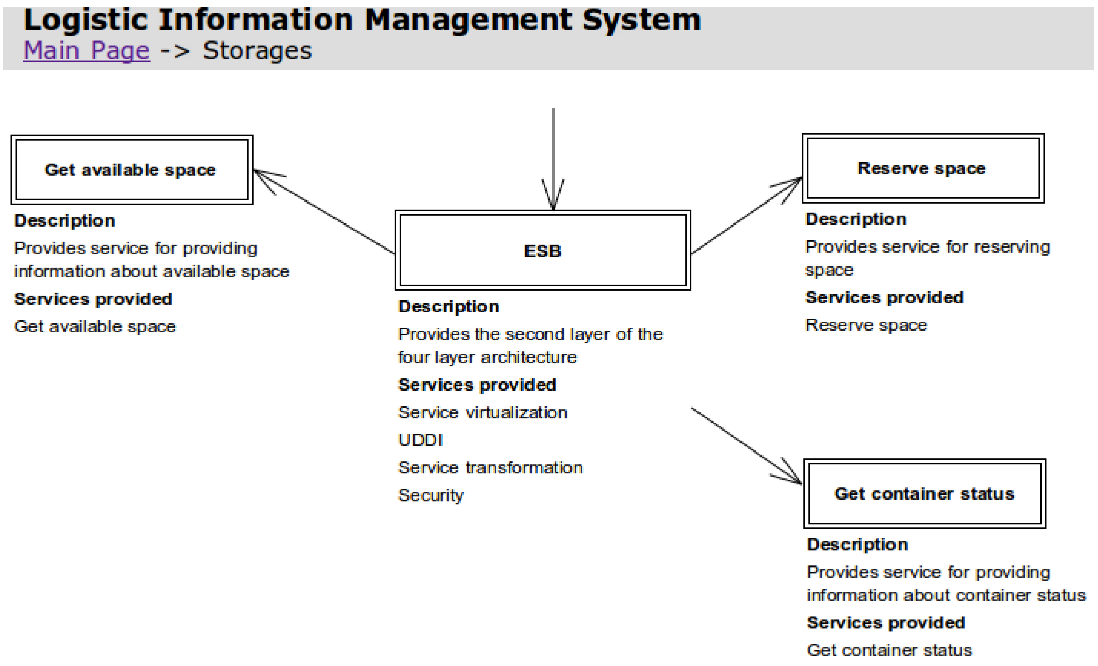
Get transport status
Description
Provides service for providing information about transport status
Services provided
Get transport status

Get available transport
Description
Provides service for providing information about available transport
Services provided
Get available transport

Reserve transport
Description
Provides service for reserving transport
Services provided
Reserve transport

Request transport
Description
Provides service for requesting transport
Services provided
Request transport

Figure 106. Specification of the Shippers component


Logistic Information Management System
Main Page -> Storages

ESB
Description
Provides the second layer of the four layer architecture
Services provided
Service virtualization
UDDI
Service transformation
Security

Get available space
Description
Provides service for providing information about available space
Services provided
Get available space

Reserve space
Description
Provides service for reserving space
Services provided
Reserve space

Get container status
Description
Provides service for providing information about container status
Services provided
Get container status

Figure 107. Specification of the Storages component

4.2.6.3 Simulation Model

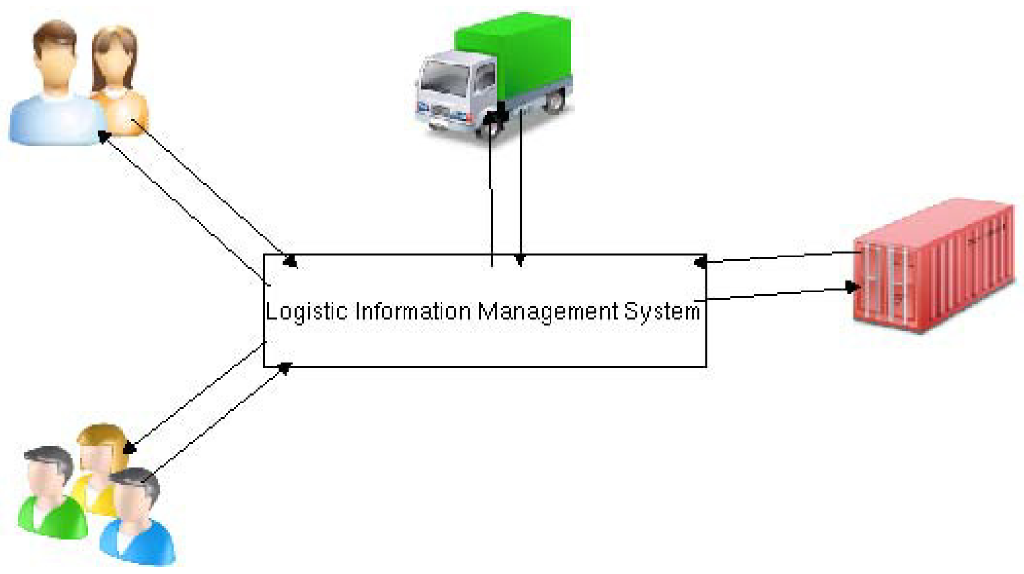


Figure 108. Top view of the simulation model of Logistic Information Management System in VSE

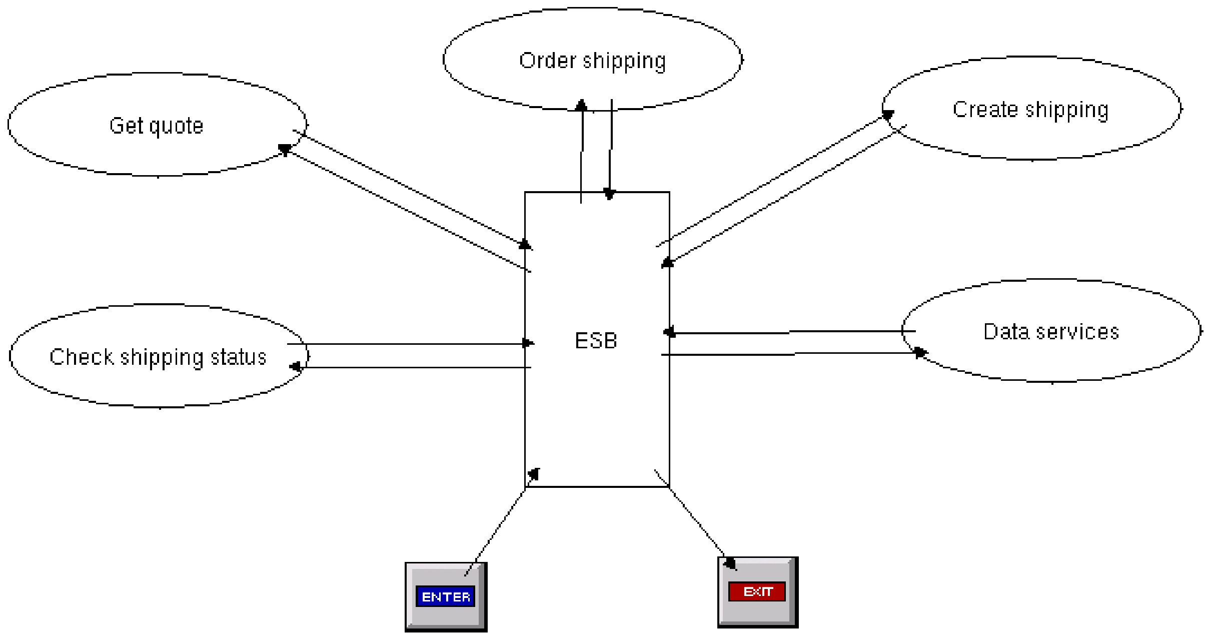


Figure 109. Logistic Offices component view in VSE

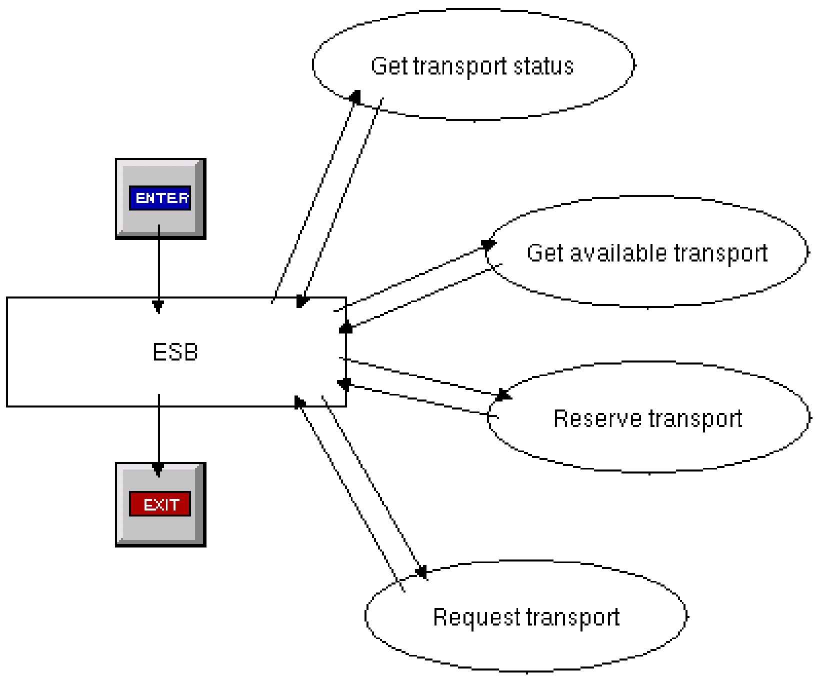


Figure 110. Shippers component view in VSE

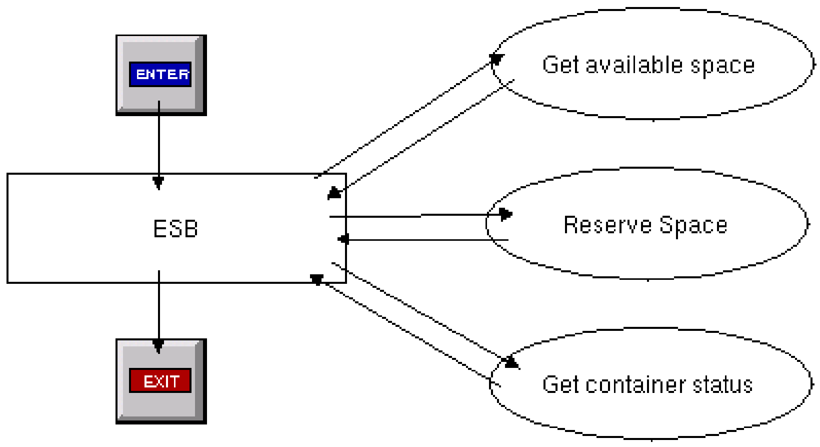


Figure 111. Storages component view in VSE

CHAPTER 5: CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

This thesis presents a CF for specifying the architecture of a network-centric software-based system of systems. The CF provides the beginning part of a larger research effort. The main goal of the overall research is to employ the automation-based software paradigm and to automatically generate a visual simulation model of a system architecture, with which experiments can be conducted to assess the dynamic characteristics of that architecture.

The developed CF enables the automatic generation of a visual simulation model representing a system architecture. It is based on such well-recognized ideas as separation of concerns, top-down approach and open system paradigm. The elements of the CF are grouped in two categories: static structure and dynamic structure. Static structure group contains the following elements:

- Component
- Connection
- Service

And dynamic structure group contains only one element: Message.

The elements of the static structure group are used to create the static representation of the network-centric system architecture specifications. Component element should satisfy the next four criteria:

- Be as a whole and complete unit of responsibility
- Minimize the dependency on other components to perform its core function
- Should have a well-defined behavior
- Should represent a separate unit of functionality

Top-down approach is used for defying component elements. Connection element is used to specify the interaction between the component elements. And service element represents business functionality which is provided by component.

Message element of the dynamic structure group constituents the elements of the static structure group to be used in visual simulation of specified system architecture. During simulation process message element represents a single action of communication between the components, i.e. a remote procedure call or invocation, message passing etc.

The proposed Conceptual Framework was specified to be as simple as possible, counting on easy extensibility and applicability with other enterprise architecture frameworks.

5.2 Contributions

Two major contributions of the research described herein include:

1. *A Conceptual Framework for specifying the architecture of a network-centric software-based system of systems.* The traditional use of diagrams and graphical images for network-centric software-based architecture specification does not provide the required level of support. Each diagram is two-dimensional static graphical and textual representations that do not reveal the dynamic characteristics of a system architecture. The conceptual framework, developed in the research described herein, enables the automatic generation of a visual simulation model representing a system architecture.

2. *The applicability of the Conceptual Framework was evaluated.* The proposed CF is evaluated in half a dozen case studies to demonstrate that it provides the necessary elements for automatic generation of a simulation model as the description of a complex system of systems architecture.

5.3 Future Research

The developed CF provides the beginning part of a larger research effort. The main goal of the overall research is to employ the automation-based software paradigm and to automatically generate a visual simulation model of a system architecture, with which experiments can be conducted to assess the dynamic characteristics of that architecture.

The CF can also be extended for specification of dynamic network-centric system architectures, i.e. system architectures which evolve during runtime.

BIBLIOGRAPHY

Alpergin F. (2007), "A Network-Centric System Architecture for Online Learning Environments", M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Ambler S. (2007), "Extending the RUP with the Zachman Framework", Ambysoft, Canada, http://www.enterpriseunifiedprocess.com/essays/zachmanFramework.html.

Balci O. (2009a), "The Open System Paradigm", CS5704 Course Notes, http://manta.cs.vt.edu/cs5704.

Balci O. (2009b), "Process Models", CS5704 Course Notes, http://manta.cs.vt.edu/cs5704.

Balci O. (2009c), "Network-Centric Architectures Overview", CS5704 Course Notes, http://manta.cs.vt.edu/cs5704.

Booch G. (2009), "SOA as an Architectural Pattern: Best Practices in Software Architecture", *The International SOA Symposium*, Arlington, VA, http://www.infoq.com/presentations/Architectural-Patterns.

Chigani A., Arthur J.D., and Bohner S. (2007), "Architecting Network-Centric Software Systems: A Style-Based Beginning", In *Proceedings of the 31st IEEE Software Engineering Workshop*, Columbia, MD, pp. 290-299.

Chigani A. (2007), "Guiding Network-Centric Architectural Design: A Style-Based Approach", M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Clements P., Bachman F., Bass L., Garlan D., Ivers J., Little R., Nord R., Stafford J. (2003), *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, Boston, MA.

Dhanji R.P. (2009), *Dependency Injection*, Manning Publications, Greenwich, CT.

DoDAF (2009a), "DoD Architecture Framework Version 2.0 Volume 1: Introduction, Overview, and Concepts", Architecture Framework Working Group, Department of Defense, Washington, D.C., May 28.

DoDAF (2009b), "DoD Architecture Framework Version 2.0 Volume 2: Architectural Data and Models", Architecture Framework Working Group, Department of Defense, Washington, D.C., May 28.

Fremantle P., Weerawarana S. and Khalaf R. (2002), "Enterprise services", *Communications of the ACM 45*, 10, pp. 77-82.

Garlan D. (2000), "Software Architecture: a Roadmap", In *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland, pp. 91-101.

Henry T.S. (2007), “Architecture-Centric Project Estimation”, M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Hilliard R. (2007), “All About IEEE Std 1471”, http://www.iso-architecture.org/ieee-1471/docs/all-about-ieee-1471.pdf.

IBM (2010), IBM Rational System Architect homepage, http://www-01.ibm.com/software/awdtools/systemarchitect/.

IEEE (2000), “ANSI/IEEE Std 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems”, IEEE Architecture Working Group, October 2000.

Josuttis N. (2007), *SOA in Practice*, First Edition, O’Reilly Media, Sebastopol, CA.

Metastorm (2008), “Matastorm ProVision, Product Overview”, Metastorm, Baltimore, MD, http://www.metastorm.com/products/product_sheets/Metastorm_ProVision_Product_Overview.pdf.

Microsoft (2004), “Understanding Service-Oriented Architecture”, Microsoft Corporation, Redmond, Washington, http://msdn.microsoft.com/en-us/library/aa480021.aspx.

Microsoft (2007), “COM: Component Object Model Technologies”, Microsoft Corporation, Redmond, Washington, http://www.microsoft.com/com/default.mspx.

Monroe R.T., Kompanek D., Melton R., and Garlan D. (1997), “Stylized Architecture, Design Patterns, and Objects”, *IEEE Software, 14(1)*, pp. 43-52.

OMG (2010), “OMG CORBA Homepage”, http://www.corba.org.

Orca Computer (2002), “Visual Simulation Environment”, Orca Computer, Blacksburg, VA, http://www.orcacomputer.com/vse/VSE.html.

Papazoglou M.P. and Heuvel W. (2007), “Service oriented architectures: approaches, technologies and research issues”, *The VLDB Journal 16*, 3 (Jul. 2007), pp. 389-414.

Paulish D. J. (2002), *Architecture-Centric Software Project Management: A Practical Guide*, Addison-Wesley, Boston, MA.

Ravichandar R. (2008), “Capabilities Engineering”, Ph.D. Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Reese G. (2000), *Database programming with JDBC and Java*, Second Edition, O’Reilly Media, Sebastopol, CA.

Sessions R. (2007), "A comparison of the Top Four Enterprise-Architecture Methodologies", Microsoft Corporation, Redmond, Washington, http://msdn.microsoft.com/en-us/library/bb466232.aspx.

The Open Group (2009a), "TOGAF: Introduction", http://www.opengroup.org/architecture/togaf9-doc/arch/chap01.html.

The Open Group (2009b), "TOGAF: Core Concepts", http://www.opengroup.org/architecture/togaf9-doc/arch/chap02.html.

The Open Group (2009c), "TOGAF: Architecture Development Model", http://www.opengroup.org/architecture/togaf9-doc/arch/chap05.html.

The Open Group (2009d), "TOGAF: Enterprise Continuum", http://www.opengroup.org/architecture/togaf9-doc/arch/chap39.html.

The Open Group (2009e), "TOGAF: Introduction to the Architecture Content Framework", http://opengroup.org/architecture/togaf9-doc/arch/chap33.html.

The Open Group (2009f), "TOGAF: Architecture Capability Framework", http://opengroup.org/architecture/togaf9-doc/arch/chap45.html.

W3C (1999), "HTML 4.01 Specification", http://www.w3.org/TR/html401/.

W3C (2006), "Web Services", http://www.w3c.org/2002/ws/.

Zachman J. (1987), "A framework for information systems architecture", *IBM Systems Journal 26*, 3, pp. 276-292.

Zachman J. (2008), "The Zachman Framework: The Official Concise Definition", Zachman International, La Canada, CA, http://www.zachmaninternational.com/index.php/the-zachman-framework.