Context Sensitive Interaction Interoperability for Distributed Virtual Environments

Hussein M. Ahmed

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Applications

May 28, 2010

Blacksburg, Virginia

Keywords: Distributed Systems, Interaction Interoperability, Distributed Virtual Environments, Scalability, Context Awareness, Service Centric Architectures.

Context Sensitive Interaction Interoperability for Distributed Virtual Environments

Hussein M. Ahmed

(ABSTRACT)

The number and types of input devices and related interaction technique types are growing rapidly. Innovative input devices such as game controllers are no longer used just for games, propriety consoles and specific applications, they are also used in many distributed virtual environments, especially the so-called serious virtual environments.

In this dissertation a distributed, service based framework is presented to offer context-sensitive interaction interoperability that can support mapping between input devices and suitable application tasks given the attributes (device, applications, users, and interaction techniques) and the current user context without negatively impacting performances of large scale distributed environments.

The mapping is dynamic and context sensitive taking into account the context dimensions of both the virtual and real planes. What device or device component to use, how and when to use them depend on the application, task performed, the user and the overall context, including location and presence of other users. Another use of interaction interoperability is as a testbed for input devices, and interaction techniques making it possible to test reality based interfaces and interaction techniques with legacy applications.

The dissertation provides a description how the framework provides these affordances and a discussion of motivations, goals and the addressed challenges. Several proof of the concept implementations were developed and an evaluation of the framework performance (in terms of system characteristics) demonstrates viability, scalability and negligible delays.

To my great parents and wonderful wife.

Acknowledgments

A five years journey could not have been possible and bearable without great support and love. I went through a lot of tough times, questions and decisions and I could not have made it through all of this without being surrounded by amazing people.

I would like to start first to thank my advisor Dr. Denis Gračanin. Dr. Gračanin had always been supportive and pushing the bar for me higher and higher, despite I didn't like it then but looking back I do really appreciate it. Through our long discussions, over Skype when I was back in Egypt and in person when I got on campus, I have learned a lot of invaluable things from him related and unrelated to my dissertation. He taught me that time management is the key and that I should always focus on quality not quantity. I cannot forget to thank him also for all the great chocolate and home made ice cream that we were always looking forward to.

I would like to express my deepest appreciation to my dissertation committee. I always look up to Dr. Yong Cao for his exceptional industrial experience and switch to Academia again with great achievements grounded in practical engineering. He was so gracious to always share his insights and experiences with me. Dr. Woodrow Winchester III with the background in Industrial and Systems Engineering (including Human Computer Interaction) provided a valuable feedback from a different yet crucial perspective. I would also like to thank Dr. Chris North for influencing the research direction of my dissertation making it more practical and with a broader impact. was fortunate also with the addition of Dr. Eli Tilevich to my committee. His software systems and languages expertise added another great perspective. In just a short period, he provided a lot of insight about my dissertation practicality, application and how it aligns with the global movement to cloud computing. It was also great of him helping me to phrase my thesis in one sentence clearly stating my contributions.

I would like to express my gratitude to Dr. Yasser Hanafy and Dr. Sedki Riad, the two key persons behind the great success of the VT-MENA program. I was fortunate to be one of the five who were selected to get the program started back in 2005 and I can't thank them enough for such an amazing endeavor that have changed every aspect of my life. A huge thank you also to Mustafa El-Nainay and his wife Neveen. After we had moved from Egypt to Blacksburg, they greatly helped us in literally everything and while they were in Blacksburg we enjoyed great weekends together.

Special thanks go to several organization and companies. Sandia National Laboratories provided full access to their propriety rule engine Jess. Crosscheck Networks provided access to SOAPSonar Enterprise Edition to evaluate the case study described in the dissertation. Project Wonderland team from Sun/Oracle provided technical support and help in integrating Wonderland with my framework.

During the time I spent on campus I got involved with a number of student orgnizations. I have served for two years as an officer of the Computer Science Graduate Council and Upsilon Pi Epsilon and I was a photojournalist for the CollegiateTimes. I learned a lot from my friends and I had lots of fun organizing events and helping out with others. I was excited about getting my photos published in the university newspaper. Thank you all guys, Andrew Waren, Rhonda Phillips, Patrick Butler, Mehmet Belgin, Jeremy Archuleta, Peter Radics, Tejinder Judge, Luke Mason and Mark Umansky for all the fun we had and all the skills and knowledge you have shared with me.

Finally, I could have not made it by any means that far without the love and support of my parents and my great wife Samah. My father gracefully invested relentlessly in my education, assuring that me and my brothers and sister got the best quality of education attainable. My mother always went beyond what any mother can go, in her support, love and in making us believe in ourselves and believe that anything we want to become we can sure easily achieve. As a family we never got separated and so staying here for two years being apart from them was too much to handle for me and them, yet whenever I felt down a phone call to them would energize me again and keep me going. At last, special thanks and appreciation to my life partner and the love of my life, my wife Samah. Being married for more than six years we have been always together, cherishing our happy days and persevering through the tough ones. During all my falls and my suspicions that I cannot make it through something, Samah was there to lift me up and push me forward. Since our engagement, Samah have been my ultimate partner, cheerleader and supporter.

To all of you who helped me or just accompanied me through my journey, thank you!

Contents

List of Figures

All images property of the author, unless otherwise noted.

Chapter 1

Introduction

The main premise of my research is that it is feasible to abstract input devices and interaction tasks and provide context sensitive mapping between devices and tasks in large scale distributed virtual environments (DVEs) without compromising performance.

In this dissertation, entitled *Context Sensitive Interaction Interoperability for Distributed Virtual Environments*, I elaborate on this premise and describe how to abstract input devices and interaction tasks in order to achieve suitable many to many mappings between devices and tasks given their descriptions. Using this abstraction I proceed to show how context sensitive mapping is done in a scalable way, without compromising system performance, to facilitate integration in DVEs with a large user base. This dissertation demonstrates then how abstraction, mapping and context awareness are all achieved in a distributed manner to ensure scalability and provide support for heterogeneous platforms.

The focus of this dissertation is on DVEs given the scalability demands of these environments with multimillion sized user bases, thousands of concurrent users and distributed infrastructures. More specialized types of applications and virtual environments (VEs) such as Collaborative Virtual Environments (CVEs), Serious VEs, Computer Games and others can are included in the presented framework.

With an ever growing number of innovative input devices the motivation is that users should be able to interact with applications using what they like from that vast variety of input devices that includes touch screens, accelerometer sensors, brain interfaces and a lot of other innovative controllers and input devices. Providing a personalized user interface and interactions has been shown to improve the user's performance [Benford et al., 1997].

The idea of personalization is more evident in CVEs. In a CVE, a number of users collaborate and perform tasks together using the same application. Usually, given that the CVE is a single application, there would be one common way of interacting with the CVE which should be used by all the collaborating users. In addition, to ensure that all the users will be able to access and use the CVE client, the interface is mostly designed based on the most common input devices which are typically the classical keyboard or/and mouse.

Personalization, as addressed by the majority of current systems, is based on visually altering the user interface or reconfiguring the input devices based on the user's preferences. Another characteristic of personalization in the majority of context sensitive systems is that only the real world context is considered while disregarding the virtual world context.

With sophisticated meta-virtual worlds, like Second Life (SL) [Sec, 2010], that support various activities ranging from business to entertainment, the user context in a virtual world should influence personalization.

Given these facts, the motivations behind the research include:

- Offering context-aware personalization with the goal of mapping input devices to interaction techniques to accomplish any afforded application task.
- Supporting legacy applications and interoperability between heterogeneous platforms.
- Bridging between virtual world and real world, eliciting the user context from the both worlds and replicating actions between the both worlds.

To highlight the motivations we can consider three different scenarios. The first one is about

Jane who wants to use a new device she just bought to access SL. When using SL in a business context, Jane would like the SL client to automatically switch from the new input device to the traditional keyboard and mouse. The second scenario is about an application developer, Eric, who would like to integrate context awareness into a DVE client he is building. Finally, the third scenario is about an interaction technique developer, Dan, who would like to test out a new input device that supports Reality Based Interaction with the VE Wonderland.

Scenario One

Since Jane is an ordinary user, she cannot modify the SL client to incorporate her new device. What Jane does then is that she gets and installs the necessary drivers for the device. She then access the Interaction Interoperability Framework to see if her device is listed. Fortunately, Jane finds her device and a link to download different device emulators for different platforms. Since she has a Mac notebook, Jane downloads a Mac device emulator and installs it on her machine.

What she did next was to run the framework client mediator that asks her for the device emulator being used. The mediator running locally, queries the framework backend, Jane's context is collected indicating that she is at home (through GPS), not busy and that she is at a party location in Second Life.

Given Jane's previous interactions, the keyboard and mouse still are considered the interaction devices to be used. Jane decides to try out the new device and she instructs the mediator to switch to that device. With that requested switch, the framework associates this decision with the current context and sends back to the emulator the needed mappings to map from the input device connected to the application tasks.

Using the framework in this way, Jane was able to incorporate a new device with a legacy application and get automatic mapping between the device and the application she is using depending on her context in both the real and virtual worlds.

Scenario Two

Eric is an application developer who wants to integrate context awareness into a client for a large scale DVE. Eric checks the Interaction Interoperability Framework for the devices he wants to support. He finds all the devices listed there, meaning that he has two options, either to get a provided adapter class to use with his development environment (with all the device supporting libraries and respective APIs) or he can get his device libraries and create his own event handler.

However, the application is not yet defined for the framework. In order for Eric to do so, he goes through the three step process of defining his application starting with the application basic data, followed by the application tasks and then all dimensions of the multi-dimensional tasks.

After Eric defines his application, since Eric will be programming for Mac and there is no adapter available for Cocoa applications, he creates a web service client proxy using the public WSDL of the framework. He then creates the necessary event handler for device and when executing, the application tasks "consume" the framework web service that checks for the user context and the input devices available for the user. Based on the user profile the service returns back an XML mapping tree for the different device events and the corresponding tasks in the application that should go with it given the current user, the user's context and suitability of such mapping.

Utilizing the framework, Eric was able to make his client context sensitive without hard-coding input devices associations to application tasks and yet the performance of the DVE is not impacted by this integration.

Scenario Three

Dan wants to test out a new input device ad provide new reality based interaction technique for a business oriented DVE Wonderland. Since the context is a major part of a reality based interaction, Dan wants to have an evaluation testbed that supports user context, social context and allow mapping the device and new interaction techniques to the Wonderland's afforded tasks.

Trying to find a solution, Dan decides to make use of the interaction interoperability framework. He finds Wonderland as an application already defined by the framework but obviously not for his new device. Dan starts then by defining his new device. He goes through the three phases of defining a device for the framework. He defines the device, the primitives that makes up the input device and then the details of each of the primitives' dimensions. After defining the device, he has two options, either to code the device handler into Wonderland client and consume the framework web services in the code or to use a device emulator.

Since the device is novel, Dan creates the event handler for that device without hard coding any events to tasks. He uses the Java web services client proxy generated automatically to consume the framework web service and provide the framework context awareness mapping to Wonderland tasks. The web service uses the input device ID to provide back with the mappings given the current user, and situational context.

Using the interaction interoperability framework Dan has now a fully functional testbed that maps between his device and the afforded tasks of a fully fledged legacy DVE with awareness of the user context in both the real plane and the virtual world of Wonderland. Given the scalability offered by the framework architecture, delays are negligible and do not affect the evaluation of the input device and the new interaction technique.

The next section defines what constitutes a DVE and why having higher user performance is of importance. Section §1.2 discusses the issues of concern in DVEs, challenges with these issues and what the ideal requirements would be. Section §1.3 outlines the approach taken in the dissertation to address the challenges and satisfy different requirements. Finally, Section §1.5 describes the structure of the dissertation.

1.1 Distributed Virtual Environments (DVEs)

This section defines what are DVEs, what motivates the use of DVEs and why this dissertation targets DVEs. As a subset of DVEs, CVEs are, simply put, DVEs designed to support multiple users collaborating together to achieve a certain task.

Elaborating on that simple definition, a VE is a computer application that embodies the user in a the form of an avatar represented in a virtual world mimicking an actual or a fictitious place.

VEs can be single user applications like desktop and console games and they could be multi-user applications supporting a number of users geographically apart to coexist in one virtual place. CVEs leverage these multiuser VEs by affording *collaboration* between the users.

In terms of interaction, this dissertation focuses on three dimensional worlds, DVEs. According to Bowman et al. [Bowman et al., 2005], the general types of interaction tasks include selection, manipulation, navigation, symbol input, and system control. However, there are more specific interaction tasks specific to the applications and the job to be done. Specifically to CVEs, these tasks should be rendered as collaborative tasks in a sense that multiple users can collaborate together to accomplish *tasks, missions or jobs* depending on the nature and objectives of the CVEs.

CVEs can generally help in any case when geographically separated users need to collaborate and do something together. Obviously, the limitation to this is that the job which is to be accomplished should be something that could be shared and manipulated digitally and remotely. Application domains like game playing, educational classes and business operations are great candidates for CVEs given the possibility of playing remotely, holding a class that does not involve physical hands-on and business operations that are based on communication and file handling primarily. The terms *Serious Virtual Worlds* or *Serious VEs* describe VEs built not for entertainment but for a *serious* objective like teaching or business.

By using DVEs and CVEs it is possible to convey the presence of other users through the avatars of the users embodied in one virtual place resulting in higher engagement and higher collaborative performance *in* the virtual world instead of using it just as a communication medium.

An example of such a tightly coupled, distributed task is Constructing a Gazebo in a CVE presented by Roberts [Roberts et al., 2003]. Geographically distributed users meet in a VE and collaborate there on building a gazebo together (e.g., two users could carry a log of wood together). Such an application could not have been possible without having a CVE.

Greenhalgh in [Greenhalgh, 1999] provides the following list of why DVEs are relevant:

- Supporting natural spatial communication (backed by literature that states the significance of *space* in real-world interactions).

- Offering a shared context and thus peripheral awareness for the users.
- Offering visualized information in the VE and communication at the same time.
- Preserving user autonomy in terms of both view and control contrasting 2D collaboration based on *screen sharing*.
- Possibility to scale to a larger user group in contrast to video conferencing that does not offer fidelity levels like in real life (being aware of the whole area and crowd and being able to interact and focus on a subset of interest).

The next section discusses three main aspects of DVEs: scale, interaction and context which are of concern in this dissertation.

1.2 Research Problems: Scale, Interaction and Context

There are a number of issues in terms of designing and building DVEs. The three main issues of concern in this dissertation are scale, interaction and context. This section describes these issues, explains why they are important and why they are crucial to the work done in this dissertation.

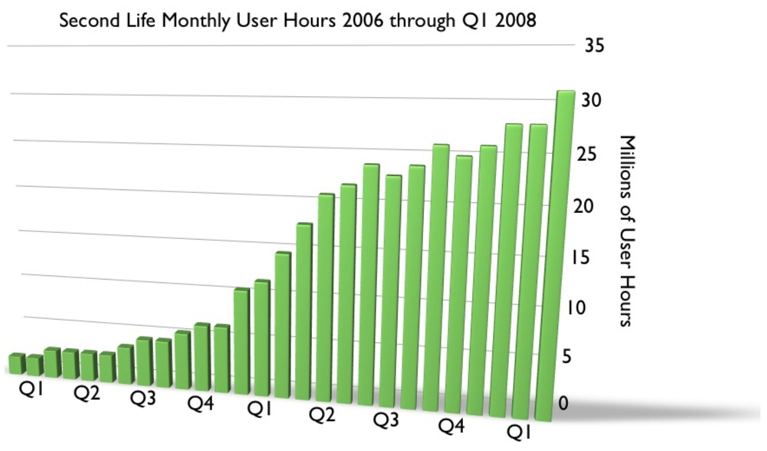
Starting with scale, scale reflects the size of the input working set. *Scalability* is the ability to handle larger working sets with a feasible increase of cost (e.g., computation and memory costs). In other words, for a system to be considered as to*scale well*, it should not have exponential-like increase in time and space for a linear increase of the input (e.g., number of users). Some of the important aspects, as stated in [Greenhalgh, 1999], include the size of the user population, the percentage of simultaneous users and the geographical distance between the users and the grid computers.

Since DVEs/CVEs afford collaboration between *users*, they are expected to support simultaneous users who are *collaborating* together. From the system load perspective, there are two measures. First, there is the population of a DVE regardless of who is *online*. A serious VE used in teaching a school class has to handle the population of all students enrolled in that class. The second measure is what percentage of the population would be simultaneously accessing the system. As stated in Section §1.1, maintaining user autonomy means that a different view should be rendered for each of the users uniquely depending on his location and the direction he is looking at. Supporting information visualization in a VE and at the same time offering communication channels alongside, apparently implies a lot of complex, time consuming computations and as the number of users increase, a DVE should still handle all this processing and message exchanges in supporting all these users [Lin et al., 2007].

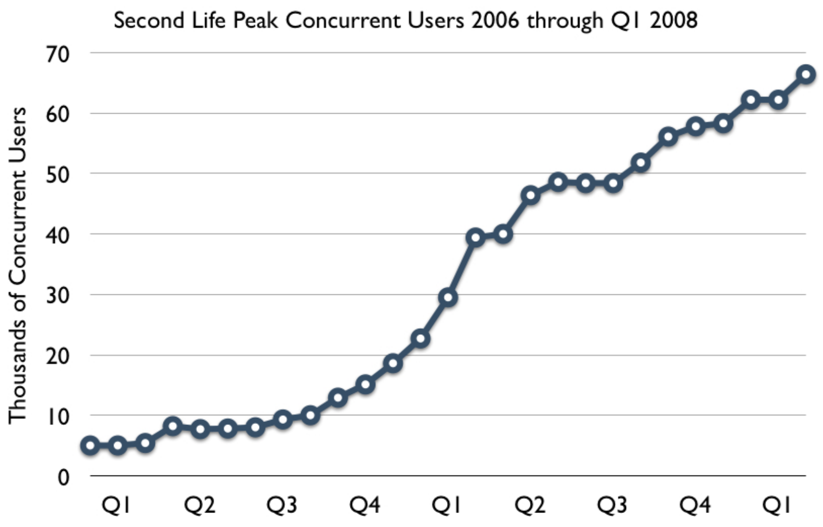
According to Roberts et al. [Roberts et al., 2003], responsiveness is critical. They give an example of delays exhibited in rendering a perspective change following a head movement can lead to disorientation and even feelings of nausea. They also believe that the adoption of tightly coupled collaboration was primarily hindered because of interface and network delays. These findings support how important scalability is for DVEs. Anything proposed to support or extend DVEs should be capable of handling large number of users across varyingly large distances decently as the DVE is expected to be. Given that the work in this dissertation is to extend DVEs, scalability is a highly prioritized design requirement.

In the light of having a large scale expectation, maintaining scalability would raise the following design issues:

- Coping with highly loaded DVEs without compromising usability.
- Communication protocol to use between the framework and the DVEs.
- Message format and structure for platform interoperability.
- Have the services offered discoverable to all the population hosts.
- Push commands to DVEs in a timely manner.



(a) User Hours in Second Life.



(b) Concurrent Users in Second Life.

Figure 1.1: Second Life Growth 2006 through 2008 reflects how DVEs/CVEs are still gaining popularity and given that growth rate it's expected to witness even more growth in population and concurrent users (Courtesy of Second Life's Online Blog http://blogs.secondlife.com/community/features/blog/2008/04/15/) [Sec, 2009] Public Domain.

In terms of the second main issue, interaction, DVEs can use both 2D and 3D user interfaces. In [Roberts et al., 2003] it is stated that in order to increase attraction and productivity of CVEs/DVEs, *interaction offered should be intuitive, reactive and responsive*. However, for majority of DVEs where performance is of importance, the interface is mostly dependent on the traditional input devices, basically the keyboard and mouse. There are a variety of intuitive and creative interfaces developed and it is a great loss not to harness the effectiveness and even enjoyment of these devices and interaction techniques for DVEs.

Interaction includes input devices, interaction techniques, interaction tasks and output devices [Bowman et al., 2005]. The state of the art VEs in general run on desktop computers, gaming consoles and even mobile devices and the whole interaction cycle is hard-coded at the design time. An input device is designated, an interaction technique is coded and the mapping between input and afforded application is put in place. As stated by Benford et al. [Benford et al., 1997], providing a personalized user interface has shown significant improvement in users performance, despite though this given fact, most DVEs offer some minor configurable control assignment like assigning certain controller primitives (like analog sticks and buttons) to application tasks but still personalization should be far more than just simple assignments.

In order to provide Interaction Interoperability the following challenges and design issues have to be addressed:

- Classify and define input devices and interaction tasks.
- Allow manipulations, additions and omissions to the knowledge base while maintaining consistency.
- Untie devices and tasks and loose the hard-coded design time principle.
- Offer on-the-fly mapping between devices and interaction tasks given large scale.
- Profile users based on their interaction choices and current context.
- Manage all knowledge base repositories with no single point of failure.
- Given a user profile, infer to find the most suitable interaction given the available host computer, input devices and running application.
- Profile applications to afford mapping user actions to application tasks.

Another interface personalization approach is through context awareness. This fact is the reason why context is to be considered of utmost importance. If an application is context aware, this means that the interface changes accordingly with a context switch, for example, certain menus show up and others hide. To achieve context awareness its obvious then that the user context should be monitored and assessed against a set of rules that if any has its condition satisfied would fire accordingly making the necessary changes to the interface as stated by the rule. There are a number of issues in building context-aware systems:

- What dimensions to consider?
- How dimensions are going to measured?
- How and in what form context rules are stored?
- How satisfied rules are going to affect the application?

Dealing with context in a DVE extends the problem even further. Being in a virtual world does not exclude the physical environment and the real world. However, the virtual world usually has a totally different context compared to the real world. A user could be in a certain place in the virtual world in proximity to other objects and avatars inhabiting that virtual world at that time. As presented in Chapter 2, there were various research projects trying to discover what the user is *doing* (activity) in a virtual world and what does a certain location in a virtual world represent.

In order to provide support for Context Awareness the following design issues and challenges have to be addressed:

- Assess the different dimensions chosen for the real world, integrate, filter and assimilate collected data.
- Assess the virtual world context given no actual physical sensors.
- Manage context rules and accommodate new rules.

- Know if a rule should no longer hold.
- Assimilate all context data collected from the real and virtual world of the concurrent users.
- Infer through the rules of all concurrent users.

The three main issues, *scalability*, *interaction* and *context awareness* are the focus of this dissertation. Addressing them together require us to consider a number of challenges and trade offs.

- Abstraction of input devices and application tasks while maintaining transparency.
- Profile and model the user, given patterns of interaction with respect to usage context.
- Provide context awareness and offer affordance of interaction interoperability while maintaining the scalability of DVEs.

It should not be necessary to build DVEs from the ground up to make the use of the Interaction Interoperability framework. In other words, the framework should support *legacy* DVEs, regardless of the development toolkit, language and platform used.

This section introduced how important scalability is, why interaction is critical for DVEs and what are the issues in trying to deal with interoperability of interaction and why adding context-awareness would make such a difference for the user experience. In Section §1.3, the approach taken in this dissertation is introduced and how the issues and challenges covered in this section have been addressed.

1.3 Research Methodology

The approach taken to interoperability of interaction is mainly built on the idea of abstraction. DVEs are built based on a technology-centered design. In other words, everything is *hard*-coded at the design time. For example, gaming consoles are shipped with their *dedicated* controllers.

Games and other applications are developed in a number of releases each for a certain platform and that platform is programmed to interpret the dedicated controller input to the afforded tasks. For example, you get a game that runs on XBox and uses the analog stick to control something and that exact game uses the Wii accelerometer to control that same thing in the Wii release. Figure 1.2 illustrates the abstraction concept which I am trying to convey.

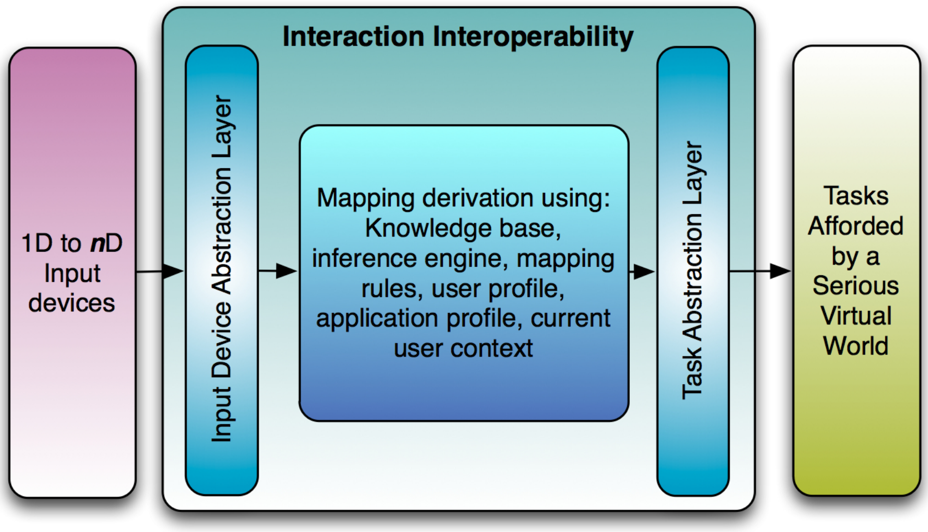

Interaction Interoperability
1D to *n*D Input devices
Input Device Abstraction Layer
Mapping derivation using: Knowledge base, inference engine, mapping rules, user profile, application profile, current user context
Task Abstraction Layer
Tasks Afforded by a Serious Virtual World

Figure 1.2: Interaction Interoperability is based on abstracting the input and output making it possible to define different mappings from a variety of devices and interaction techniques to a variety task possibilities.

What is suggested in this dissertation is to abstract both sides of the interaction cycle; the input on one end and the task to be done on the other. Doing so means that applications (DVEs) should untie inputs from the tasks and expect to get the needed matching matrix through the services of the Interaction Interoperability framework. This abstraction concept is not only applicable for *software* applications but also to hardware and automatic control settings controlling any physical things again using any input device.

Abstraction appears to be the answer to Interaction Interoperability but we would also like provide better personalization rather than just untie devices from tasks. The approach will be to profile the user in trying to build a user model based on the user likes and dislikes.

The approach to user profiling is based on getting feedback from the applications with user selections and decisions. Archiving what the user *liked* using to play or work in a certain application and what the user *disliked* and switched from would be the basis of having a user model in which inference in new cases could be done with a considerably high confidence of what would be favored by the user. The user records provide the knowledge needed to infer the user's preferences. In this dissertation that is referred to as *User Interaction Pattern*.

Extending personalization requires including context sensitivity. The concept is extended to have the interaction patterns reflecting what the user (dis-)liked given the current usage context. Elaborating on the idea, a certain device with a certain application could be favored only in a certain context. In order to include context as a factor in the decision making process and in building the user model, it is necessary to clearly define what *makes up* the user context.

According to the widely accepted definition of context [Dey, 2000], there are four primary dimensions and these are location, activity, time and identity. Secondary dimensions could be derived from those primary ones. The approach taken to assess and monitor context was to have assessment of two contexts for a user at any given time; the usage context of the real world in addition to the context in the virtual world.

The context-aware interaction interoperability framework needs to be scalable and accessible. The approach taken for these two issues is to have all affordances based on distributed discoverable services that implies basing the idea on the service oriented architecture which inherently supports a lot of what the system requirements mandates. The details on using a service oriented architecture (SOA) for the reference implementation of the framework are described in Chapter 6.

The next section builds upon the idea presented and the extensions made to previous work highlighting the key contributions of my dissertation.

1.4 Key Research Contributions

The first research contribution is in effectively abstracting input devices (while taking into consideration major criteria and factors presented in the body of literature) and extending that by using an ontology (*describing* instead of only classifying). Previous work done in in this area and research domain was mainly in trying to find crucial classifiers and the end result was just a taxonomy and/or a devices hierarchy. The contribution of my work in this area is in making it possible to formally describe an input device through a set of attributes and given the ability to do so it makes it possible to build upon that further research experiments and standards development.

Another research contribution is abstraction of the application tasks . The end result of an interaction between a human and a computer is the execution of a certain task. As is the case with input devices, other researchers provide classifications (taxonomies) of application tasks. My approach is to define and formally describe an application task having a full understanding of the task through a comprehensive set of attributes and a palce in the taxonomy of application tasks. Using a formal definition of tasks it should be possible to re-target the interactions for different platforms and integrate applications at the task level.

From more practical perspective, the contribution of my work is also in facilitating testing out new input devices and interaction techniques against legacy systems, i.e.providing a scalable testbed to support interaction techniques and reality based interactions testing. My work can make it possible for users to have the freedom to choose whatever interface they like to use in a DVE. With the framework components I have designed, its possible to transparently utilize all these functionalities with an added advantage of context sensitivity as well. Given the framework design, services offered by the framework are discoverable and distributed. The framework is based on open standards with the objective of offering the highest system interoperability possible as well as the scalability needed.

Finally, there is one other contribution that should be of an impact on research and industry and that it is in bridging between real worlds and virtual worlds both in terms of context and actions. Context-awareness in previous research was mainly based on the user's real world context disregarding the importance of contextual dimensions in the virtual world. Research can be extended given that wider awareness of context and practically it is possible to extend personalization to virtual worlds and maintain consistency between *mirrored* VEs, specially serious ones.

1.5 Dissertation Structure

This section describes and explains the structure of this dissertation and highlights what is being discussed in each chapter, as follows:

- Chapter 2 Related Work.
- Chapter 3 Interaction Interoperability in DVEs.
- Chapter 4 Context Awareness and Context Management.
- Chapter 5 Data, Information and Knowledge
- Chapter 6 Interaction Interoperability Framework.
- Chapter 7 Reference Implementations.
- Chapter 8 Evaluation.
- Chapter 9 Conclusion.

In Chapter 2 background and related work in the field are highlighted. A brief description of projects related to the different areas of DVEs, Interaction, Context Awareness and Personalization are discussed highlighting the points of contribution, points of contrast or in relation to the work of this dissertation. The sections that make up the chapter are divided by areas of research which are DVEs/CVEs, Interaction, Context Awareness and Personalization.

Chapter 3 lays the ground for the concept presented by the dissertation which is Interaction Interoperability. The concept is presented, explained and discussed at the highest level of abstraction.

The sections of Chapter 3 are Motivations, Interaction Interoperability Model and the concept development. The Motivations section highlights the importance of the concept and how does it fit in the context of DVEs, the next section illustrates the general model of the concept and the last section provides a foundation how the realization of the concept can proceed.

In Chapter 4, context awareness and management are discussed, challenges presented and how they are handled and the approach taken to assess, monitor and adapt to context. There is a separate section that focuses on the idea of having dynamic rules, how these are generated and under what circumstances.

Chapter 5 describes the knowledge base that the framework builds on (including data, information and knowledge) and the approach taken in defining and using the knowledge base.

Chapter 6 presents the Interaction Interoperability Framework and the related concepts, challenges and approaches. It explains what would be the suitable components, interfaces and layers needed to fulfill the design requirements in place. Using the framework makes it possible to implement a complete solution to offer context sensitive interaction interoperability given all modules, components and interfaces designed.

Given the interaction interoperability framework, Chapter 7 describes the different reference implementations built as a proof of concept as well as for evaluating the design against the hypothesis and goals put forward.

Chapter 8 evaluates the framework and describes a complete implemented case study. From the user perspective the framework should provide affordances for a better user experience. From the system perspective the framework should meet the design requirements to cope with highly loaded DVEs.

Chapter 9 provides an overall discussion of the dissertation, summarizes the conducted research, highlights the contributions and provides suggestions for future work.

Chapter 2

Related Work

In this chapter, a literature review in the areas of Interaction, Context Awareness, Service Centric Architecture and User Modeling is presented. Since the research work presented is interdisciplinary (interactions and interfaces, distributed systems, highly scalable virtual environments) the review presented focuses on significant work done in each of the fields and provides important facts, experimental setups, designs and findings are gathered and concluded.

The research methodology was primarily based on extending and building on the advantages and avoiding the disadvantages of previous work compiled in this chapter.

2.1 Interaction

For the area dealing with adapting interfaces to input devices, there is a misconception found between the terms interaction “retargeting” and “interoperability”. Most of the work done was in trying to build toolkits that could be used to build applications that support different input devices. Figueroa et al. [Figueroa et al., 2002] proposed InTml (Interaction Techniques Markup Language) which was built to ease the **development** of VE applications with a platform independent set of reusable components.

In [Dachselt and Rukzio, 2003], Dachselt et al. developed an extension specifically for the eXtensible 3D (X3D) standard but again to facilitate **building** applications that would take advantage of an extended set of behaviors and sensors (event listeners) that were not supported by the basic standard.

What is meant by the term interaction interoperability is simply how to adapt different interaction techniques to interaction tasks. In [Irawati et al., 2005], Irawatia, Calderna, and Ko presented a framework that handles multi-modal manipulation for 3D Interactions using object ontology to solve ambiguity of object manipulation. The ontology of the *object* stores information about constraints on how an object may interact with other objects. It also describes the spatial characteristics of the object related to other things in the environment.

Taking into consideration the current user context and the object ontology, it combines several modes of interaction to create an integrated interpretation to make object manipulation in 3D VEs more intuitive. The main component of concern presented in [Irawati et al., 2005] is the semantic integration component. The main role of this component is integrating input from various modalities into a single interpretation. Given any number of inputs (including speech) they are mapped to an action command, selected object, new target location and the movement direction. Getting that, the interaction manager —another component— finds the proper translation and rotation to fulfill the user command and then shows the result in 3D by translating or rotating the selected object.

A major work of interest in the field of interaction interoperability is by Masso et al. [Massó et al., 2005]. Their work was to have an abstracted user interface based on what is called *UsiXML*, an XML-compliant User Interface Description Language [Usi, 2008]. UsiXML provides a Concrete User Interface description that remains independent from any toolkit, whether graphical or virtual. What Masso et al. state is that they used on UsiXML because it does not only support the description of a UI, but also the method that can be followed to produce such a UI. Based on this it is possible then to simulate in the virtual world the look and feel of the same UI in virtual versions of different platforms, such as PDAs, PCs or even Interactive Walls. In addition, it supports migrating a UI from one platform to another (e.g., 2D to 3D) by visually representing this transition.

Another piece of work that appeared later again in Masso et al. [Massó et al., 2006] was focus on user interface migration. They have defined the migration of a UI as the action of transferring a UI from one source location to a target one. Given that a location could be any computing platform, an interaction surface or an interaction space. In [Massó et al., 2006] they have cited a number of publication that classifies the types of migration. Assuming the UI to be decomposed into two components, the control which is responsible for the UI behavior and the presentation which is responsible for presenting information to the user, control migration migrates only the control component while the presentation remains. In presentation migration, it is just the opposite: the presentation component is migrated while the control remains on the source platform. Mixed migration however, different parts of both the control and the presentation are migrated.

What Masso et al. also state is that there should be a special UI to perform the steps needed for the migration. Some of these steps are identification of migration possibility, proposal for migration, selection of migration alternative, and execution of the migration itself. Given the high complexity of handling UI events and procedures, the UI responsible for migration is even more complex and not always visible to the end user. This special UI is referred to as the meta-user interface which is the UI for controlling the run-time migration of the UI of the interactive systems.

Meta-UI is implemented in a number of different ways. It could be system initiated by which the system initiates the migration, user-initiated by which the user initiates the migration, or mixed-initiated, and in this case the user and the system collaborate to perform the migration. What was strictly done in [Massó et al., 2006] is that they have developed a meta-UI as a VE for controlling run-time UI migration with all the steps of the migration graphically represented in virtual reality mimicking the real world. Below is a screen shot of the system showing the VE used as a meta-UI. Another mapping proposed based on UsiXML was by Limbourg et al. in [Limbourg and Vanderdonckt, 2004]. Their addition was to give the developer more control to assure how will the mapping be done to the final user interface (FUI).

Others that dealt with device-independence and multi-modality but in the same way as

UsiXML were Simon et al. [Simon et al., 2005]. They were motivated as they state by single authoring for device- and modality-independence. They developed a framework for developers to use, mitigating all retargeting problems in the same manner as we can say like [Massó et al., 2005].

Again concerned about retargeting, focused on mobility, Repo in [Repo, 2004] proposes an architecture that makes it possible to develop adaptable user interfaces in a context-aware pervasive networking environment. They make it viable to generate mobile-capable user interfaces using HTML, WML or Java from Web-based user interfaces (without the need to re-code applications).

Regarding the web, specifically X3D, Stewart et al. [Stewart et al., 2007] proposed external interactivity to X3D making it possible to control a VE using MIDI controllers mainly built for audio professionals.

Another approach to handle adaptation not interoperability specifically but in a way similar to ours was the work of Celentano et al. [Celentano et al., 2004]. Their work was focused on trying to anticipate the user behaviors by monitoring the way they interact with objects. Identification of recurring patterns of interaction was used then to shorten users' interactions when they approach objects belonging to the same class previously accessed.

A system that tries to tackle two different problems, device-independence and network-transparency was developed by Taylor et al. [Russell M. Taylor et al., 2001]. The system was called VRPN. The main essence of the work is that the device, the VE and the user each could be on a different machine. VRPN factors devices based on their functions and maps them to connections. VRPN main focus is on the networking part. The main accomplishment is that the device should not be directly connected to the host running the VE but it could be just accessible via a computer network and control is done by message exchange.

The following section discusses some related work in the area of designing and developing interfaces and interactions for DVEs/CVEs. Highlights are on the main factors and influences taken into consideration in designing CVE interfaces.

2.1.1 Interaction in Virtual Environments

As defined in [Benford et al., 1997] a CVE is a typical VE using virtual reality technologies to visualize a space were multiple users meet to collaborate despite the fact that these users are geographically apart in the real world.

The main initiative as highlighted by Benford et al. in [Benford et al., 1997] is that CVEs would provide more extensive and successful cooperation than other collaborative technologies since they support a greater degree of social interaction. The main focus of their work was to focus on practicality in designing CVE and as they claim this is not possible unless such environments are specifically designed for a certain domain and a specific application rather than being generic. Another point that should be noted is that Benford's work was published around ten years ago and they boldly stated that at that time supporting technologies are much more enabling to design successful CVE so this would be even better nowadays than before.

There are many factors that affect the process of collaboration in the real world. These factors vary from being clear and understandable to subtle like postures and face expressions. Interactions in CVEs have got a great deal of variability ranging from simple text chatting to 3D interactions in a fully immersible, audio capable, complex 3D graphical environment.

As stated in [Bullock and Fahlén, 2000], richness of interaction in a CVE does not necessarily depend on the fidelity and complexity of the interface to the CVE. It is stated that it is possible to have a simple text-based CVE that offers a far superior collaboration experience to a sophisticated, graphical CVE that supports embodiments, gestures and audio.

A number of questions have been posed by Bullock and Fahlen in [Bullock and Fahlén, 2000] trying to opt for better interaction techniques to be used in CVEs. One of the questions was, what are the needs and demands of newly emerging class of inhabitants that are totally different from researchers accustomed to a number of VE interactions. Another question, how can CVEs be made habitable when presented using different forms of interaction and presentation technologies. What is implied here is that CVEs could now be deployed on various devices ranging from static workstations to mobile PDAs and with interaction techniques ranging from symbols entry to sophisticated interfaces including trackers and voice driven interfaces.

Another major point questioned by Bullock and Fahlen [Bullock and Fahlén, 2000] is the differing capabilities for users in the same CVE. Their questions are finalized then with what new interface metaphors and devices can be explored in order to achieve a much more successful collaborative environment. Summing it up they stated that it is needed to understand how and which interactions lead to practical and successful collaboration.

In [Xu et al., 2006], Xu et al. presented a challenging CVE for product assembly design supporting designers from different places with different platforms to engage in the same assembly scenario and complete the assembly task synchronously. In [Xu et al., 2006] they introduced a web-based CVE for real time assembly design, focusing on how feasible and practical that such a CVE could be.

Ray [Ray, 2008] presented what he called as the Interaction Framework For Innovation (IFFI). The framework presented had the ability to create reusable three-dimensional interaction techniques. The main essence of the IFFI is in offering reusable interaction techniques through what Ray calls "Filters". The idea of building an interaction technique is then achieved by interconnecting a set of filters to represent that technique.

IFFI does not abstract input devices per se, it actually expects an abstract view of the devices through the lower layer which was a VR toolkit. The filters also constituted associations to a set of tasks that would make up an interaction technique. From the design of the IFFI and the clear illustration of its location within the VE architectural layers it would be possible to incorporate the framework presented in this dissertation to further abstract both sides of the IFFI facilitating different architectural layering as well as the other functionalities offered like context awareness and system personalization.

Considering the three main components of an interaction which are device, technique and task, Ray's framework would represent the abstraction and usability of the middle part and the framework presented here can complement that dealing with the input and execution of tasks.

2.1.2 Input Abstraction

The interest in input devices taxonomies started since in early eighties The initial work was done by Foley et al. 1981 and later published as a technical report [Foley et al., 1984]. The main essence of the work was to classify input devices based on their physical structure.

Following that, in 1983, Buxton [Buxton, 1983] added a crucial modification that he calls the lexical and pragmatic considerations. Buxton's considerations mainly was to indicate what the device is used to do rather than how the device is built. Devices were classified under position, motion or force. Another classifier was the number of dimensions, one, two or three. Extending this, to differentiate between a light pen and a touch screen — being both 2D, position sensing devices — Buxton had subclasses for directly hand manipulated and indirect mechanically manipulated devices.

Following Buxton's taxonomy, Card et al. [Card et al., 1990] and Bleser and Sibert [Bleser and Sibert, 1990], each proposed an extended taxonomy. In [Card et al., 1990], Card et al. placed a *design space* for input devices.
They represent an input device as a six-tuple, (M, In, S, R, Out, W) where:

M is the manipulation operator.

In is the input domain.

S is the current state of the device.

R is the resolution function mapping from the input domain to the output domain set.

Out is the output domain set.

W are general properties.

An enhancement they made was that, having the output domain specified, it is possible to represent a discrete device like a switch, in contrast to Buxton who was only focused on continuous devices.

On the other hand, Bleser and Sibert [Bleser and Sibert, 1990] developed an artificial intelligence application built in Smalltalk representing the input device taxonomy in a form of a class hierarchy defining devices by their input domain, data output, the physical actions that can be performed on them, and their physical packages.

In 1992, Jacob and Sibert [Jacob and Sibert, 1992] claimed then that the perceptual structure of the devices has been neglected. They proposed another classifier to be added which discriminating integral devices from separable ones. The meaning behind this is that with an integral device, movement is in Euclidean space, cutting across all the dimensions of control, while a separable device constrains movement to a stair-step pattern; movement occurs along one dimension at a time.

Following that contribution, Lipscomb and Pique [Lipscomb and Pique, 1993] created two graph structures, one for tabletop devices and another for elevated devices which are devices held by or mounted on the user. The graph nodes show device characteristics and the edges show how characteristics are related. The limitation of this work is that they were discussing only analog devices.

Wang and Mankoff [Wang and Mankoff, 2002] and Carter et al. [Carter et al., 2006] analyzed the bandwidth of the input device. They elaborated their viewpoint with an input emulation layer that tried to map between a two-key keyboard (low-bandwidth) and the QWERTY keyboard (high-bandwidth).

Another example was how to map from a mouse (high=bandwidth) to QWERTY keyboard (low-bandwidth) which is simply drawing a virtual keyboard on the screen. However, the limitation of their work was that they were focused on 2D WIMP interfaces with no regard to the more complicated 3D devices.

There are a number of development kits also that offer input device abstract but these offerings are based on developing the VE using these toolkits in contrast to incorporating functionality in any toolkit or platform. A representative example of such an open source project is DirectFB [dir, 2010]. DirectFB is an open source thin library which is built to provide hardware graphics acceleration, input device handling and integrated windowing. FB stands for FrameBuffer Device and it offers complete hardware abstraction. It is evident that solutions like DirectFB present an alternative way of *developing* applications that are built on top of that virtual layer. Using such approach could be feasible for certain applications or situations but it offers no flexibility in choosing a development language or toolkit nor specific abstraction in contrast to complete virtualization.

Other major contribution in the area of toolkits used in developing DVEs is DIVERSE [Kelso et al., 2006, Div, 2010]. The idea behind DIVERSE was to build an open source virtual reality toolkit which can be used in building *D*evice *I*ndependent *V*irtual *E*nvironment - *R*econfigurable, *S*calable and *E*xtensible, hence the name. Using DIVERSE developers can build VE applications that can run on DIVERSE supported platforms. The toolkit supports a set of *predefined* input devices and offers an API to access them. The major contribution of the toolkit is in *building* cross platform VE built using the toolkit API and leveraging network connectivity and scalability through dynamic load distribution trying to eliminat high CPU load which significantly decreases performance of DVE systems. The main impact of DIVERSE is in transparently utilizing a distributed scalable architecture, based on the idea of abstracting virtual reality hardware in order to facilitate porting of applications over different platforms.

2.2 Context Awareness

There were a number of different definitions derived for Context Awareness, mostly they were around the definition of Panayiotou [Panayiotou, 2000] which was *A set of premises expressed in some language, gathered intentionally or unintentionally in a relevant, coherent manner and which can itself constitute and adequate set of inferences (meaningful) or lead to some meaningful results (inferences)*."

The most widely used definition though was coined by Dey [Dey, 2000] as "*Any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*."

Ferscha et al. [Ferscha et al., 2004] stated that with users interacting with their applications from anywhere and at any time, different usage situations can happen.

Answering then what is context-awareness for, Anusuriyain in [Devaraju et al., 2007] stated that context awareness enables a new kind of applications that extend the functionality of the humans subconscious, acting at the right time, based on the right level of information.

Anil et al. [Shankar et al., 2007] presented results from an empirical user-study which investigates if information from the user's environment helps a user interface to personalize itself to individual users to better meet usability goals and improve user experience and their findings. That emphasizes the need for user-context for personalized user interfaces which can better meet effectiveness and utility usability goals as well as an overall improved user-experience.

Concerning the factors involved with context-awareness, it was highlighted by Chow [Chow, 2006] that the research issues on context-awareness includes modeling the context, having context specification, acquisition of the user context, interpretation, and context management. These factors were restated then by Santos et al. [Bonino da Silva Santos et al., 2007] and they added that these factors would complicate offering ad-hoc context-awareness solutions.

The norm of contextual information gathered according to [Dey, 2000] has four primary types referred to as context dimensions. These four dimensions are location, identity, time and activity. If more contextual information sources could be derived from the four dimensions then these sources would be considered as secondary dimensions.

In order to define what makes a system to be context-aware, Dey [Dey, 2000] states that the system uses context to provide relevant information or relevant services to the user.

For feature classification, context-aware features would include presentation of information and services to a user, automatic execution of a service and tagging of context to information for later retrieval. In [Ferscha et al., 2004], context computing has two major issues, first is identifying relevant context (identity, location, time, activity) and the second is using that obtained context (automatic execution, presentation, tagging)

Jones et al. [Jones et al., 2004b] described two studies that investigated a number of factors affecting the deployment of context-aware systems. Their findings were that place-type as an isolated piece of information does not determine information needs. The user routines and social relationships must be an integral part of this. While people were willing to share their location information, they demanded control to specify particular individuals and groups who may or may not access this information and filter information based of certain places based on these groups.

Stressing specifically interoperability in context management systems, Hesselman et al. [Hesselman et al., 2007] presented a context management system by that serves as an interoperable context provider for applications on different platforms. The applications are "interested" in being context sensitive and rely on a context middle-ware that would bridge between context elicitation details and higher level abstraction of context provided at a higher, platform independent level.

2.3 Service Centric Architectures

In the work of Brien et al. [O'Brien et al., 2007] the quality attributes of service oriented architectures have been discussed. They have started first by defining what is a service oriented architecture.

According to their definition, it is an architectural approach for building systems where there are components that are either service users or service providers. A service is seen to be a distributed component with the following characteristics: self-contained; has a published interface that abstracts the underlying logic; location transparent; can be implemented in different languages or platforms and still interoperate; is discoverable and dynamically bound.

From Brien et al. point of view, choosing and designing an architecture that satisfies the functional as well as the quality attribute requirements they suggest; reliability, security, and performance is vital to the success of the system. A way to guarantee the level of quality is to go for service level agreements (SLAs) signed between the service provider and the users.

Defining the main reason which service centric architectures are built for which is interoperability, Brien et al. state that it refers to the ability of a collection of communicating entities to share specific information and operate on it according to an agreed-upon operational semantics. The main part of their work is that cross-vendor and cross-platform interoperability suffers when services start to use features beyond the two basic standards: Web Service Definition Language (WSDL) and Simple Object Access Protocol (SOAP). Over the last several years, many Web services standards have emerged from a number of standards bodies. Web services development platforms do not implement the same versions of all standards, so interoperability may not be as seamless in practice as it is in theory. This raises the issue of Semantic interoperability where further research is needed to achieve seamless interoperability between systems.

2.4 User Modeling and System Personalization

Zhang et al. [Zhang et al., 2006] have proposed an architecture which links cross-system personalization and users modeling. They note that despite a number of applications and web sites that use personalization, the user profile is private and exclusive for an individual application and not shared.

Another problem also is that the user can not control all personalization aspects since these aspects are may be controlled and hidden in personalization engines. Zhang et al. started by categorizing the approaches that involve cross-system personalization into four categories, Generic User Modeling, Multi-Agent based, Ontology based and Unified User Context Model.

Starting with generic user modeling systems (GUMS), information about the user is stored and maintained in a central repository. User information acquired by one application can be employed by other applications, and vice versa.

The Multi-Agent based approach is built upon the idea of having a decentralized user model employed and dispersed in various autonomous agents as fragments. Based on these user model fragments, various goals of the user can be achieved in variety of specific contexts.

For the Ontology based approach a service-based architecture was used in a project named

ELENA. The learner model was based on two of the most important standards for learner modeling, PAPI (Public and private information) and IMS LIPS (learner information package specification). For sharing learner model in the learning network, a domain and learner ontology were introduced. Heckmann et al. [Heckmann et al., 2005] introduced the GUMO (General User Model Ontology) for the uniform interpretation of distributed user models in intelligent semantic Web enriched environments. They developed a user model exchange language UserML to enable decentralized systems to communicate between user models. To deal with the challenge of reuse user information in ubiquitous computing environment, they suggest representing user profile as statement. The GUMO based approach is efficient in decentralized and mobile services where users only require some simple services.

The Unified User Context Model approach is based on the idea of cross system personalization and a multi-dimensional Unified User Context Model (UUCM). The required user information was integrated by the mediator named as context passport. Reusing the user model is done by exploiting the same user profile in several systems. The communication between Context Passport and service provider was achieved by a Cross-system Communication Protocol (CSCP). The UUCM Based Approach introduces a method to support personalization across different systems. However, the system requires every participant exploit UUCM model and support CSCP protocol.

Recalling what Zhang et al. recommend, building a system that can glue whatever is available rather than start from scratch, is based on employing web services. The web services store and communicates a user model based on the role they act and by this effectively support the various existing personalization systems and their user models.

The ontology based User Role Model (URM) was used for modeling users and their related roles according to the service they accessed. It classifies the users by the role they usually act in different service and mapping it to the role they act in current task (taskrole). Every role was described by the five dimensions: Geninfo, Preference, Relationship, Task and Taskrole.

Geninfo Dimension General personal information of the user. It contains the basic information required by websites for information accessing, e.g. user id, location, email, phone number, etc.

Preference Dimension Facets about the habit, competence and general interest of user.

Relationship dimension Describes the social relationship the user involved in

Task dimension Describes task related information about the user. It contains goal, history task and current task.

Taskrole dimension Define the concrete role of users in a specific task.

The user ontology can be extended according to various character of the service provider participant in the system and the information required retrieving personalization service.

2.5 Network and Content Interoperability

Given that the focus of my research is about interaction interoperability with platform independence, it is important to address network and content interoperability. Network interoperability in general reflects how heterogeneous, loosely coupled systems could be able to communicate and work in harmony together. On the other hand, content interoperability reflects the idea of having media and other content in general communicated and shared among different sites, applications and platform.

Section §2.5.1 goes through some of the significant work and standards in the area of network interoperability while Section §2.5.2 focuses on content interoperability.

2.5.1 Network Interoperability

Wireless-enabled mobile devices are becoming very popular. Almost all new cellular phones can communicate through WiFi networks and Bluetooth. The same applies to PDA devices the mobile gaming consoles. The two most widely adopted architectures are either a client-server or peer-to-peer (forming an ad-hoc network). In [Luiz Lima and Calsavara, 2007] this fact is stressed and it is stated that difficulties that these platforms have to cope with are numerous and range from heterogeneity and interoperability to mobility issues. Lima and Calsavara in [Luiz Lima and Calsavara, 2007] propose adaptations and extensions to the CORBA's interoperability model so that it can be used to transparently handle message exchanges among objects across ad hoc networks.

A phenomenon that is noted and discussed by Merabti [Merabti, 2006] is the existence of broadband Internet as an integral part of our household infrastructure. Every device we own now including TVs, gaming consoles, Audio and Video equipments have a network interface that allows it to be accessed and controlled globally. This resulted in the evolution of a number of things supported by this including on-demand multimedia services, home automation with the help of wireless sensor networks, remotely controlled home appliances through immersive technologies and home health care monitoring. In [Merabti, 2006] Merabti proposed a framework on self-adaptive networked appliances and illustrate how services can be described semantically, and automatically composed using high level descriptions related to what rather than the how.

A number of other projects dealing with the network interoperability problem have been reviewed in [Merabti, 2006]. Most of the projects were mainly directed to the domain of multimedia in how to distribute, compress, share and control. Some of the projects reviewed were directly tied to the networking problem even at the embedded systems level like the project RUNES, and others introduced the use of Mobile IP and IPv6 in the area of home networking. The most relevant of these projects is the ePerSpace project. ePerSpace framework addresses interoperability problems and the management of service platforms including service and context adaptation using personalized data, but the main goal of ePerSpace is to create a trusted and interoperable integrated framework to seamlessly interconnect heterogeneous audio and visual devices.

2.5.2 Content interoperability

From the perspective of DVEs, the widely used content interoperability standard used to create 3D interfaces and model virtual reality environments on the web was VRML [vrm, 2010]. VRML stands for **“Virtual Reality Modeling Language”** and it was the language that was used by 3D artists to create virtual reality scenes that can be accessed through the Internet using a VRML browser. The last standard released by the ISO was VRML97 and a number of applications and systems were built using VRML to serve different domains ranging from serious educational and training applications up to games and entertainment applications.

VRML97 was replaced by it successor, X3D. The name X3D stands for “eXtensible 3D” and this standard was proposed and is maintained by the *Web3D Consortium* [Web, 2010] which is a member-funded industry consortium committed to the creation and deployment of open, royalty-free standards that enable the communication of real-time 3D across applications, networks, and XML web services. As defined by Web3D, “X3D is a royalty-free open standards file format and run-time architecture to represent and communicate 3D scenes and objects using XML”. In other words, X3D is an open standard for 3D content delivery that contains not just geometrical description but also the behavioral description of the scene.

The main advantage in VRML was that it was built with performance being the ultimate goal. VRML content was so compact and so this offered a fast parsing speed as well as that it was context free. However, although VRML did the job right there were some additional needs/requirements.

Nowadays most applications need to communicate with other systems and distributed applications will move to the mainstream as developers produce applications that consume web services provided by third parties, and build web services for others to use. In order to fulfill this need, we should go for an XML based standard.

The second motivation is the need to incorporate new graphics technologies in a standardized way. There was need for a standard that makes use of the supporting graphics and other hardware available.

The third motivation was to have a 3D description file that appears the same across different platforms and browsers. The problem with VRML, as stated in literature, is that since there is lack of adequate specification of behavior in the VRML standard.

The last motivation for X3D was that the evolution of the standard was too slow with last ISO released standard was by 1997. However, the evolving field of developing 3D user interfaces for the web do deserve a much more competing standard that evolves quickly with time.

2.6 Summary

In this chapter, I presented a literature review of prior work in interaction, context-awareness, service centric architectures and interoperability (network and content). My work builds on the prior research in trying to extend the approaches presented to satisfy the objective of interaction interoperability. The contributions of this dissertation are in achieving the design goals of abstracting interaction, utilizing context in both virtual and real planes and offering a modular service centric based architecture that would scale well for thousands of concurrent users typical for DVEs.

Chapter 3

Interaction Interoperability in DVEs

Context Sensitive Interaction Interoperability is to be defined as the ability to map between any input device suitable to any application task afforded given the current context of the user in both the real and virtual worlds. The objective is also to afford such interoperability as a discoverable distributed service that can be used by applications aware of the service as well as legacy ones.

This chapter describes the concept of Interaction Interoperability proposed in this dissertation. Section §3.1 describes the motivations behind the idea of going for interaction interoperability. Section §3.2 presents the interaction interoperability model. Finally, Section §3.3 illustrates what points should be focused on concerning the design requirements for interaction interoperability.

3.1 Motivations

There are several motivations behind going for a middleware that would support interaction interoperability. There is a problem in developing applications in general and VE applications in particular. User interaction components of VE applications are sometimes poorly designed and evaluated.

The majority of VE research and design effort has gone into the development of visual quality and performance. As a result, there are many visually compelling VEs that are difficult to use and “unproductive”.

Given that these VEs can be good entertainment applications, their usability problems prevent them from being useful for efficiently solving real-world problems.

As critically stated by Gabbard et al. in [Gabbard et al., 1999] “We have reached the point in VE development when we should shift from largely open-ended explorations of new technologies to more scientific studies of the benefits and impact of VEs on their users”. So, it would be desirable to deal with applications as just affording tasks to be accomplished and there would be a mapping layer to map from a vast amount of techniques to the application interaction tasks, hopefully making it possible for users to pick their better choice.

The main motivations behind the work of this dissertation in offering context aware interaction interoperability, are:

- Abstract input devices making it possible to interface applications to any device.
- Abstract application tasks eliminating hard coding input to task mapping.
- Make it possible for a user to use any input device in a preferred way to interact within a DVE.
- Make it possible for every user in a DVE to use his/her own input device in a preferred way.
- Making it possible to automatically adapt the interface not only the layout but also input device and interaction technique to match the current context of the user.

Having described some of the motivations which have informed context sensitive interaction interoperability, the next section describes the interaction interoperability model that would fulfill the goals set.

3.2 Interaction Interoperability Model

A user interaction starts with an input device (like a keyboard or a game controller) and ends with the task that is to be accomplished (like entering text or navigating in a virtual world).

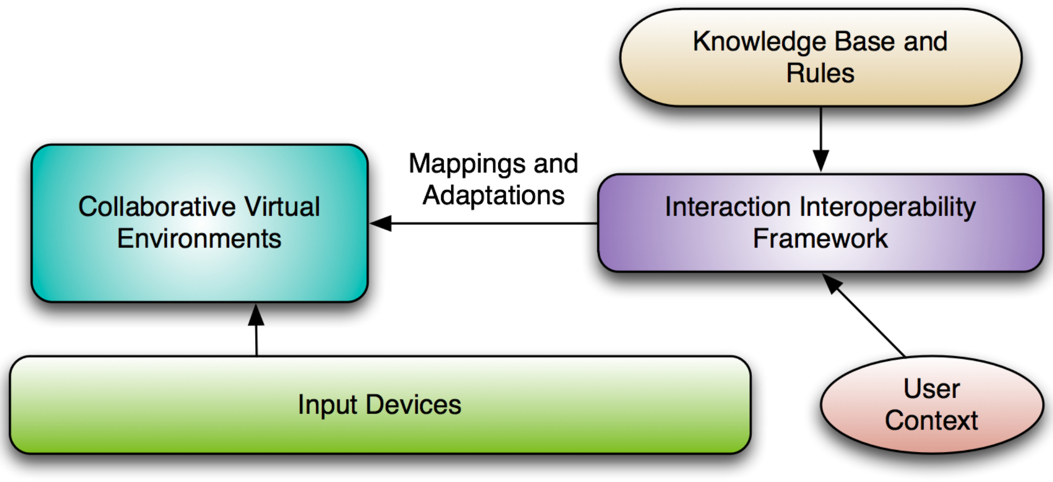


Figure 3.1: Interaction Interoperability Model

Figure 3.1 illustrates the high level model of interaction interoperability. An interaction interoperability frameworkcan be used to support a DVE. Supplied with a knowledge base of devices, techniques and applications, in addition to context rules that are fired according to the user context elicited, the necessary adaptations are sent over to the DVE. In that way the DVE can take advantage of context sensitive interaction interoperability *transparently*.

In order to achieve interaction interoperability, our approach is based on abstracting the input and output making it possible to define different mappings from a variety of devices to possible tasks afforded by the application.

Figure 1.2 illustrates how interaction interoperability can be provided. Any input device should be abstracted by breaking it down into a number of controllable dimensions, define if these dimensions are integral (change together) or separable [Jacob and Sibert, 1992] and have a range for each of the dimensions. However, in order to be able to have dynamic, on-the-fly mapping, devices can not be just segregated, they must be explicitly and individually formally *defined*. Having an ontology of input devices (will be discussed in Chapter § 5) was the optimal way to realize this.

In terms of tasks performed, abstraction is achieved by having for every application a defined set of tasks it affords. Again, tasks afforded are defined using another ontology that has a thorough definition of the different classes of tasks like navigation, control and manipulation. For each task there are possible dimensions of movements and for each dimension there is a defined range. As an example for the screen pointer movement there are two dimensions, with ranges of values going from $(0,0)$ to $(Max-X, Max-Y)$. A task like opening a door in a virtual world would have only one dimension and a range of values between 0 degrees and 90 (or possibly 180) degrees.

In terms of scale, the results in [de Freitas, 2009] indicate that the growth of online serious games and serious virtual worlds offer learning beyond the formal learning context as well as support distance and distributed learning. This motivates our work to support such a distributed nature of likely large scale interactions that would need a scalable, distributed framework.

When it comes then to context awareness, the activities in *virtual worlds* may be affected by the real-world situation. Simply stated, there are two planes, real and virtual, with two different contexts. There is the real life the user is living in and there is the virtual world. As an example, for the location context dimension, there is the *physical* location of the user and his location *in the virtual world*. The user could be at home physically but he could be in a meeting room in the virtual world. The same is for other dimensions, e.g. time were the current time of the physical location and the virtual time of the virtual world are available.

With this level of complexity, especially when dealing with secondary context dimensions derived from the primary ones, an interaction interoperability framework should employ a complete layer of context management that should acquire, analyze and filter the context of the user based in both real life and virtual presence.

A use-case diagram of Interaction Interoperability is shown in Figure 3.2. The use-case illustrates the system boundary, the services offered, and the different parties involved.

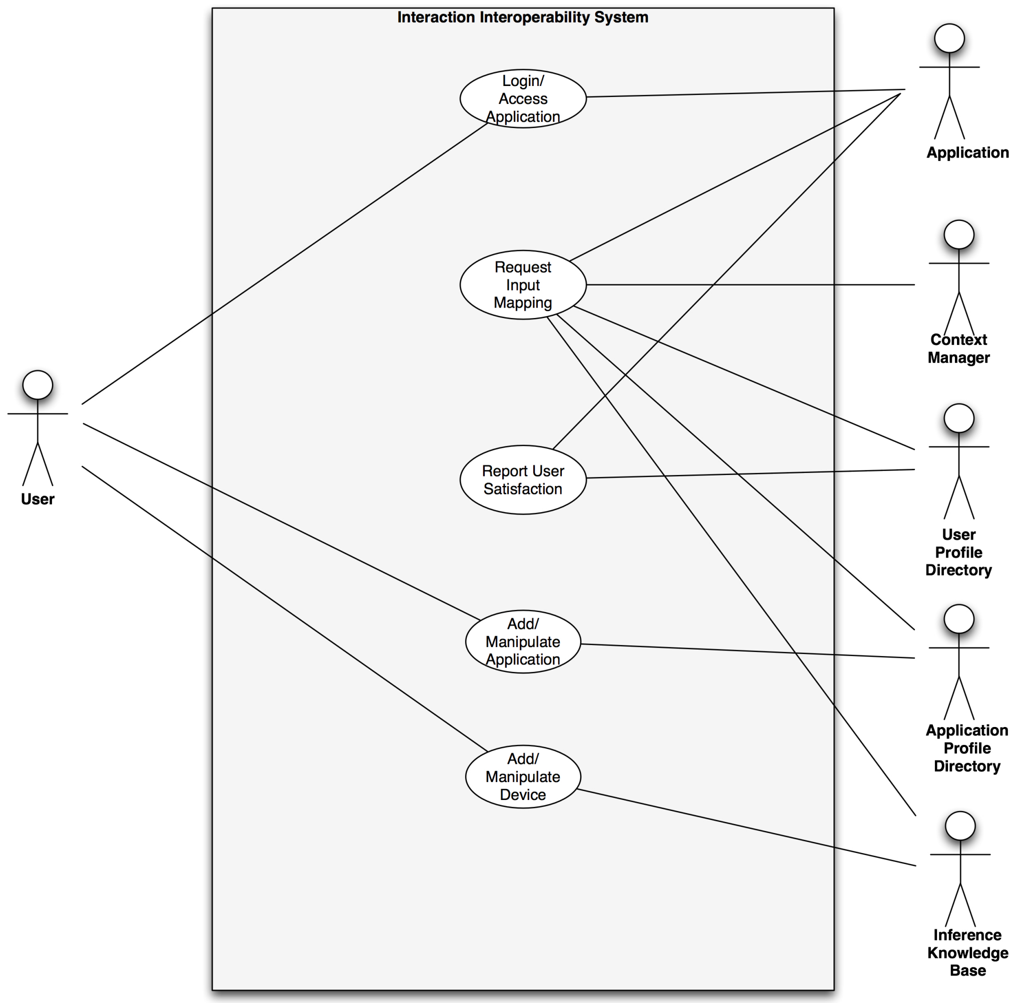


Figure 3.2: Interaction Interoperability use-case diagram.

Chapter 4 describes the context management layer and analyzes each of the context dimensions. I describe approaches on how to acquire an accurate description of the user context in order to reach useful decisions to support personalization.

3.3 Interaction Interoperability Design Requirements

This section summarizes the design requirements as informed by the model of interaction interoperability discussed in the previous section.

Given the motivations and the model described, the design requirements developing a solution would be as follows:

- Capability to define input devices including novel devices.
- Capability to profile and define application tasks.
- Support for legacy DVEs without the ned for DVE modifications.
- Profile and model user preferences.
- Capability to create new mappings based on user profile and context change.
- Handle high loads to cope with heavily loaded CVEs to avoid compromising usability.

The first point, defining input devices, is not just classifying an input device according to certain classes criteria and features. Instead, the device should be *explicitly* defined, or precisely described through a set of attributes.

The requirement demanding description in contrast to classification is to make it possible to support automatic inference based on a set of rules that are checked against the attributes of the device for which the decision is made. Having decisions based on classes of devices rather than specific devices won't make this possible. Just to illustrate, if there was a class for game consoles' controllers, then the XBox, PlayStation and Wii controllers should all three be treated the same and that is not practical.

The second design requirement focuses on the other end of the interaction, the application task. Interaction interoperability uses the concept is based on mapping an input device manipulation in a specific dimension within a specific range to an execution of an application task. For this, a profile of the application stating all the tasks afforded and the attributes of each should be predefined. Component based applications usually do have a profile for integration purposes, mainly concerned with function calling and programmatic interfaces in general. Having such a profile would still be different from the profile expected for *interaction interoperability* to take place.

Next requirement is to support legacy DVEs without demanding special or ground up development and that is to opt for the highest practicality possible and to offer trying out new devices and interactions against legacy application which already exist.

That point would be for the sake of the users who want to try out new interaction techniques and that would also make interaction interoperability a great testbed for evaluating interaction techniques against fully fledged applications already in use in contrary to evaluations done on application mock ups and prototypes. The legacy applications include a lot of features and complexities that are not found in application mocks and prototypes used in usability studies.

The main objective behind the framework and concept is personalization (backed by a large body of literature) an how it elevates user satisfaction and performance. Profiling and modeling the user interaction patterns given a certain context would make it possible to offer personalize interaction interoperability which is context-sensitive as well.

Given all the previous design requirements, these support the capability of making up new mappings based on user profile and context change. This capability is possible given the three complete profiles of: the input device, application used and user. With a set of rules reflecting the user interaction patterns in association to different contexts, it is possible then to select and enforce certain mappings for recurring cases and even infer new ones based on similarity of factors present.

Finally, it is feasible to have thousands and even millions of concurrent users in DVEs. Therefore, the framework should sustain high loads and should not to negatively impact the DVEs, reduce responsiveness and defeating the whole purpose of giving the user a more enjoyable, efficient interface.

These set of points highlight the major and not all the design requirements needed. These are the crucial ones and more are to be discussed in trying to satisfy each of these requirements.

Figure 3.3 illustrates the proposed approach of having a service-centric interaction interoperability framework. The whole system is based upon a service-centric architecture which basically resembles a set of discoverable web services that are totally uncoupled and distributed with the

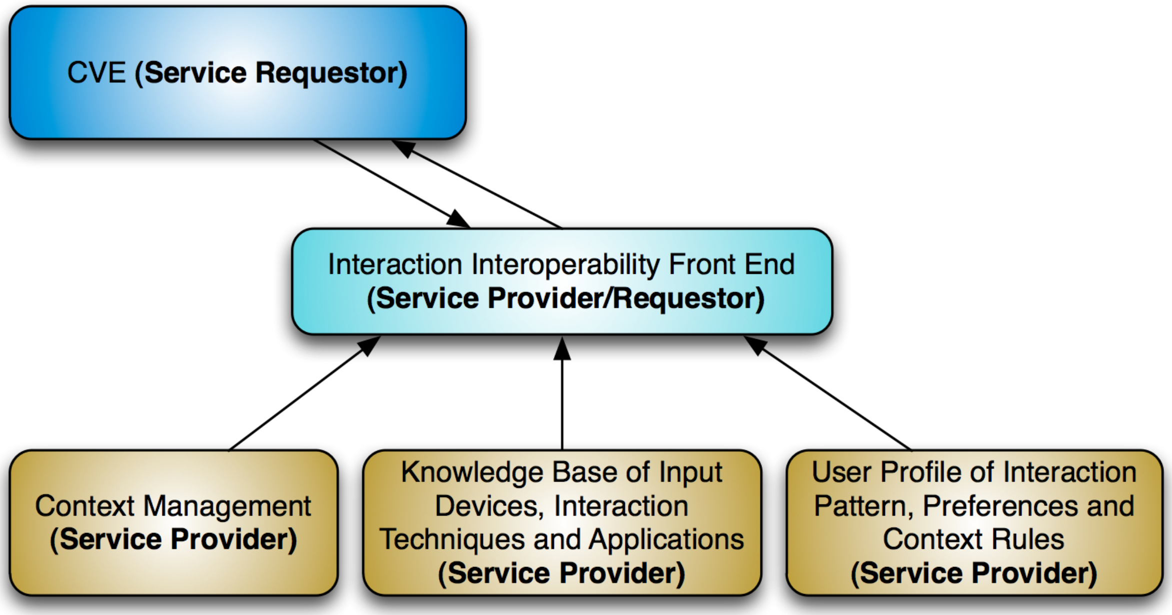

CVE (Service Requestor)
Interaction Interoperability Front End (Service Provider/Requestor)
Context Management (Service Provider)
Knowledge Base of Input Devices, Interaction Techniques and Applications (Service Provider)
User Profile of Interaction Pattern, Preferences and Context Rules (Service Provider)

Figure 3.3: Interaction Interoperability Service Centric Proposed Approach

main front-end web service being a complex service that consumes others in doing its job.

In Chapter 4 Context Management is discussed by contrasting related work and design requirements. The downsides of classical systems are described, as well as how improvements were incorporated to satisfy the design requirements of interaction interoperability. Following that Chapter 5 presents the knowledge needed by the framework to offer interaction interoperability. Chapter 6 builds on these two chapters presenting the framework design.

Chapter 4

Context Awareness and Management

There are different definitions of Context Awareness. They are mostly derived from the definition of Panayiotou [Panayiotou, 2000] which is “*A set of premises expressed in some language, gathered intentionally or unintentionally in a relevant, coherent manner and which can itself constitute and adequate set of inferences (meaningful) or lead to some meaningful results (inferences)*.” The most widely accepted and used definition though was the one placed by Dey [Dey, 2000], “*Any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*.”

Context, as defined in literature, mainly influenced again by [Dey, 2000], is made up of four different primary dimensions; identity, activity, time and location. Using these four primary dimensions, the secondary dimensions could be derived (e.g. the user schedule, and so forth).

As stated in Section §3.2, these four context dimensions should be elicited from both planes, the real and the virtual one. The identity is provided by the user identity in the real world and the avatar character in the virtual world. For location, there is the physical, real world location and the symbolic location in the virtual world. For time, there is the current , real world time at the physical location and the time in the virtual world. Finally, for the activity dimension, there are two different activity states of the user, what the user is doing in the real world and what the user is doing in the virtual world.

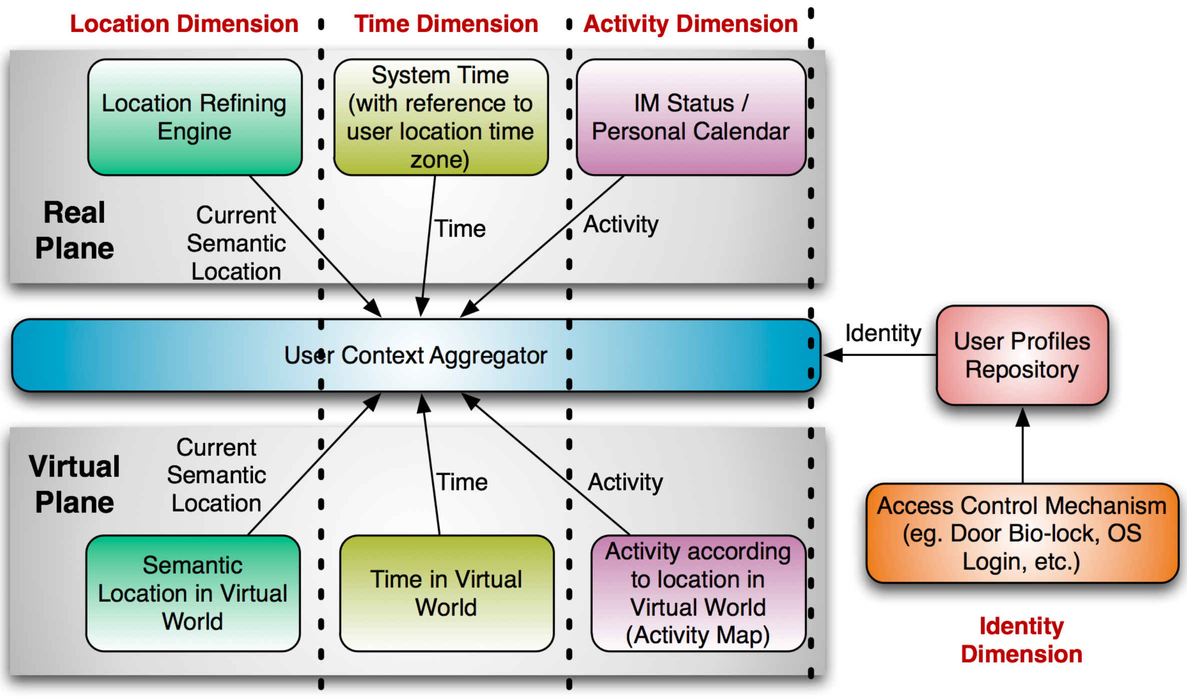

Location Dimension
Time Dimension
Activity Dimension
Location Refining Engine
System Time (with reference to user location time zone)
IM Status / Personal Calendar
Real Plane
Current Semantic Location
Time
Activity
User Context Aggregator
Identity
User Profiles Repository
Virtual Plane
Current Semantic Location
Time
Activity
Semantic Location in Virtual World
Time in Virtual World
Activity according to location in Virtual World (Activity Map)
Access Control Mechanism (eg. Door Bio-lock, OS Login, etc.)
Identity Dimension

Figure 4.1: Context aggregation using the different dimensions of real and virtual planes.

Figure 4.1 illustrates the elicitation of context from the user's real and virtual worlds. As illustrated, for each of the dimensions, two different sets of data are collected. For location, in the real world it would come from physical positioning engines and GPS devices. However in the virtual world the location would just be a semantic location reflecting a specific part of the virtual world. As explained, the same would apply for the other dimensions with the only exception is the identity dimension. It is *assumed* that there is an *alignment* between the real and virtual planes. This makes sense since the user account in the real world is linked to some account for that same user in the virtual world.

Another dimension that would influence the decisions taken is a proximity to devices, resources and even the other users around. This dimension is considered to be a secondary one since it is based and basically derived from another primary one which is the location dimension.

This dimension reflects *who and what is around* the user. Given we know what input devices are available around, what computing machines and if the other users of influence are around, this should drastically affect the decisions to be made.

4.1 New Rules Generation

The majority of context aware applications and frameworks presented in literature use a set of predefined rules that takes into account different values for the context dimensions and given the conditions in place a decision is reached by inferring through all the possible rules that could fire at that time.

Supporting interaction interoperability, I *could have not* just settled for a static set of rules. The objective is to offer the most personalized experience possible and that makes it impossible to have an exhaustive list of all possible rules. Also, there is a design requirement mandating ever changing set of input devices, applications and user profile preferences. Given these requirements, I had then to make it possible for the rule knowledge base to evolve with time having new rules being added and possibly existing ones get modified or deleted.

In order to make this possible, I have built on the work of Dargie and Waltenegus in [Dargie, 2006]. The scheme they proposed is based on associating decision events which signify the user activities with a set of context primitives which are procured at the exact same time the decision events are produced given the current context. Based then on this decision-context associations, new context rules are generated.

Simply put, the idea is to literally monitor what the user is trying to do given the context, available input devices and a specific application tasks performed. This monitoring would be in a sense is that whenever a user decided to make an **intervention** the application should report this back to the framework reflecting an unsatisfied user. The feedback provided by the user can be used to evaluate the current rules in place given the current specific context setting. This set of rules, if no longer favored by the user can be modified or deleted and if no rule is available for the current setting then a new rule should be added.

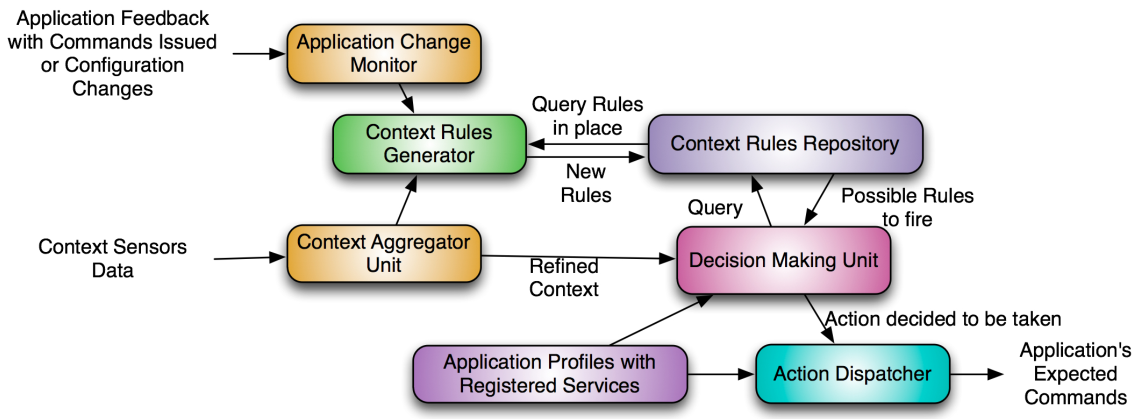


Figure 4.2: Context rules query, matching and dynamic rule generation.

Figure 4.2 illustrates the feedback loop in place to report user interventions aiding in readjusting the context rules in place.

In this chapter, the research challenges of building context-sensitive systems have been reviewed, from the definition of context to what makes up the user context and how to integrate the real plane and virtual plane contexts to resemble the user status. I went through each of the dimensions and how elicitation, aggregation and filtration takes place modularly. I have also proposed the integration of the feedback loop based extending the work of [Dargie, 2006] to accommodate the framework ever changing knowledge base in addition to the user like and dislike changes.

Chapter 5 discuses the knowledge base. Chapter 6 builds on Chapters 4 and 5 to describe the approach taken to realize the framework.

Chapter 5

Data, Information and Knowledge

In order to develop a framework that can map between devices and application tasks, that framework should be backed by the necessary knowledge about these entities. In this chapter data, information and knowledge needed are explained and discussed.

Through the following sections I will go through each level of the knowledge hierarchy [Rowley, 2007] discussing what does it involve and how that is required and used by the framework.

5.1 Data

At the data level, the framework depends on having user accounts like any other systems we deal with. The user account will have demographic data, login credentials and so forth. On top of a traditional user account I will have additional data associated with the user. Associated data include be the mappings that the user likes or dislikes. This data would be beneficial in inferring through at higher levels.

Other data repositories, similar to the user directory, are the supported applications data repository. This data include an application identifier associated with a certain platform, communication architecture and links to a set of tasks afforded which would require mapping.

The last type of data to be needed and consumed by the framework would be contextual data. Contextual data is collected for each user, including location coordinates, time, and other data sensors around. Data collected would be used in reasoning and in resembling a certain meaningful context as it was stated in Chapter 4.

5.2 Information

Based on data, we infer information. Given a user account, and mapping preferences, a user *profile* is formed. Gathering contextual data for a certain user at a certain time would also make up as another piece of information. Another major information part are the ontologies of input devices and interactions to be executed.

The problem, as pointed by Buxton [Buxton, 1983], is that the design of interfaces and input devices is mainly based on creativity. In order to infer and make up mappings for devices to applications, there should be a formal definitions rather than just classifications through taxonomies. I have designed an input devices ontology in order to define, categorize devices and provide governing rules maintaining consistency overtime given the extensibility support which would make it possible to extend and manipulate the ontology.

The input devices ontology is illustrated in Figure 5.1. The hierarchy starts with the parent class *PrimitiveDevice*, named after Bleser and Sibert's [Bleser and Sibert, 1990] with the difference is that in [Bleser and Sibert, 1990] it was an aggregation relationship instead of inheritance.

At the next level of the hierarchy, comes *Digital Switch, Position Sensing, Motion Sensing, Force Sensing and Speech Command* reflecting that I have took into account the lexical and pragmatic considerations raised by Buxton [Buxton, 1983]. Concerning that same level of classes — inheriting *Primitive Device* —, I contrast my work to Lipscomb and Piquene [Lipscomb and Pique, 1993]. They had two different graphs, one for tabletop devices and another for elevated devices (held by or mounted on user). My approach was to simply have these as subclasses. A limitation is that the taxonomy in [Lipscomb and Pique, 1993] is just for analog devices. I included other classes such as

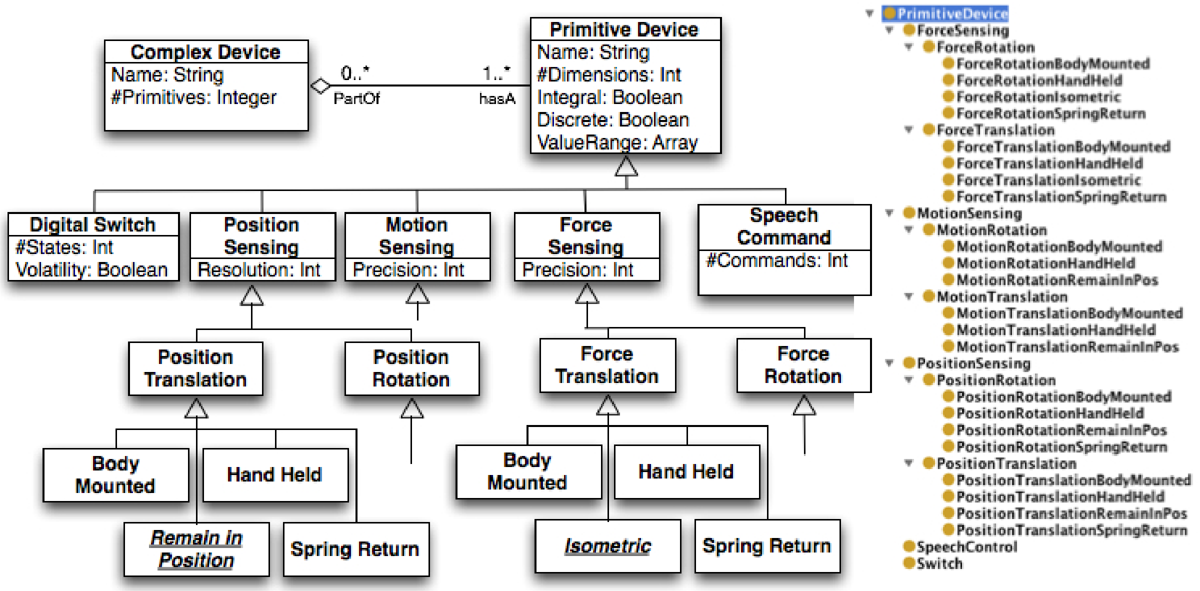


Figure 5.1: The class hierarchy of the OWL-DL Input devices ontology showing inheritance, some properties and some of the cardinality restrictions (all classes are shown on the right side).

Digital Switch and *Speech Command*.

Lipscomb et al. [Lipscomb and Pique, 1993], contribution was a classification based on the physical characteristics. I address that at the next level down the hierarchy with the subclasses: *Remain-in-position, Spring-return, Isometric, Held-up and Body-mounted devices*. There are some super-classes however that do not fully fit. As an example, a position translation device cannot ever be an isometric device which is defined in [Lipscomb and Pique, 1993] as "*devices that have such a large spring constant that they cannot be perceptibly moved*".

The perceptual structure, proposed by Jacob and Sibert [Jacob and Sibert, 1992] indicates if the device is integral or separable. I facilitated such discrimination with an attribute of the superclass *PrimitiveDevice*.

Highlighting the difference between a subclass and an instance (e.g., an XBox controller should not be considered as a *class*), it is just an instance of *GenericGameController*. Figure 5.2 shows the XBox controller breakdown to primitive devices.

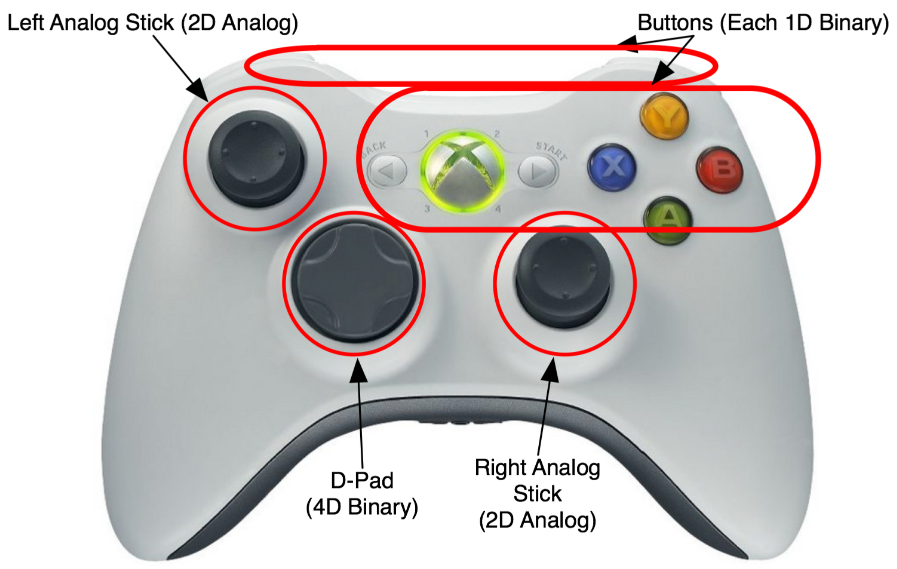


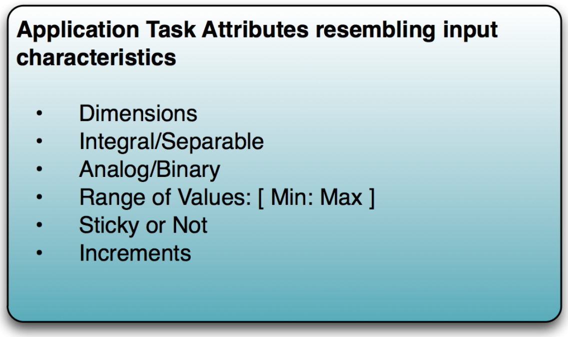
Figure 5.2: The XBox controller breakdown into primitive devices, each with a number of dimensions, nature and value range. (Courtesy of Microsoft Hardware www.microsoft.com/hardware 2010) Public Domain

However, if there is an innovative device that does not resemble a class, a new class can be created under the appropriate hierarchy. Referring back to properties discussed previously, if a new addition is well defined with appropriate properties, a reasoner could easily infer the type of that new entity.

To test for effectiveness, I have created different classes and individuals to see how flexible and manageable the ontology is. A number of devices, including 2D and 3D ones were added including: simple mouse, light-pen, head-trackers, analog sticks, D-Pads and complicated game controllers. One of the merits of a semantically rich repository, creating major subclasses makes adding further devices with all their appropriate superclass properties a straightforward process.

In Chapter 7 sample devices and mappings are presented. Chapter 8 presents the evaluation of extensibility and revisits the attributes used in defining a number of devices and mappings it to application tasks.

For interaction tasks, the classes used are based on the taxonomies provided by Bowman et al. [Bowman et al., 2005] for 3D interactions. The tasks belongs to one of the classes: Selection, Manipulation, Navigation (Travel and Way-finding), Control or Symbolic Input. However, given an object-oriented description of an ontology, it is possible to have constraints allowing the accommodation of new classes through inheritance while maintaining consistency and enabling inference. To further ease the mapping of an input device to an application task and assure higher suitability of the association, the attributes for defining the tasks were designed to resemble the devices' attributes. Figure 5.3 shows the attributes that precisely map to the input device. Comprehensive examples of case studies are presented in Chapter 7 and Chapter 8.


Application Task Attributes resembling input characteristics

- Dimensions
- Integral/Separable
- Analog/Binary
- Range of Values: [Min: Max]
- Sticky or Not
- Increments

Figure 5.3: Task attributes resembe input devices attributes to assure better mapping and avoid possible mapping problems like domain and range.

5.3 Knowledge

Given the data and information discussed, at the knowledge level, rules and constraints enforced as part of the ontologies are used by the inference engine to classify new additions and make up the mappings between an input device dimension range and an application task.

In addition to the knowledge base made up of the input device, interaction techniques and application profiles and constraints, another part of the knowledge base is made of the context rules discussed in Chapter 4.

Not every context instance would have a rule that handles it. However, for a context made up of dimensions (discussed in Chapter 4), a number of rules could possibly fire for conditions based on one or more dimensions satisfied. Also, as stated previously, this knowledge base is ever changing, not just by adding new rules but also by altering and even deleting rules that do not hold anymore, as discussed in Section §4.1.

Having a large, ever changing set of rules though for multi-million users demand a high performance rule engine to handle rules execution set. Going for the classical *if-then* statement set would cause two major problems. First, if-then statements are part of the code that is to run and that means if the rule set changes then this means the it should be possible to make on-the-fly changes to running code. Not many languages support this (in addition to security and performance issues). Second, the ability to enforce new rules or change the working set of rules should again comply with the interoperability objective of the framework.

The way the framework deals with context-awareness rules is by relying on database queries that satisfies the current conditions. Every context rule is stored in a relational record with reference keys to all the context dimensions that makes up a context. A typical database record would include reference index fields to: the user identity, location, time, user activity and the machine being used. Loading and monitoring the working set of context rules translates then to a query given the context being fed through the framework Context Management module. The result set retrieved would resemble the rules *that exists* and are *satisfied*.

For context rule generation, that would translate then to updating the existing rules or deleting them As stated previously in Chapter 4, a user intervention is used as a feedback to update the context rules set stored for the user. Receiving the intervention through the framework front-end web service, a query is made to retrieve what is already there for the context to be modified. If a result set is retrieved this means that there is a rule or more that are in place for that same context that the user was unsatisfied with. If no results are retrieved on the other hand, this means that there are no rules that specify that context and therefore there were no actions taken. Given the former case, the set of records reflecting the satisfied rules are checked to see the discrepancy between what was retrieved and what the user decided to go for. An update or a delete is issued then to assure that this context does not carry any unsatisfactory rules and the new records are created reflecting the current context and what the user have selected.

If a rule engine is to handle a working set of rules then this simply means that feeding the engine a different set of rules sent in an interoperable way, basically XML, would elevate again distribution and system interoperability. Going for standardization and yet also flexibility, JSR 94 of Java Rule Engine API puts forward a set of constraints and rules to be met. Any rule engine that comply to JSR94 can then be used and accessed from any Java application giving great flexibility and interoperability as well. One such engine is Jess [jes, 2010].

Through this chapter I have discussed all the knowledge needed to support interaction interoperability. In the previous chapter (Chapter 4) context management have been explained and discussed. Now, building on these two main pillars, Chapter 6 describes the design of the Interaction Interoperability framework that would satisfy the research challenges and design requirements.

Chapter 6

Interaction Interoperability Framework Design

In this chapter I present the design approaches that were taken to put the framework in place and make interaction interoperability possible given all the constraints and requirements. Making use of the extensible knowledge base discussed in Chapter 5, supporting context awareness discussed in Chapter 4 and handling high loads typically expected for DVEs and games as shown in § 1.2.

Through the following sections, the architectural design pattern and the framework architecture that would satisfy the system requirements are discussed.

6.1 Architectural Design Pattern

Given context sensitivity, the distributed nature of the framework, and other system requirements (mainly scalability) a design pattern that would offer such flexibility should be adopted. Costa et al. [Costa et al., 2005] presented three different architectural patterns that could be applied in developing context-aware services. The three proposed patterns are Event-Control-Action pattern (ECA), the Context Sources and Managers Hierarchy pattern and the Actions pattern.

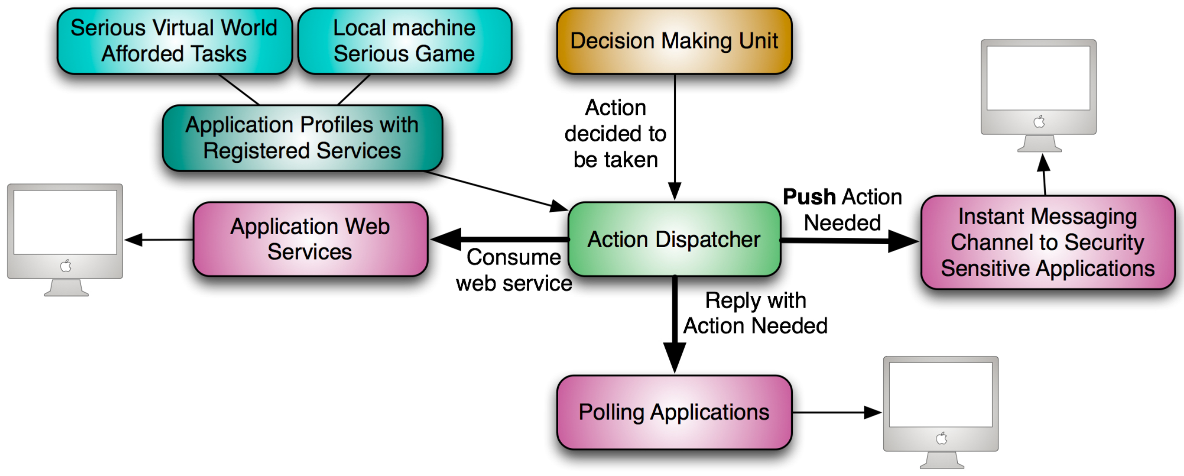

Serious Virtual World Afforded Tasks
Local machine Serious Game
Decision Making Unit
Application Profiles with Registered Services
Action decided to be taken
Application Web Services
Action Dispatcher
Push Action Needed
Instant Messaging Channel to Security Sensitive Applications
Consume web service
Reply with Action Needed
Polling Applications

Figure 6.1: Task abstraction and execution for interaction interoperability supporting various types of applications.

ECA was the one widely adopted pattern because of decoupling context acquisition from reacting to changes. With this, the ECA architectural pattern provide a structural scheme that enables distributed functionality in service platforms by separating, gathering and analyzing context information from triggering action in response to contextual changes, under the control of an application behavior description.

According to [Costa et al., 2005], the assumption is that context-aware application behaviors are described as condition rules, in the form of *"if <condition> then <actions>"*. Conditions are logical combinations of events reflecting a certain situation which is then followed by triggering of actions based on the control rules in place. Another crucial part is the Controller component that checks if a condition turns true and it executes the Action Performer component that triggers the actions described in the satisfied rule.

One level of ECA may not be sufficient to deal with a number of dimensions in real and virtual planes. Therefore, I used a hierarchy to assimilate contextual information from different sources. Still, when it comes to action, both patterns suggests a single action to be taken which may not be enough to support our goals. Given this limitation I decided to have a *hybrid* pattern based on ECA with a hierarchy of Context Managers and Sources and the actions to be taken following the Actions pattern.

The hybrid architectural pattern I have adopted supports all the requirements while also eliminating shortcomings thus making it possible to achieve a decoupled, distributed framework.

6.1.1 Framework Architecture

Supporting distributed virtual worlds like Second Life or Wonderland implies that we should be capable of handling a large number of *simultaneous* online users. In order to afford scalability, reliability and availability the framework should be based on a distributed architecture, given the disadvantages of a autonomous system in such settings.

The framework should be capable of supporting multiple platforms so this mandates that the chosen message formats and transfer protocols should be content and network interoperable.

Given that certain rules could fire automatically with a change in context, there is a requirement that the framework back-end should be capable of **pushing** commands instructing applications to switch behavior.

Given the system requirements, the best architecture candidate would be the service oriented architecture (SOA). The SOA has a number of features and advantages that would satisfy the requirements discussed previously.

By definition, SOA is based on *discoverable services* fulfilled by *loosely coupled functionality units* that accept *platform independent messages*, basically XML.

One requirement that still won't be fulfilled simply by SOA is the *push commands* requirement. Since the entry points will be web services which are to be **consumed** by client applications, the solution to this design problem was to add a push capable protocol. The ideal candidate for such a protocol that would enable the framework to communicate commands with applications running on the user host machine without compromising the security of both ends would be XMPP [XMP, 2009].

Figure 6.1 illustrates how XMPP supports the design goals of a distributed framework by having a number of XMPP servers and passing over commands from and to clients offering higher scalability, reliability and with no single points of failure as well as improved availability (Figure 6.2).

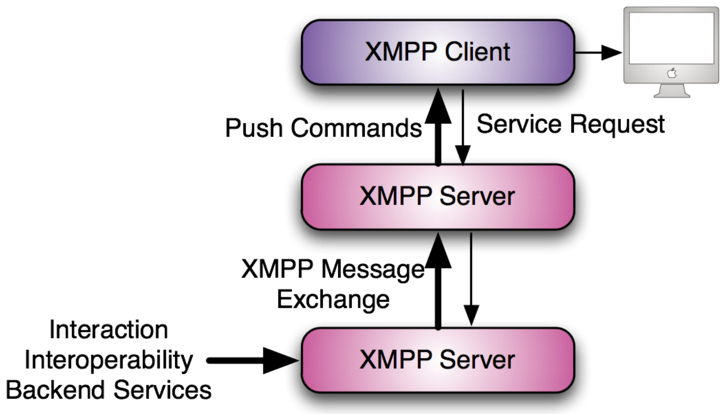


Figure 6.2: Communication based on XMPP protocol through Instant Messaging Servers and Clients

6.2 Context Switching

As previously discussed, the user's context can frequently change and context-sensitive rules are activated to reflect the change in context.

The problem I just posed is that context switching in the majority of the context-aware environments would not involve a *radical* change of the interface. That would mostly likely involve alerts or even slighter changes like colors and contrast (as example switching a monitor to a different color combination depending on time of the day). However, for context-aware interaction interoperability, taking a context sensitive action would involve switching from the *one input device to another*. Even if the device is not changed, the interaction technique can change, i.e. the user should *change* the way the input device is used in order to accomplish the same application tasks in a different way.

From the technical perspective there are no problems, given the architecture I have discussed. From the usability perspective though, that may seriously affect the user. Personalization and context-awareness would then be perceived as intrusive hindering user performance.

To tackle this problem, the approach was to overlay a dialog box (or just a question with a yes, no prompt) or menu. Whenever the context change and that change requires an action to be taken, this action should be **confirmed** *(as opposed to taken)* by the user. The user can ignore it and continue using the same interaction technique or accept the change and be *aware* of *what is going to happen* and *why*.

When there are frequent context changes, if the user chooses to ignore a suggested action at a certain time, this decision would count as a rule and this suggestion would not be repeated again. Fluctuations (repeated context swings within the same time frame) are handled differently. When a fluctuation is detected, the user is asked then to either *ignore that* ***same*** *switch in the future* or not. The difference from the the previous case is that the feedback is not considered negative, the rule would still hold but fluctuations in *time locality* would be eliminated.

6.3 Framework Design

Based on Chapter 3, there are four different areas to manage; input device profiling, application profiling, user modeling and context management. The framework is built up of four main units corresponding to these four areas in addition to a fifth unit that would be the decision making unit.

Figure 6.3 illustrates the framework five main modules. Going in a hierarchical fashion, the five modules and their internal submodules are discussed through the following five sections.

6.3.1 Input Devices Management

As stated in the previous section, input device abstraction is the basis of interaction interoperability being able to flexibly assign different devices to application afforded tasks. As stated in the related

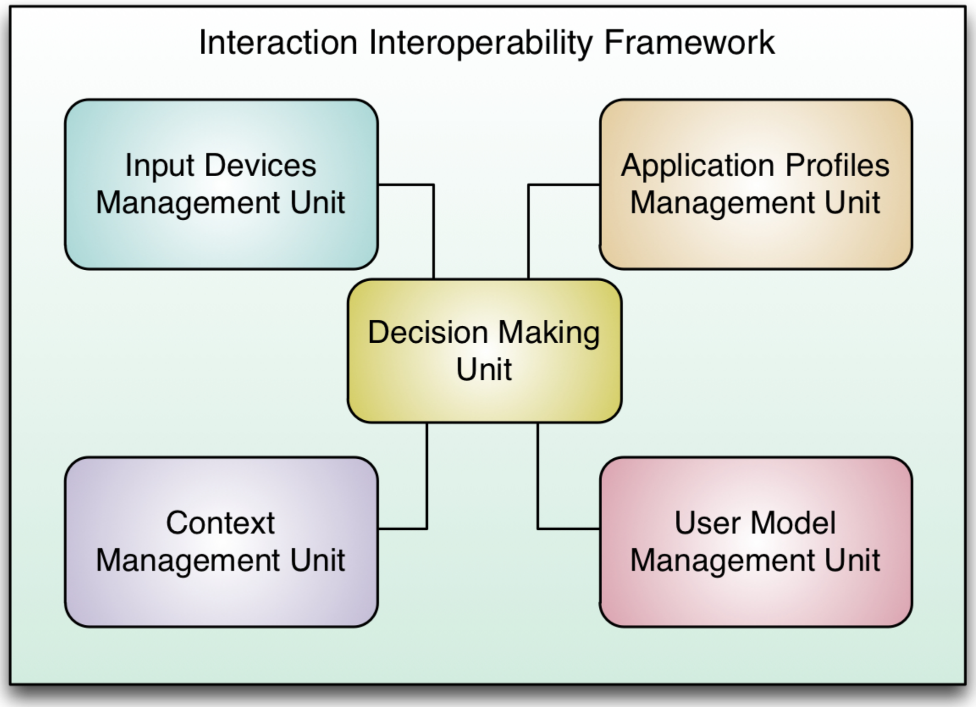

Interaction Interoperability Framework
Input Devices Management Unit
Application Profiles Management Unit
Decision Making Unit
Context Management Unit
User Model Management Unit

Figure 6.3: Interaction Interoperability Framework Five Units.

work (Chapter 2), previous work was trying to basically classify input devices based on different factors. I believe that just classification won't be enough to base the decision of assigning a device to a task. What is proposed here is to actually define and describe each input device to make it feasible to have an automated decision making unit assigning and mapping input devices to applications.

The Input Devices Management unit is made up of two modules; the Input Device Abstractor and the Input Devices Repository. The abstractor module is used to abstract the device breaking down the definition into a set of questions that would unify device definitions. The abstractor module should be able to read a profile document of a device that would typically include the number of controllable dimensions, the domain range of each of the dimensions in addition to some other physical characteristics listed below based on the work in [Buxton, 1983, Card et al., 1990, Jacob and Sibert, 1992, Jacob et al., 1994, Lipscomb and Pique, 1993, Jacob, 1997]:

1. The perceptual structure (Integrality versus Separability).

2. Rotational versus Translational; nature of movement.
3. Unbounded range versus Bounded Range of values.
4. Volatile or Non-volatile; keeping the mechanical position and corresponding output value when released or not (Remain-in-position versus Spring-return).
5. What is being sensed (position, motion or force).

How easy is adding new devices and manipulating ones already listed would be crucial in terms of the usefulness of the framework. To accomplish this in an easy fashion, the user is guided through a series of questions about the nature of the device and based on the answers given the device profile would be created. Guiding the questions being based on previous answers would limit possibility of errors. Another intuitive way to handle device profiling would be to communicate with the device, if possible, through the operating system or platform and try to get as much information as possible about the device at least again to minimize human errors.

Regarding the other module of the unit, the Input Devices Repository, it would be the module responsible for listing all device profiles while maintaining consistency, minimizing possibility of errors and conflicts. The device profiles repository should be expandable allowing additions, manipulations and omissions and again while enforcing consistency rules. Implementing such a repository is non-trivial, for example using a flat file or even a database table would not be enough to suffice the requirements.

6.3.2 Application Profiles Management Unit

The Application Profiles Management Unit is made up primarily of two components; the Application Profiler and the Profiles Directory. The Application Profiler has the role of building up the profile of an application. As mentioned in Chapter 3, the application profile mainly lists the afforded tasks with details of each needed to satisfy the task. In order to facilitate mapping from input devices abstracted in the way stated in Section §6.3.1, the application tasks again cannot be simply classified according to different classes like navigation, selection, manipulation and control as in [Bowman et al., 2005]. More details are needed:

1. How many dimensions make up the task (as an example positioning a mouse cursor in just 2D or 3D)?
2. What is the range of values expected by the application for each of the dimensions?
3. Is the range of values discrete or continuous?
4. Is the expected value bounded or not?
5. Is the expected value sticky or not (controller should be kept in position or not)?

As stated in Chapter 2 there was previous work in dealing with auto profiling applications. In case of the application with no such support (legacy applications), similar to the approach taken for profiling devices, implementations can be based on wizards guiding the profiling of an application.

For the second component, the Application Profiles Directory, this is typically a globally accessible directory holding the different profiles of application versions allowing manipulation of records only based on specified credentials. The global access eliminates the need to re-profile applications, once an application has been profiled (automatically or manually) this record should be usable by all service subscribers (consumers).

6.3.3 Context Management Unit

The main component of the Context Management Unit is the Context Aggregation Module. The module is responsible to collect and aggregate user contextual data through different providers of data (location, activity, time and other secondary dimensions). There are still three submodules that are responsible for collecting each of the three dimensions; location, activity and time since each dimension is not likely to be collected through a single source.

Activity Aggregation Submodule Collects user activity status through a number of venues. The first one is through the local IM application of choice. The second one is through user calendars (desktop, web and mobile). The third one is through the activity map of the DVE currently being used.

Location Aggregation Submodule Collects all location data on different precision levels and unifies the location after that. Sources of location information include GPS enabled devices, indoor positioning engines with a room size precision (e.g. WLAN based) and finally, more precise positioning sensors (e.g. Bluetooth and RFID).

Time Aggregation Submodule Again, as highlighted in Chapter 4 there are two planes to deal with, the virtual and real, there will be at least two times to track and would impact the context and these would be the time in the user real world and the time in the virtual world.

Something to note here is that the Context Management Unit should not be considered as the unit responsible for storing context changes across time, user preferences or even the context rules. The unit just consolidates a meaningful overall context based on all the context dimensions for a user across different precision levels. Figure 6.4 illustrates the unit and the three submodules.

6.3.4 User Model Management Unit

One of the crucial components of the framework is the User Model Management Unit. User modeling from the perspective of interaction interoperability is to not just store basic data about users but to record interaction patterns. Patterns reflect what interactions did the user like and others disliked with a certain application in a given a certain context. As stated in Section §6.3.3, context updates are not just consolidated through the Context Management Unit and not stored, in this unit then context updates, likes and dislikes are stored for every individual user account.

Individual users profiles are not static then given this constant updates. The user profile actually evolves over time. The decision making unit (discussed next) references the profile of the

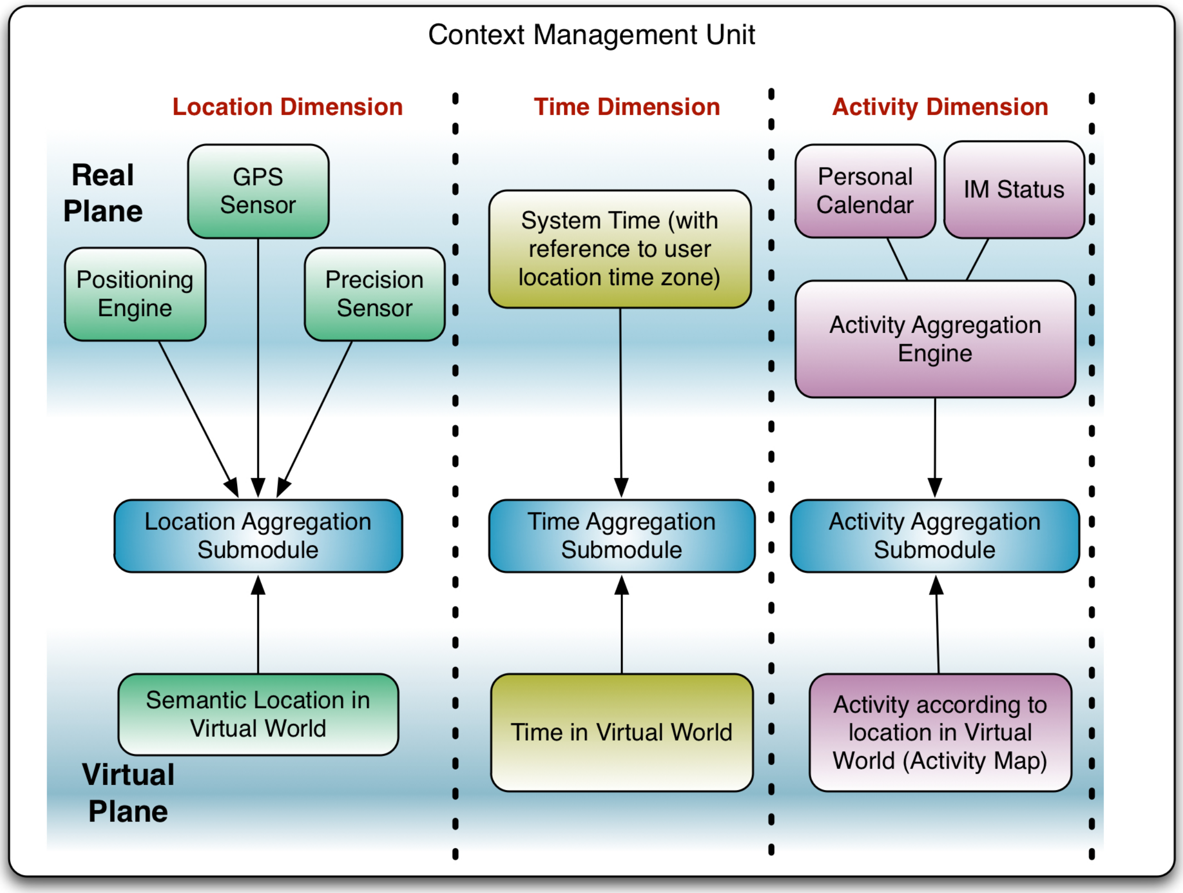

Context Management Unit
Location Dimension
Time Dimension
Activity Dimension
Real Plane
GPS Sensor
Positioning Engine
Precision Sensor
System Time (with reference to user location time zone)
Personal Calendar
IM Status
Activity Aggregation Engine
Location Aggregation Submodule
Time Aggregation Submodule
Activity Aggregation Submodule
Semantic Location in Virtual World
Time in Virtual World
Activity according to location in Virtual World (Activity Map)
Virtual Plane

Figure 6.4: Context Management Unit and its three context elicitation and filtration submodules.

user and updates it again with any necessary changes which may include using different applications, devices, interaction techniques across changing context. Historical records are fundamentally the rules that would guide the mapping decisions derived.

6.3.5 Decision Making Unit

The Decision Making Unit is the unit responsible for actually making up the mappings necessary to match an input device to an application given a certain user in a certain context. The main building blocks of the Decision Making Unit are four blocks; Active Rules Repository, Rule Engine, Action Dispatcher and Context Rule Generator. Starting with the first block, the Active Rules

Repository, the reason for this block is to actually make the framework feasibly scalable. Given context sensitivity, this means that any change in context would need the system to go through the rules and check to see if any conditions are satisfied to fire. Having a huge user base, with every user having his own set of individual rules would create a huge load to process all these rule. The idea is to have an Active Rules Repository that holds only the currently active rules. Whenever a user logs into the system the rules of that user are pulled from his profile and added to the set of currently active rules. That set is the working set of rules that are to be monitored for conditions satisfaction. Again, once the user logs off, his/her associated rules are removed from that active set minimizing the overhead on the rule engine.

As expected with a set of changing rules, having *if-then* statements hard-coded somewhere would not be a viable solution. There is a need for an efficient rule engine. The Rule Engine module would be responsible to work on the active rules repository, executing actions of rules that have their condition satisfied. Given a *dynamic* active set, the engine would have to handle only those active rules not the exhaustive list associated with all the user base profiles.

Based on the set of rules and the rule engine, the next step is executing action for the satisfied rules. This is done through the Action Dispatcher module which is called by the Rule Engine upon a satisfied rule that is fired. Given the action type of the rule, action host and even the current context the action dispatcher would execute the action intended accordingly, being it a *push command* to an application or some *reply to another polling one* or even *consumption of an application service*. This could have not been done if the framework was implemented just to enable certain applications running on a pre-specified platform so the actions taken through the rules would be specific action all done in the same manner.

The last module in the unit is the Context Rule Generator. From its name it sounds as if it should be part of the context management unit but actually it is not. Typical context aware systems have a set of pre-defined rules created at design time. This could not be possible to support the framework. The framework is dealing with ever changing repositories of devices, applications, users and continually changing contexts. To keep up with changes of user preferences and other conditions with no pre-defined rules in place, the rule generator monitors running applications with users logged in. Once an action is taken *at the application side* the conditions *preceding* that action are analyzed, what was the context, what application and what did the user like or dislike, given *this action*. The *status* of the rules repository is checked to see if a rule exist with that *condition-action* combination. If so, nothing happens. If not, it would try to find rules with the same condition but not action and those should be deleted and another would be created reflecting the new favored condition-action combination. It is apparent then that the output from the generator is sent back to the set of rules.

6.3.6 Incorporating devices and applications within the framework

Three scenarios were presented in Chapter 1 illustrating the idea of integrating legacy and framework aware applications as well as defining new devices and applications. Below are the points needed to incorporate devices and applications to the framework presented.

Incorporating a device in developing an application involves the following procedure. First, the device should be connected using a supported application accessible connection. Following that the device driver and supporting development library should be integrated in development. Coding the application, given the classical event driven programming model, an event handler for the device should be created without hard coding any mapping to application tasks. The novel device should be defined then and the event handler will get the task execution mapping from the framework during runt-time given the logged in user and his/her current context.

In incorporating a device to support legacy applications, again given it is connected and supported by the operating system, the next step would be in utilizing a device emulator that would generate/emulate the legacy application expected events. Again, the device if not already defined in the the framework, the task should be defined. Achieving then context aware mappings between the defined device events and tasks, the emulator is fed through the client mediator running locally a set of mapping rules to generate the emulated events accordingly.

To incorporate an application within the Interaction Interoperability Framework, one starts by defining the application, version and platform. The application tasks are then defined and for each task the class of task is specified (selection, manipulation, navigation, symbolic input). Task dimensions and dimensions attributes are then defined Given the service centric architecture of the framework, a services client proxy can be automatically generated by whatever development platform is used. During runtime, the application can then consume the framework services to get the context relevant mappings for current user for the devices available at his disposal.

6.3.7 Supporting Legacy Applications for Evaluations

As previously stated, the idea of supporting Reality Based Interaction (RBI) evaluations is based on affording to test new input devices and interaction techniques with already existing real world applications that users do already have a background experience with. In order then to support *any legacy* application without the need of having that application aware of the interaction interoperability framework this means that this intervention should be totally transparent to the application layer. The idea of achieving such transparency is actually widespread and its based on having a platform specific hardware emulator that would capture input events from the operating system and generate other fictitious ones based on the application expectations. It is still obvious though how the framework stand apart from such low level hacks given automated mappings generated based on the input device profile, application profile and user context and preferences.

To facilitate this transparency for legacy applications without the need of low level user intervention there is a client-side unit that would handle this functionality, the Client Mediator. The mediator unit is used to support legacy applications. It is made up of two parts, the communication module and the input device emulator. The communication module is responsible for communicating events and requests to the framework and waiting for actions to execute based on decisions taken at the back-end. The input device emulator is a platform specific module that interprets the input of input devices not supported by legacy applications to other expected actions. Off the shelf emulators can be used given the scripting of emulation mappings are generated by the framework back-end.

Chapter 7

Reference Implementations

The framework laid out in Section §6.3 could be implemented in a number of various ways and based on several architectures including a standalone desktop application, client-server (with thin or thick clients) and the service oriented architecture. Different architectural pattern would have advantages and disadvantages, especially given the set of requirements put forward with a number of trade-offs.

In trying to satisfy the requirements set forth, I decided to use SOA. Brien et al. [O'Brien et al., 2007] defined SOA as an architectural approach for building systems where there are components that are either service users or service providers, with a service being a distributed component with the following characteristics:

- Self-contained.
- Has a published interface abstracting underlying logic.
- Location transparent.
- Implemented in different languages or platforms and still inter-operate;
- Discoverable and dynamically bound.

It is obvious that the characteristics do align with the requirements of the framework and basing the development on SOA means also that the language and platform selected would not limit the usage to specific platforms only.

Four different case studies were conducted as a proof of concept to evaluate the impact of interaction interoperability framework on usability issues as well as to have an estimate measure of incurred delays of the framework.

One of the case study was a desktop game built from scratch, the other three were the integrations in the frameowkr of Second Life (SL) [Sec, 2010], Sun Wonderland [Won, 2008] and X3D [Web, 2010]. The details of the preliminary implementations are discussed in Section § 7.6.

The fifth case study demonstartes the implementation of the framework building blocks as presented in Section §6.3. The chpater concludes with a discussion of how supporting legacy applications and using the framework as an evaluation testbed has been realized.

7.1 Input Devices Management

As stated in the design chapter, Input Devices Management entitles *abstracting* the devices and handling a repository of all devices added to the framework.

In abstracting the device in order to ensure that this process would not be too complicated, the decision was not to useXML definition or a standard RDF record. The implementation is based on a web based interface that goes through the class definition of the input device one by one asking the user a series of questions to make up the device profile.

Typical questions on the device profile creation wizard are:

1. Is your device a complex one made up of more than one primitive (analog sticks, buttons, etc..)?
2. How many primitive devices make up your complex device?

3. Name your primitive device (e.g. Left Analog Stick, Button X).

4. Does the primitive device have on/off states (Binary)?

5. How many controllable dimensions does the primitive named (e.g. Left Analog Stick) has?

6. Does primitive device “A” keeps its mechanical position when released or not, i.e. is it spring loaded like a joystick or a touch button or a normal button that pops up again if you released your finger?

7. What is being sensed by device “A” (position like a slider, motion like a mouse cursor, or force like a joystick).

8. Name dimension #1 of primitive device A.

9. Does dimension “X” has an unbounded or a bounded range of values?

10. What are the Minimum and Maximum values of dimension “X”?

11. Do dimensions “X” and “Y” of primitive device “Left Analog Stick” change together or independent of each other (e.g. moving the device would possibly change the value of both dimensions)?

Answering the set of questions going through the device definition wizard the device record would be checked for possible conflicts. However, the guided questions wizard was designed in a way to avoid conflicts or misleading definitions that would render the framework unable to match the device and its dimensions to application tasks.

Going through the the device description wizard, the result would be comprehensive record of the device. Through this record future mappings could take place Given all the needed information, in contrast to only classifying the device, mapping can be inferred depending on the nature of the device, perceptual structure, physical characteristics, number of dimensions, range of values, and so on.

There are three different options for storing device records. The first was to have the devices stored simply as a database record that has referential relation to all other factors considered (dimensions, primitives, types etc.). The advantage of this approach is primarily simplicity and remarkable interoperability and distribution capability. The downsideis that it would be so difficult to constrain access to database table and enforce rules that would eliminate the possibility of redundancy and conflicts that would hinder inference and mapping.

The second option uses Resource Description Framework (RDF) records [RDF, 2010]. RDF was primarily motivated to have descriptive meta-data of web resources for use in cases were information would be processed by applications, instead of only being displayed in a browser. The strength point of RDF (building on XML) is in having a common framework to express *information* that can be exchanged between applications without losing its meaning due to lower level technical issues.

If the design the implementation was to follow the RDF standard then to have a repository in which records could be stored, queried and managed an example of a decent choice would be Sesame, an open source RDF framework [Ses, 2010]. However, even with RDF records there would not be constraints and governing rules to maintain consistency.

Following the framework design, the ideal implementation venue was through the Web Ontology Language (OWL) [OWL, 2008]. A major problem here from the implementation perspective was how to go for ontologies management and still have a distributed system. The widely used tool for building and managing OWL ontologies is Protege [Pro, 2009]. There is a web server implementation that manages an OWL file but this would not be a distributed system since there is no distribution of ontologies. With the objective of having a scalable, inter-operable system it should be possible to have the ontologies built distributed among a number of servers.

With the drawback of the traditional OWL file based systems and the dire need to have the devices described in such a highly descriptive OWL ontology the decision was to use the Jena framework [Jen, 2010].
Jena is a Java open source framework that has an API that allows working with RDF and OWL,

supports SPARQL [SPA, 2008] and includes a rule-based inference engine which is also needed by the Interaction Interoperability framework.

The reason for choosing Jena is because it fortunately eliminates the disadvantages of the other approaches. Jena enables ontology rules enforcement and is still based on storing the ontology in a relational database engine which can be easily distributed over a grid with total transparency. Given these factors, input devices can now be stored in an OWL ontology, retrieved using SPARQL queries and the repository can potentially scale well.

7.2 Application Profiles Management Unit

Similar to creating an input device profile, an application cannot just depend on its class and its task classes. An application should precisely and concisely defined going through basically a similar wizard like the interface wizard with a set of questions to build the profile.

As stated in Chapter 6, the profile should hold descriptions for the following:

- Number of tasks afforded.
- Is the task an on/off (binary) or continuous.
- Dimensions of each task.
- Are (should) the dimensions be integral or separable.
- What are the bounds of each dimension if any.
- What are the increments/interval of change.
- If mapping an analog to a binary (step down) what would be the range thresholds.
- Should the controller better be sticky or not (spring loaded or not).

The profile created through these questions should be stored in a profile storage. The framework design was laid to have a globally accessible, inter-operable interface. In implementing the storage the repository is based on a MySQL database engine storing the profile across a set of relational tables taking referential integrity rules as a major way to enforce the limited rules needed to maintain consistency (in contrary to the case of input devices).

Despite depending on a classical relational database engine to store a repository of application, the interface to add and manipulate records is restricted to the framework web service. To add a new application profile or manipulate an existing this is to be done through a web application (JSP page) that consumes the main framework web service to perform the intended actions. Enforcing such an interface not only ensures that all the rules and constraints in place but also makes the profile directory again distributed and inter-operable, given a distributed database system with a web service front end exchanging standard SOAP messages.

7.3 Context Management Unit

The Context Management unit is built up of a number of modules [Ahmed et al., 2010]. Following the complex web service model, the context management unit encompasses all the context awareness needed by the interaction interoperability framework and is accessible through web services and SOAP protocol. For the interaction interoperability framework to do its job as web service it consumes the context awareness unit web service.

According to the framework design there are three submodules each for collecting one of the three dimensions; location, activity and time. Each of these submodules in the reference implementation was a legacy system that is built upon elicitation of a certain context dimension.

For location in the the real world there is a positioning engine named Ekahau [Eka, 2010] which is Wi-Fi based location tracking system. The engine offers an API which is web accessible through HTTP calls/requests. The engine gives every object and identification number, name and description. These objects basically coincide with a physical tag which is to be attached to objects like computers and controllers and carried around by users. These tags are the means by which the engine tracks an object which is associated to a tag.

Based on the same approach of consolidating interfaces into globally accessible, discoverable web services, the context management web service acts as a liaison between the interaction interoperability web service and the positioning engine making it possible to make queries and issue commands to the position engine through a SOAP request to the context management web service.

In the virtual world, with in the case study is Second Life, some objects spread around at the points of interests have Linden Scripts [Lin, 2010] associated to them acting as an avatar radar. Whenever an avatar gets in close proximity with the designated objects in Second Life, the objects send an HTTP call with the Avatar around, from here the server the updates the location that avatar was last seen.

Regarding the time dimension, for the real world this would be obviously the system date and time. For the virtual world, the Linden Script sends over to the server the value it get from a function named “llGetTimeOfDay” given that in Second Life, days cycles are four hours long (three hours of light, one hour of dark). In contrast to the function “LlGetWallclock” which gets the actual, real world time.

For the activity dimension, the user activity is being computed based on all previous factored inputs; location in real world, location in virtual world and time in both virtual and real planes. The semantic meanings of different activities would be at busy in real plane, busy in virtual plane, free in real plane and finally free in virtual plane.

For the activity and time dimensions, there are no subsystems that actually elicit and aggregate the data read, however, still elicitation is done through servers like primarily Second Life. The front end to query for these dimensions is the context management web service, again through SOAP requests and all responses are formatted by the web service for a normalized XML response enveloped in a SOAP message.

The fourth and last dimension is identity, discovered through the SOAP request sent by an application running on certain machine. For the application to request any service from the framework in every call made to any of the web services operations, the user identity, as a unique global ID number, is sent and associated with the call or the service per se.

The context web service has all its operations consumed by the main interaction interoperability framework web service as previously stated. However, still the web service is discoverable and it does have its own WSDL descriptor. Given how autonomous the web service is, the services can still be accessed through any web, desktop or mobile application serving even systems other than the interaction in interoperability framework.

Other services offered by the context web service include more semantic rich operations. Two major examples of these operations which are built upon more basic ones are *getUsersAround* and *getDevicesAround*. As their names imply the former retrieves the list of users around the machine being accessed and this is to discover who will be using the application and who else is around the user right now. The later is used for the preparation to the mapping request. The operation *getDevicesAround* is helpful in knowing what tagged devices are available around for the user/application to utilize. Based on this set, the most preferable device would be the one of focus. A sample SOAP response is shown in code listing7.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getDevicesAroundResponse xmlns:ns2="http://ContextWebService/">
         <DevicesList>
                <id>1</id><devicename>Keyboard</devicename>
                <id>2</id><devicename>XBox</devicename>
                <id>3</id><devicename>Wii Mote</devicename>
         </DevicesList>
     </ns2:getDevicesAroundResponse>
   </S:Body>
</S:Envelope>
```

Listing 7.1: getDevicesAround SOAP Response

7.4 User Model Management Unit

The user model management unit, as described in the the design, is responsible for tracking and storing the user interaction pattern. In contrast to the traditional user models, the model of interest in interaction interoperability is basically the model of how the user would like to interact given a certain application, as set of available devices, at a given location at a certain time of the day.

Modeling the interaction pattern of the user was implemented through a set of relational tables in which a complete log of the user selections and inferences made in association to their context. The advantage of such an implementation again leverage the distributed nature, maintains consistency through referential integrity among the tables and queries are simply formulated through standard SQL. A result set of a query made against a specific user, application and context would be the user interaction pattern which could include already the case in hand and if not the closest match would be decided based on the number of equivalent factors.

Pulling out the user interaction pattern, or simply put, log, happens whenever a new mapping set are pulled by an application. The interaction interoperability web service offer this operation as a private function that are to be consumed within the application and unlike the context web service its not publicly listed and its not offered for other applications to consume.

7.5 Decision Making Unit

In this section, the prime function is described and that is the main operation of the framework. The Decision Making Unit is made up of four blocks; Active Rules Repository, Rule Engine, Action Dispatcher and Context Rule Generator.

Describing the procedure of asking for and getting a mapping set will clarify and highlight what is involved in going through these steps.

The procedure starts when a client application is started at the user side. Given the framework is built using SOA, all the web services and their operations can be listed and discovered and with the formally structured WSDL of the services, any client on any platform can access the framework services. A sample of the two commands needed to build .NET applications to be clients that can consume the framework web service are shown in listing 7.2

```
wsdl  http://localhost:40768/IIF/IIWebServiceService?WSDL

csc  /t:library  /out:IIWebServiceService.dll  IIWebServiceService.cs
                      /reference:System.Web.Services.dll  /optimize
```

Listing 7.2: Using WSDL to generate a DLL file for .NET application to reference to consume the framework web service

The application is configured with its assigned ID that is used in all correspondences with the framework services. When the application starts it pulls out the machine ID. The machine ID basically references the computer that is running the application. With the application not tied to any location tracking system, getting this ID of the machine would be the basis of pulling out the location of that machine and all users in proximity to that machine.

As the interaction interoperability framework gets the application startup sequence, the main web service consumes the context management web service through a number of operations. The location of the machine as stated, the users around the machine, the (tagged)devices available in proximity to the user and the machine. Getting these responses the user interaction pattern (historical records/logs) are pulled from the database on a stages. Queries start with all current factors in the search criteria hoping to find an exact match. If nothing is found with that exact match a series of queries goes with more relaxation going step by step again hoping for the closest match possible.

If an exact match in the user log is found the system sends off the exact mapping used previously again with an assumption that it would still be favored given the same conditions. If no exact match is found, the first hit to follow (with the query relaxation procedure) would be assumed to be the best decision and again the mapping decision would be formed as SOAP response as shown in listing 7.3

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getMappingResponse xmlns:ns2="http://IIF/">
      <MappingList>
      <TaskDeviceMappings>
      <TaskMapping><TaskID>1</TaskID><InputID>2</InputID></TaskMapping>
      <TaskMapping><TaskID>2</TaskID><InputID>3</InputID></TaskMapping>
      <TaskMapping><TaskID>3</TaskID><InputID>4</InputID></TaskMapping>
      <TaskMapping><TaskID>4</TaskID><InputID>5</InputID></TaskMapping>
      </TaskDeviceMappings>
      </MappingList>
    </ns2:getMappingResponse>
  </S:Body>
</S:Envelope>
```

Listing 7.3: Mapping SOAP Response

At the application side, upon receiving the mapping set, the application lets the appropriate input libraries handle the input devices events accordingly. Some events will be directly mapped, others would require thresholding, as an example in mapping an analog input to an binary task, and some events would be total ignored. Code snippet B.8 illustrates how the event handler in the application adapter or that could be built by the application developer looks like.

Back again to the design of the framework, there remains one functionality that should be added and that is the dynamic rule generation. The framework relies on the application here to *report back* any user **intervention** to the input configurations. This intervention is considered as the incident of an *unfavored* mapping occurrence. Obviously, the application would allow the intervention of the user (and that is greatly supported by previous work contrasting adaptive interfaces that do not allow this) but still it sends back to the framework the *more favored* mapping set as selected by the user.

Receiving a user intervention report would result in a two step procedure. The first step is to query the user log for an exact match of factors (context + application + machine). If a match is found this means that the there is a misleading record or precisely a mapping that is not favored anymore. The action taken then is to delete that record from the user set of interactions. The next step that follow in the procedure, which would happen in both cases if a match is found or not, and that is the insertion of a new log record for the user that would result in replicating exactly the intervention just made at the application side.

From the conceptual point of view what happened is to query the user log reflecting the pattern of interaction, based on the context rules and user preferences the mappings are sent to the application and for feedback, as stated, interventions cause the framework to update the *rules* to reflect that *new* preference. Implementation wise, the interaction pattern and user context rules are database records that are queried according to the satisfying foreign keys of a certain context (location, activity, identity and time) and the favored device would be the result of the query.

The query parameters are basically the rule conditions and the query results (if any) would be the action(s) to be taken.

For updates based on user intervention, the framework web services use SQL manipulations updating the respective records accordingly. The old rules mapping to records are updated to be less favored and the new rule is a manipulation of the favored record that would reflect the conditions present and the action that *should* be taken given these conditions.

7.5.1 Incorporating devices and applications within the framework

Based on the scenarios presented in Chapter 1 and the explanation in Section §6.3.6, this section describes detailed procedures to incorporate devices within the framework in developing applications and in supporting legacy applications.

Incorporating a device in *developing* an application involves the following points:

- Use a supported, application accessible connection like USB, Bluetooth or others.
- Get operating system supporting driver that would enable low level communication (protocol and synchronization)
- Get supporting development library for the development platform (e.g. XNA for Xbox, MS Wii Managed Library for Wii).
- Create an event handler for the device in the application.
- Define the device and its primitives for the framework.
- Task execution mapping to events is pulled from the framework web services through automatic web service client proxy generation.

Incorporating a device to support legacy applications involves the following points:

- Use a supported, accessible connection like USB, Bluetooth or others.
- Get operating system supporting driver that would enable low level communication (protocol and synchronization)
- Get a device emulator like GlovePIE for Windows (http://glovepie.org/glovepie.php) and ControllerMate for Mac (http://www.orderedbytes.com/controllermate/).
- Define the device and its primitives for the framework.
- The emulator task execution mapping is pulled from the framework web services as an XML map.

Incorporating an application within the framework involves the following points:

- Define the application, version and platform.

- Define the tasks of the applications.
- For each task, define class of task (selection, manipulation, navigation, symbolic input).
- Define task dimensions and dimensions attributes.
- Define attributes including is it integral/separable, should be sticky or not, binary/analog, discrete/continuous, range of values, intervals/increments.
- Create a Web Services client proxy referencing the framework WSs WSDLs.
- Consume the web services in the application to get the context and mappings needed for available devices.

7.6 Preliminary Implementations

As a proof-of-concept, I have built three different implementations Again to stress our standing, we are focusing on the concept of building a framework that would enable mixing and matching input devices, interaction techniques and applications. I am not concerned with a specific input device, or a specific interaction technique or effectiveness. However, I am concerned about the viability of a supporting architecture that could handle interaction interoperability in DVEs with a massive number of users.

Details of the four cases were presented in [Ahmed et al., 2008a,Ahmed et al., 2008b,Ahmed et al., 2008c].

7.6.1 XNA Implementation

The first implementation was a game developed using Microsoft's XNA Game Studio [xna, 2010]. The game was hard coded to support the Wii Controller in a number of interaction techniques, using the DPad, using the joystick of the Nunchuk extension for the Wii Controller and using the accelerometer. Another supported device was the gamepad of Microsoft, the XBox 360 Controller, supporting also a number of interaction techniques. Aside from these nontraditional devices, the keyboard could also be used and again with the flexibility of choosing to play using the right oriented arrow keys or the left oriented "WASD" keys.

The developed game featured a knight who was controlled by the user to move forward in an attack mode, to turn left and turn right. There were randomly moving and generated goblins that were supposed to be chased by the knight and destroyed using a frontal collision. If a moving goblin has a non-frontal collision with the knight, that reduce the knight's health. To finish the game the user should destroy all the goblins moving around.

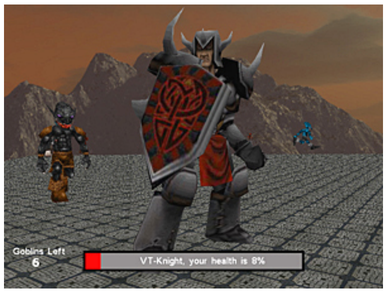
The game was a single player game to test for the acceptance of the concept of interaction interoperability. Ten test subjects were asked to try playing using the different available input devices. Once users are accustomed to all, they were asked to select one device to play a timed game. At the end, a questionnaire was given asking about their first and second favorite controllers and their quantitative opinion of interaction interoperability.

The results collected, in addition to the feedback taken through observation of the subjects, and the questionnaire they filled, showed significant acceptance of the concept. Results validated our hypothesis that no single input device can be considered the best for all users and for all tasks. There was no single device dominantly favored and there was no significant time difference between users using different device. Figure 7.1 shows some screens from the game and a student playing during the study.

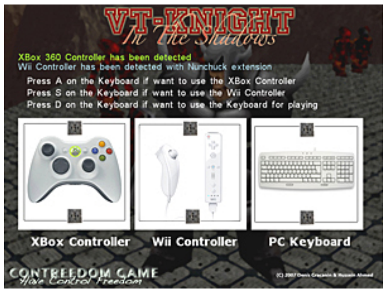
7.6.2 Legacy DVEs Support

The second implementation supports *legacy* DVEs with client applications unaware of the framework. The selected sample DVE was MPK20 [MPK, 2010] created by Sun Microsystems using the Project Wonderland Toolkit.

The client wrapper was needed to support a number of nontraditional input devices which

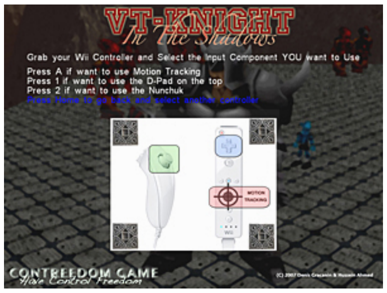


(a) Gameplay screen snapshot.



XBox 360 Controller has been detected
Wii Controller has been detected with Nunchuck extension
Press A on the Keyboard if want to use the XBox Controller
Press S on the Keyboard if want to use the Wii Controller
Press D on the Keyboard if want to use the Keyboard for playing

(b) Input device selection.



Grab your Wii Controller and Select the Input Component YOU want to Use
Press A if want to use Motion Tracking
Press 1 if want to use the D-Pad on the top
Press 2 if want to use the Nunchuk
Press Home to go back and select another controller

(c) Wii remote controller interaction techniques.

(d) A student playing the game using the accelerometer of the Wii controller.

Figure 7.1: A preliminary implementation of the proposed framework using a simple game with three tasks (move left, move right, and move forward) and three input devices (keyboard, Wii remote controller, and XBox 360 controller).

are originally not supported by the DVE. The wrapper had the job of mapping between the Wii Controller and the XBox 360 controller to keyboard key presses for the DVE client application.

This implementation setup used one server running on one machine and a number of client applications connecting to the server via the traditional Ethernet 100Mbps computer local area network.

With this setup, we have tested having users connected through different machines collaborating in the same DVE, but each using a different input device with a totally different interaction technique.

This setup was feasibly achievable with the help of the client wrapper component running locally on each host to accomplish the emulation task using the mapping rules of every user and the available input device. Figure 7.2 shows a screen capture of the DVE in which the user could now navigate using any input device.

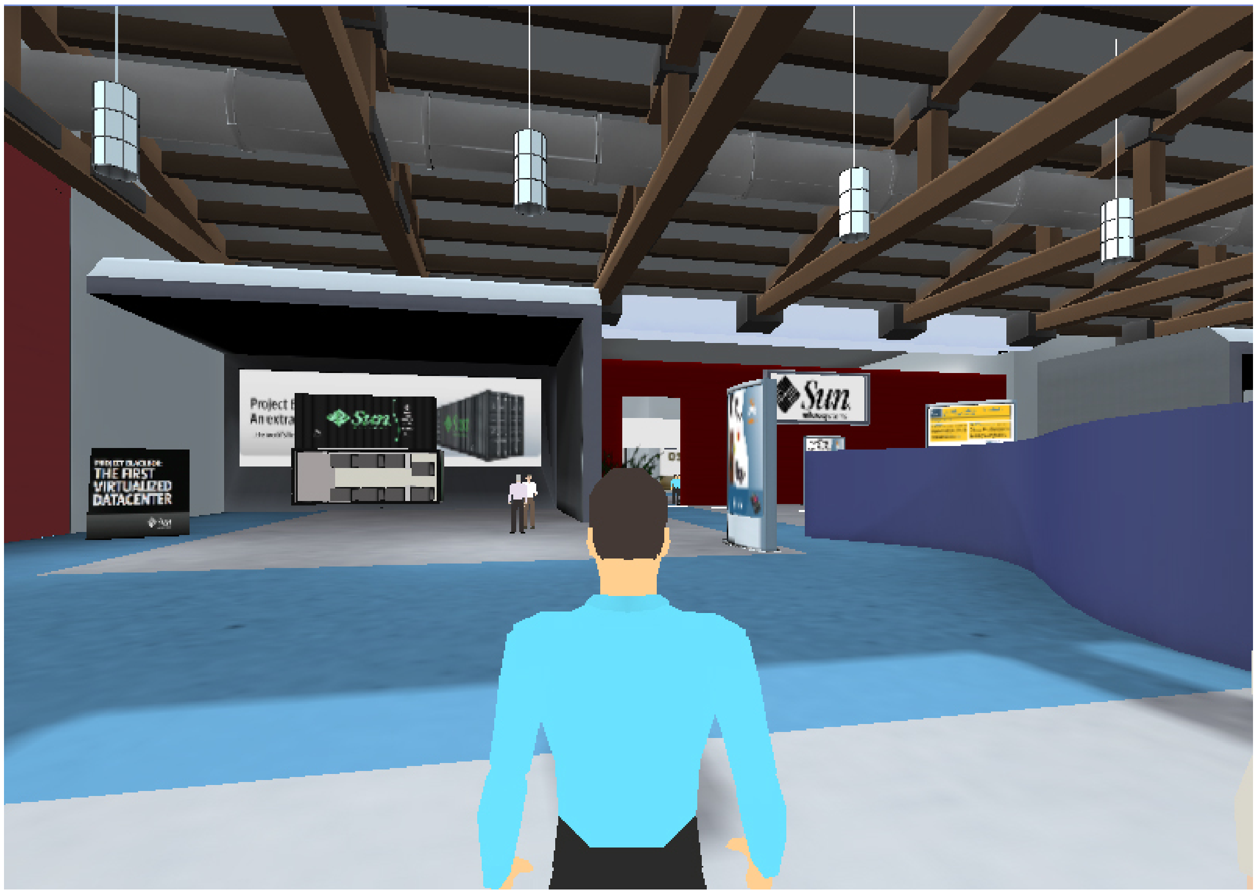


Figure 7.2: A screen capture of the MPK20 [MPK, 2010] in which the user can navigate using any input device with the help of the client-wrapper component.
(http://labs.oracle.com/projects/mc/mpk20.html) 2010 Public Domain

7.6.3 X3D Support

The third preliminary implementation, done early in designing the framework was focused towards the web. In this scenario, I supported *legacy* DVEs built using the currently evolving X3D standard and being unaware of the framework existence. The sample VE built using X3D was a social science education project named *"ON-A-SLANT VIRTUAL VILLAGE"* [OnA, 2009]. The interaction with the VE is completely left for the X3D browser to decide. Using the Flux browser [Flu, 2008], navigation was done moving the mouse up, down, left and right having the left button pressed.

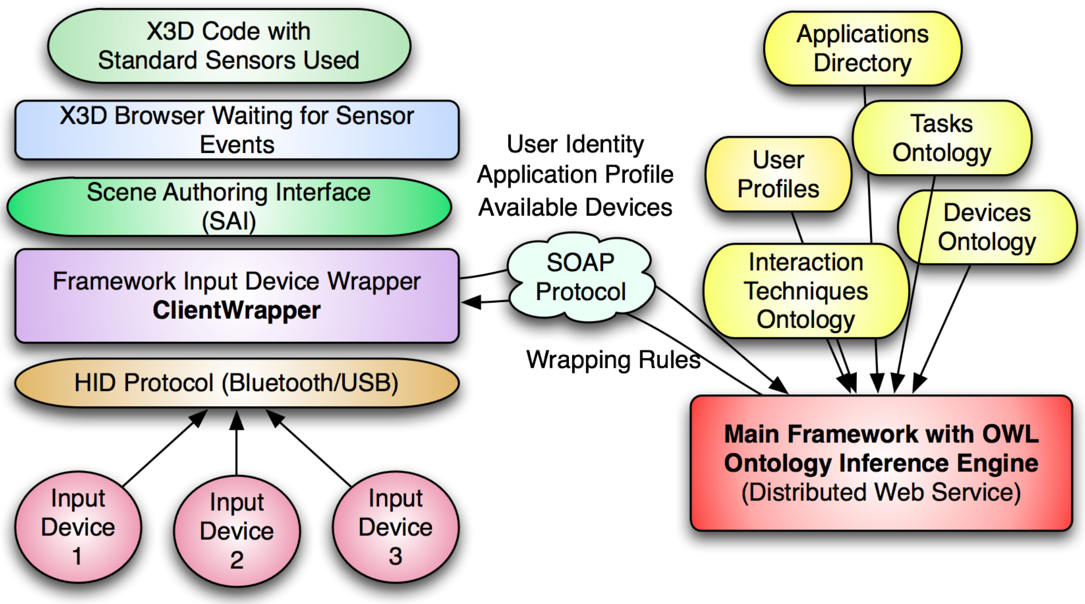

X3D Code with Standard Sensors Used
X3D Browser Waiting for Sensor Events
Scene Authoring Interface (SAI)
Framework Input Device Wrapper **ClientWrapper**
HID Protocol (Bluetooth/USB)
Input Device 1
Input Device 2
Input Device 3
User Identity
Application Profile
Available Devices
SOAP Protocol
Wrapping Rules
Applications Directory
Tasks Ontology
User Profiles
Devices Ontology
Interaction Techniques Ontology
Main Framework with OWL Ontology Inference Engine (Distributed Web Service)

Figure 7.3: Architecture for wrapping input devices and interaction techniques.

To realize the proposed approach, a Java web service was built to accept the user id, application id and available devices. It would reply back with a mapping file to be used by the client wrapper to execute the hardware emulation application that would then translate different devices and techniques to what is expected by the Flux X3D browser.

The architecture of the framework visualized again from the X3D perspective is illustrated in Figure 7.3.

Two input devices were tested and these were the Wii controller and the XBox controller. Different interaction techniques for using Wii like the D-Pad buttons, the accelerometer, and the analog stick of the Nunchuk were tested and proofed effective navigation of the environment as opposed to having to drag with the mouse in order to perform the same tasks.

In this chapter I have presented a number of reference implementations of the framework. I have discussed that the framework can be implemented in other various ways and I have justified the reasons behind going for all the implementation decisions taken.

In Chapter 8, evaluations of both the conceptual design and the implementations are presented. A complete case study implementation with all the framework components and functionalities has been created as a proof of concept and to assess the framework performance using a SOA testing tool to see how well it can scale and perform in coordination with other applications.

Chapter 8

Evaluation

The interaction interoperability framework presented in this dissertation contributes an approach of how to abstract input devices and application tasks. Previous work dealt only with building a taxonomy that classifies devices not describe them. With this fact, its required to evaluate the approach taken in abstracting devices to offer interoperability. Section §8.1 explains how the evaluation of the framework's extensibility to add and describe devices was made.

In Section §8.2, another side of evaluation is explained. That other side is evaluating interoperability and performance of the framework. Interaction interoperability is provided for DVEs that could scale to millions of users. With that in mind, the framework is expected to perform with the least delays (network, interpretation, inference) possible so not to defeat its purpose and hinder usability instead of achieving higher user satisfaction through context sensitive interaction interoperability.

In Section §8.3 a complete case study of Virginia Tech's LumenHaus is presented highlighting sample device configurations, application tasks and sample mappings. All the required functionality in the design have been implemented in the case study to aid as a proof of concept for my work and to assess the framework performance which is discussed in Section §8.3.1.

The last section in the chapter, Section §8.4, discusses the idea and viability of using the framework as an evaluation testbed for new reality based interactions that lacks fully fledged applications for user studies and can make good use of interaction interoperability to test innovative RBI interactions against legacy applications not built for them.

8.1 Flexibility and Extensibility

A major assessment factor to consider in evaluating the interaction interoperability framework developed is how is it extensible and flexible in terms of adding and manipulating records of new devices and new applications. With no qualitative measures available to asses these factors the evaluation from this perspective was to follow the evaluation of similar approaches of like [Buxton, 1983, Wang and Mankoff, 2002, Lipscomb and Pique, 1993, Jacob et al., 1994, Bowman et al., 2005, Jacob, 1997, Carter et al., 2006, Russell M. Taylor et al., 2001, Bleser and Sibert, 1990, Jacob and Sibert, 1992, Card et al., 1990]. Evaluation of taxonomies and classification criteria like these of Buxton [Buxton, 1983] and Jacob [Jacob et al., 1994] were mostly proved to be satisfactory by fitting a number of devices into their taxonomy and contrasting theirs to previous work.

With the interaction interoperability framework, the idea of having input devices and application tasks in an ontology in contrast to a taxonomy is novel. Despite the significance of having a novel design the challenge becomes then in evaluating the proposed solution.

Similar to the previous work in the area of device abstraction, mainly in classification, the evaluation was based on trying a number of cases, including complicated and novel ones in order to prove the viability of the proposed solution. Following that same approach, a number of devices like the XBox Controller, Wii Mote, Play Station Controller, Wii Fit Balance Board and a flight controller joystick are tested to prove how flexible and extensible the framework is in abstracting all sort of input devices with ease while maintaining accuracy and consistency. Figure **??** shows XBox and Wii Mote descriptions according to the framework.

In order to abstract an input device the following attributes should be defined (from Section §6.3.1):

1. Number of controllable dimensions
2. Unbounded range versus bounded range of values.
3. Domain range of each of the dimensions.
4. The perceptual structure (integrality versus separability).
5. Rotational versus translational; nature of movement.
6. Volatile or non-volatile; keeping the mechanical position and corresponding output value when released or not (spring loaded or not).
7. What is being sensed (position, motion or force).

Creating a device object for the XBox Controller it would be as follows:

First, given the controller is a composite device it would be created as an object of the class “Composite Device” which is basically an aggregation of an array of devices. These devices can be composite devices themselves or just primitive devices. The parameters specified would then describe each of the device primitives. Taking two sample primitives, the left analog stick and the DPad it described below how can they easily be described.

Left Analog Stick Definition:

Dimensions Two, X and Y.

Continuous/Binary X and Y are continuous (real values)

Unbounded/Bounded Bounded

Dimensions domain range X (-0.5:+0.5) , Y (-0.5:+0.5).

Perceptual Structure Integral (moving the stick X and Y change together).

Rotational/Translational Translational (no rotation axis)

Volatility Volatile (spring loaded)

Sensed Position.

DPad Definition: The DPad is a composite device which is made up of four primitive devices, basically four binary buttons. Defining the DPad then, same as its parent controller would be through defining its primitives.

DPad Button Definition:

Dimensions One.

Continuous/Binary Binary.

Unbounded/Bounded Bounded.

Dimensions domain range X (0:1).

Perceptual Structure N/A (only one dimension).

Rotational/Translational Translational (still applicable)

Volatility Volatile (spring loaded in contrast to an nonvolatile switch)

Sensed Position (pressed or not).

To evaluate other input devices that do have novel interactions like the Wii Mote. Again similar to the XBox controller, the Wii Mote would be objects instances of the class composite device that is made up of other composite devices and primitive ones. **Wii Mote Accelerometer Definition:**

Dimensions X, Y and Z.

Continuous/Binary Continuous.

Unbounded/Bounded Bounded.

Dimensions domain range X, Y and Z (-1:+1).

Perceptual Structure Integral (moving the remote will affect all three).

Rotational/Translational Rotational (there is a rotational axis for each dimension)

Volatility Volatile (it won't maintain the value)

Sensed Motion (not position sensed, only motion).

As noted in the literature review chapter, some taxonomies failed behind with the inability to add certain input devices with certain unique qualities covered by a taxonomy and not the others.

The framework abstraction methodology is based upon all the major proven descriptors used through literature. Abstraction is built upon grounded and proven classifiers including separability, dimension count, range, physical characteristics and the pragmatics of the device. Based on classical classifiers and using a varied number of sample descriptions of different devices and application tasks in the different case studies, I concluded that given the list of devices tested and previous test cases used through literature are perfectly fitted for this approach. Therefore the approach taken in abstracting the devices and having an ontology enforcing trivial rules proves to be a viable and effective way.

Regarding application tasks, again as it was explained in Chapter 6, application tasks are to be defined in a similar fashion to the tasks and that what makes the mapping straight forward and yet maintaining practicality and ease of defining and manipulating applications. To evaluate extensibility in terms of supporting applications the framework, as stated previously, application profiles have the following attributes:

- Task number of dimensions.
- Range of values expected for each of the dimensions.
- Range of values binary or continuous.

- Is the range of values expected bounded or not.
- Value expected to be sticky or not (controller should maintain position).

In previous work, as stated in the related work chapter, profiling from the interaction perspective was mostly for the sake of porting an application interface from a platform to another. Evaluation of these systems was a qualitative evaluation based not even on the completeness of the definition but on how can that just fulfill the need to define the mappings.

To assess extensibility, any application is an instance of the general class “Application”. The application is then basically an aggregation of “Task” objects. Each of these task objects is then defined accordingly. To show extensibility, two different applications are described demonstrating how varied application domains can still be well profiled making it possible to map devices to applications.

To start, the sample case study application which is the LumenHaus (the solar house project of Virginia Tech), which was the proof of concept implementation, it had four tasks aside from simple menu navigation.

The house is designed with an idea of that, anything controllable around the house is a control point. So the four tasks were: select previous and select next control point, increase value and decrease value of point(or set/reset for binary nodes).

Go to Next Control Point

# dimensions One

Bounded Bounded

Range of values 0:1

Binary/Continuous Binary

Sticky No

Increase Value by an increment

# dimensions One

Bounded Bounded

Range of values 0:1

Binary/Continuous Binary

Sticky No

Its to be noticed that the value increase task is precisely named "Increase by an *Increment*" which means that its a binary task even if the node to be controlled is an analog node.

Another case considered is Second Life. In Second Life the tasks were to move forward, backward, turn right, turn left and interact with a command menu on screen with a pointer. The navigation tasks were simply mapped like the node control tasks. For the screen pointer interactions the task is described as follow:

Position Screen Pointer

# dimensions Two X & Y

Bounded Bounded

Range of values X(0:Max X) and Y (0:Max Y)

Binary/Continuous Continuous

Sticky Yes

Again with analogy to previous work basing evaluation on a set of cases, it is demonstrated how the interaction interoperability framework task abstraction is capable of handling varied tasks. The proof is that, exhaustively spanning what previous systems and taxonomies offer in classifying and describing tasks has been successfully described and classified using the attributes and class hierarchy of the framework.

The last point to evaluate in terms of the framework extensibility and flexibility is a crucial point and it is **how easy** it is to add and describe a device or a task. For the sake of platform interoperability and given the use of SOA, all objects and data exchanges use XML. This facilitates interoperability and improves usefulness.

The lists of devices and tasks previously stated as examples can obviously be converted trivially to XML. Each attribute that defines a device, device primitive or application task would map to an XML node with the corresponding value.

For the proof of concept implementation, additions were done using a wizard like interface that just basically filled in the value of each of the attributes one by one giving the user a set of options to select from where applicable. There is a limitation for the addition which is that the user should be familiar with computer interaction terminology starting simply from dimensions. The way to overcome this problem would be in devising an easier set of questions that do not ask direct questions that demand knowledge of the field.

Assessing the framework extensibility, there is a technical limitation right now. There is an assumption in place and that is the operating system support. For any device currently to interact with the framework through a client application there is dependability on the operating system to detect, configure and give access for the application to read from the device or the device to signal an event. In order to overcome this problem, a viable solution would be to have a hardware middle-ware that would eliminate the application dependability on the operating system. The ideal case in which the framework could then be fully utilized would be having any input device available for the application with nothing needed to be configured or the application recoded somehow to support or include a certain library that would facilitate communication to and from the device. An example of this drawback was in supporting Wii on the Windows application. Without getting the Wii managed library for C#, it would not have been possible to interface with a Wii controller.

To conclude the evaluation of extensibility and flexibility, for both devices and applications, the framework descriptors include the majority of the descriptors suggested by the body of literature aggregated. Given that flexibility and clear XML definitions based on a set of defined XML nodes, it is possible to make a direct mapping between the interaction devices and tasks even using a simple set of rules. The rules can be coded even as a set of if-then statements of a reasonable size and still be able to handle any sort of possible combinations creating a clear mapping graph. Sample profiles and mappings are listed in details in the case study presented in Section §8.3.

In the following section, the framework evaluation from the systems perspective is presented, highlighting delays incurred, performance charts given different loads and test scenarios to see the overhead incurred by integrating the framework.

8.2 System Interoperability and Performance

In this section the framework is evaluated from the systems perspective. There are two sides of the evaluation to consider. The first is interoperability and the second is system performance.

In evaluating interoperability there are no metrics to assess but it would be through critical analysis of all communications, storage and information exchange.

Analyzing the different communication channels I will revisit again the framework components. The framework is made up of an input device management unit, an application management unit, a user management unit and a context management unit, and finally the decision making unit. All five units are built as web services that are accessible through SOAP requests. Some of these units, as described in Chapter 6, may consume others in order to do their own job, as an example is the decision making unit which is clearly a composite web service since it basically accomplish its job consuming all the four other units.

Given the use of SOA, it would not matter even if each of the framework units was implemented in a different language, on a different platform or running on a different server. With this architecture in place and knowing that communication between units is done through SOAP, reflects high interoperability in both internal and external communication. With the system loosely coupled, technology was not a barrier when developing the reference implementation (server or client side).

With all the data and information exchange between the framework units done via discoverable web services through XML SOAP messages, this renders the framework highly interoperable. Any application can pull the WSDL file of the advertised services and create a proxy with all operations offered in any language. This means that the framework is usable by all sort of applications running on mobile devices, desktop computers with any OS, gaming consoles or any other device that can have access to the Internet and able to form and parse SOAP messages. A sample of the direct integration based on the advertised WSDL is shown in Section §7.5.

On the other hand, regarding performance, in the early phases of the project performance was assessed using an open source tool named PushToTest [Pus, 2010]. Another tool, SOAPSonar [SOA, 2010] which has more functionality and reporting features was used in evaluating the last reference implementation which was the LumenHaus (solar house) case study discussed in Section §8.3.

In evaluating the system performance a test suite was recorded with the client application running and going through a number of steps that demand consuming the framework web services that in their turn consume each others as well. In evaluating a typical user, the LumenHaus controller (discussed in §8.3) which does physically control the house nodes was used as usual. In stress testing, doing so would have affected the results. In a typical scenario, with a number of houses are to be managed and supported, each of these houses should have its own controller and it should never happen to have one physical controller to control all managed houses.

In doing stress testing then, the physical controller web services were modified so not to make an *actual* call to the hardware controller and an empty loop was added instead to mimic the typical delay of a single loaded controller as it was measured in the one user scenario test run.

In testing, all the framework servers and the database server were running on a single machine. Since the delays reported are directly influenced by the machine running the servers, below is the technical specifications of the computer:

Operating System Windows 7 Professional

Processor 2.4 GHz Intel Core 2 Duo

Number of Processors 1

Number of Cores 2

Memory 2 GB 667 MHz DDR2 SDRAM

Bus Speed 800 MHz

The SOAP tests and the test suites created in addition to the reported results are all under the case study section, Section §8.3. In order to have a reliable evaluation of the framework, the case study created based on Project LumenHaus had all the framework components in place including context elicitation, rules management, rules generation (given a user intervention) with all functionalities working and demonstration of the working physical control was done against the actual house controller.

8.3 Complete Case Study

As a case study that utilizes all the framework components laid out through the dissertation, an application was created that would be used in controlling a computerized solar house. The house is named LumenHaus and it was the house Virginia Tech built to compete in the 2009 Department of Energy Solar Decathlon. A virtual version of the house was created in Second Life.

The application created had a Windows desktop application that takes advantage of Interaction Interoperability by consuming the framework web services. The application consuming the services offered was able to detect the user accessing the system eliminating the need for a typical user login. Again, through the context-awareness support web services, the devices around the user were known.

Through the interaction and context web services the desktop application gets a list of devices (subset of the ones available) most favored given the *current context*. Starting the application, the device that is on the top of the list of favorites is automatically elected to be the device to be used.

Devices available for the user were the XBox controller, Wii controller, and the keyboard. Using any of these devices in any way that is most preferred by the user, the user can then manipulate any control point in the house.

Figure 8.2 illustrates the two ends for the device to application mapping needed. On the left hand side, the four main tasks afforded are listed with their attributes and on the right hand side some of the devices (complex and their primitives) are shown again with their attributes that describes them. According to matching pairs and direct translation of analog to digital (achieved by splitting up the range and setting thresholds given that vice versa is not possible), a mapping set is concluded given the user preferences, available devices and current context.

Figures 8.3(a) and 8.3(b) illustrate two possible mappings going down from an analog range to a limited binary one and another straight forward showing binary-binary mapping.

Back to the client side, for the application to map whatever input its getting through the selected device, it makes a SOAP request to the framework web services to pull out the necessary mapping matrix like the one shown in Listing 7.3

Given the mapping matrix the client application gets from the framework backend, the application listens to certain events only and will take the corresponding action as it is precisely *dictated* by the framework. Any action performed then on the client side was passed to the LumenHaus controller server that executes that action on the real house. Figure 8.1 illustrates the architecture of the case study taking advantage of the framework web services to bridge between the physical and virtual planes.

In bridging the real and virtual planes, any user action performed through the Windows desktop application using any input device was replicated to the house in Second Life. As an example, moving a sliding door through the interface the door would then be physically moved in

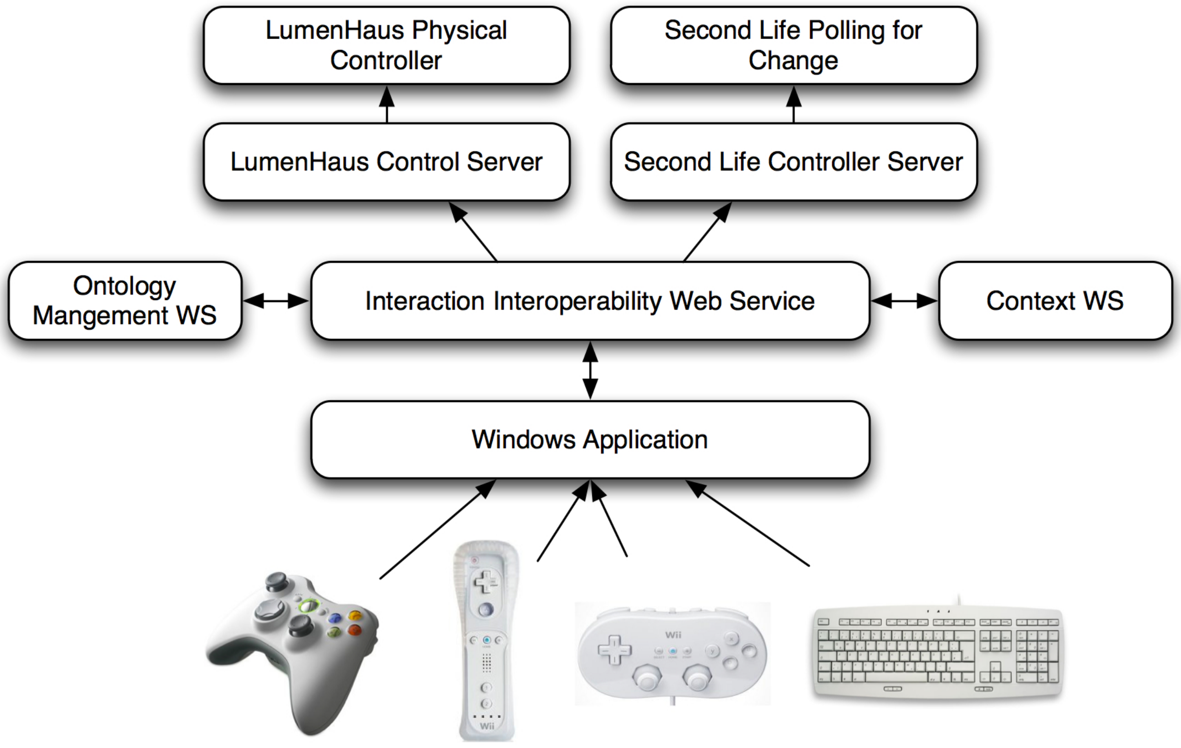

LumenHaus Physical Controller
Second Life Polling for Change
LumenHaus Control Server
Second Life Controller Server
Ontology Mangement WS
Interaction Interoperability Web Service
Context WS
Windows Application

Figure 8.1: LumenHaus case study utilizing Interaction Interoperability.

real as well as the virtual door in house model in Second LIfe would reflect that same movement, thus maintaining consistency between the physical and virtual houses. In dealing with the Second Life grid there were two approaches, first is to push an object attributes update to Second Life to actually control an object. The second approach was to assign a Linden script (Second Life scripting language) to the object to be controlled and that script had an always running timer set to execute a function at a given interval and that function is to pull from a server the new object update needed. In implementation, given the later approach is straight forward I decided to go for it for the proof-of-concept but for commercial scale deployment pushing would yield much better performance given that the framework will be pushing updates *if* desired in contrast to allows getting polled by the Second Life servers for no reason.

There could be a case when the retrieved list of favorite devices given the current context of that specific user does not reflect the user preferences, at least not any more. In that case the user has the option of **intervention**. Intervention basically reflects selecting another device from the available devices other that the one that was automatically elected. These interventions, as explained in Section §4.1, are considered the feedback loop that feeds the framework. When a user intervention occurs, the client sends an intervention signal which signals that the rule that fired setting the wrong device is not valid anymore and implying that given the *current context*, *user* and *application* the new device selected by the user should be the most favored.

As a response to an intervention request or signal, the framework services will *de*-mote the device that was selected for this context and *pre*-mote the new one that the user manually decided to use. On a new login or primarily another request for that same combination, the device that got promoted will be the framework choice reflecting the user change of likes and dislikes. Thus, demoting and prompting devices would be the basis for the new rules pulled out.

Figure 8.8 shows screen capture of the LumenHaus .NET application controller that handles three different input devices; the keyboard, XBox controller and Wii Mote. No login screen is needed, the user with access to the computer in closest proximity is assumed to be automatically the one in control.

8.3.1 Case Study Evaluation

In evaluating the framework case study, a memory profiler and SOAPSonar were used to analyze the inherent delays and response times. The primary test made for SOAP web services is named as a WSDL Report Card. The goal is to check for compliance with industry standards and business to business practices and polices. The test is based on 55 criteria rules and the result is a school-like report shown in Figure 8.4.

The testing process starts with the tool profiling the WSDL of the web service and accordingly SOAP test calls are to be created for the different operations supported. Figure 8.5(a) shows a sample SOAP test call for the mapping operation. In creating the test call, the tool interface offers a variety of ways to input testing values for the expected parameters either fixed values, from a CSV or even a database.

After creating all the test calls, to determine if a call did actually succeed or fail, it should be stated what qualifies as a successful response. There are different criteria available to tweak and consider for evaluation and that would basically depend on the operation being called and what should be considered a success or a failure for that specific call. As an example, a certain operation could be considered a failure if the response exceeded a certain threshold of milliseconds. Another operation can disregard the time constraint, thus in this case regardless of the delay it can still be considered successful if other criteria like a response match is achieved.

Figure 8.5(b) illustrates setting a success criteria for the web service operation setPoint, which updates the value of a certain control point, to be successful if the XML response match the expectation set which would be a response back with the same ID and the requested value. If the XML response is not precisely the same then that response would be considered a failure.

Having a complete, comprehensive set of SOAP test calls for the different operations offered by the web service, the next step is to create a test suite. The test suite basically reflects a test scenario. In building a test suite, a comprehensive yet practical scenario should be considered that still though should be aggressive. A number of test suites were created to test the framework. Some test suites were to aggressively push for a high frequency of mapping requests, others were created to test the limits of the controller liaison operations and others to contrast between the framework actual processing and other irrelevant processing inherent by other interfaces like Second Life and the LumenHaus controller.

Figure 8.6 shows a screen capture from SOAPSonar test suite builder interface. On the left hand side are the SOAP tests created in the previous step with each having a defined set of input values and the corresponding success criteria defined. On the right hand side, a scenario is created by dragging SOAP tests in a certain combination and sequence. A typical testing scenario was as follows:

1. Request list of control points.

2. Request interaction mappings
3. Update then Get Control Point — Repeated 50 times
4. Context Change (devices available/user in proximity/any of contextual dimensions)
5. Request interaction mappings
6. Request list of control points.
7. Update then Get Control Point — Repeated 50 times
8. Context Change (devices available/user in proximity/any of contextual dimensions)
9. Request interaction mappings
10. Request list of control points.

Repetitions of contextual changes, mandating a new request interaction mappings and again a sequence of sets and gets were deliberately put in place to see the effect of such a high change frequency despite that such an aggressive won't be a typical working scenario. The results of executing the scenario with response time as the metric, Figure 8.7 shows the performance iterating through the scenario 100 times.

A question to raise here is what would be the number of simultaneous users. Given the services are all based on SOAP protocol and even the rule engine is working in a stateless fashion, having simultaneous users would not make a difference from the session handling perspective. That means 1,000 SOAP requests are still going to be treated identically if they were requested by a single user or multiple users. However, still the performance of simultaneous *calls*(in contrast to users) is to be considered.

Analyzing the response chart shown in Figure 8.7, the high peaks reflect the SOAP *request list of control points* calls. The dashed line overlaid over the chart highlights the range of response time for all other calls including *set point*, *get point*, and even the major call of *get mappings*. It is clear from the representative chart shown (and others for different scenarios) that the delays introduced by the framework were not significant to impact the overall performance of the system utilizing the framework services. It is evident that the delays inherent by other interfaces, mainly the house controller in order to take physical action, do have much more impact.

Based on this case study it can be clearly concluded that the interaction interoperability framework should be considered as a viable solution in bridging virtual and physical world given it does not have any negative impact on performance and thus on usability being introduced to offer its afforded services. Systems that can tolerate delays in the range of 100ms to 300ms can fully utilize the framework services without resulting on any perceived extra delays.

System on the other hand that are to operate within a much more strict response times (less than 100ms) are to expect perceived delays in taking advantage of the framework services.

A great advantage aside from performance metrics was in integrating the framework with various interfaces. There were no changes needed to any of the application ends, the physical controller and SecondLife to accommodate interacting with the framework. A loosely coupled architecture that is totally interoperable through the exchange of strictly XML SOAP messages has a great advantage by allowing different platforms and programming languages to interact and take full advantage of the framework discoverable services.

8.4 Discussion

Reality Based Interaction (RBI) as defined and presented by Jacob et al. in [Jacob et al., 2008] is a post-WIMP (window, icon, menu, pointing device) interaction that builds on the user pre-existing knowledge of every-day, non-digital life. As Jacob et al. [Jacob et al., 2008] stated, there are four RBI themes; Naive physics, Body Awareness and Skills, Environment Awareness and Skills and finally Social Awareness and Skills. Utilizing these themes in designing interactions as suggested, surveyed and proved again in [Jacob et al., 2008] elevates usability dramatically given how they closely resemble interaction in the users' real, non-digital world.

An evaluation testbed that is characterized by the following points is an ideal candidate to evaluate newly developed RBIs:

- Practically, supporting any input device regardless of physical structure, characteristics, limitations, dimensions and dimension ranges.
- Interfacing to legacy framework-unaware applications making it possible to tryout input devices not originally supported and thus opening up to test devices against any application or application domain of interest.
- Assessing the user context in real-time and not only working on a set of predefined rules by dynamically generated new ones on demand makes the RBI themes Environmental and Social awareness inherent in the evaluation to be conducted.

Given how post-WIMP interfaces use one of the four themes, nowadays input devices are moving towards this trend with so creative and intuitive ideas coming up everyday. In order to test such innovations what typically happens is that a dedicated application is built from the ground up utilizing what the input device or interaction technique have to offer. Given this case, evaluations are done on incomplete, non-realistic application failing to reflect how would this device or technique perform in real life applications. Even worse, if the decision was to go for whats being introduced applications have to be re-built all over hard coding the support for the new interaction. Interaction Interoperability solves these problems.

The major gola of the proposed framework is to be a testbed to evaluate input devices and interaction technique to control all sort of applications. That provides the researchers and developers the opportunity to assess how would the proposed solution perform with a variety of applications. There is no need to develop a dedicated application prototype built just for testing and far from being complete.

As a testbed, offering Interaction Interoperability should not be only be for applications built from ground up with the framework services put into consideration. The objective is to allow interoperability for legacy applications as well, giving the possibility to experience RBI for those applications that are already out there. There is a large user base lacking such great interactions due to the interface based on traditional WIMP interactions.

In this chapter the framework was evaluated from a number of dimensions; flexibility, extensibility, interoperability, performance and functionality. A number of input devices and application tasks were tested. An to have actual evaluation a complete case study was developed that again aided in testing and evaluating the framework functionality and performance. From the test results collected, it was found that the framework does not incur major delays that would negatively impact the application, hindering responsiveness, usability and thus defeating the whole purpose of the framework. The framework web services being fully implemented aided in proving the concepts laid down early in the dissertation including device abstract, application tasks abstraction, communication interoperability, mapping deductions, scalable and realistic context-awareness and the personalization goal that was set forward with the possibility of intervention.

Analyzing the performance charts, with a delay of around 200ms, I do believe that integrating the framework with other DVEs should not be considered as a significant overhead given the other much bigger delays that result from 3D intensive rendering, content transmission and as shown on the charts for the physical controller, the delay caused by the control part was around 20 times more than that of the framework.

Referencing the research problems listed in Chapter 1, scale, interaction and context going through the work done, case studies, proof of concept implementation as well as performance evaluation it can be concluded that the objectives have been satisfied and the concept and design put overcome the challenges stated previously. Sections §8.4.1, §8.4.2 and §8.4.3 revisits the challenges stated noting how the framework I have designed and developed copes with these challenges in order to satisfy the goal and objective of offering transparent, context-sensitive interaction interoperability.

8.4.1 Scale

Below are the main research challenges regarding scalability and system interoperability, revisited again highlighting how in designing the framework I have approached and solved these challenges.

- *Coping with highly loaded DVEs without compromising usability.*
  Designed as a distributed system with loosely coupled components that consume each other eliminates any bottle neck, single points of failure and takes advantage of all the positive points about Service Centric Architectures. Based on *stateless* calls there are no sessions to track and handle and making it grid ready and with location awareness, geographic load balancing can also be utilized.

- *Communication protocol to use between the framework and the DVEs.*
  The framework front end is a set of discoverable web services with an advertised WSDL file. Any platform or programming language can create automatic delegation to the operations supported by the web services making it possible to integrate any system or DVE with the framework and utilize all its functionalities. As a proof of concept the LumenHaus case study had a Microsoft .NET client that was able to consume the advertised web services starting forward taking advantage transparently of context-awareness and interaction interoperability.

- *Message format and structure for platform interoperability.*
  Messages expected to be received by the framework and those who are dispatched by the framework are SOAP messages which are totally interoperable based on XML. Aside from the SOAP envelope, the message content is also an XML document tree that can be parsed using any automated XML parser offered through any language.

- *Have the services offered discoverable to all the population hosts.*
  In contrary to the traditional HTTP client/server architecture and the distributed approach of peer-to-peer, the framework is based upon the standard Serve Centric Architecture that mandates having a discoverable WSDL file accessible through a Universal Description, Discovery and Integration (UDDI) directory.

- *Push commands to applications (CVEs) in a timely manner.*

  The design as laid in Chapter 6 utilizes the XMPP protocol which is widely adopted for instant messaging and based on such protocol it is possible to push notifications and configuration commands given a change of context without a reconfiguration request from the client.

8.4.2 Interaction

Below are the main research challenges regarding interactions, revisited again highlighting how in designing the framework I have approached and solved these challenges.

- *Classify and define input devices and interaction tasks.*

  Based on major and a significant body of literature, various taxonomies and classifiers have been studied. In contrast to just classifying devices, since the goal of the framework is to actually map these devices to tasks fine description through detailed attributes were added. A comprehensive summation of attributes that were used in previous research was created to reflect device/task class attributes. Devices and tasks are classified according to a hierarchy with relations including inheritance and aggregation and as a new class or just an instance a set of attributes are assigned to reflect fine details that would be used in matching pairs together.

- *Allow manipulations, additions and omissions of devices, techniques and tasks while maintaining consistency.*

  The framework knowledge base is based on an object-oriented design and ontology rules and constraints and even at the database records, referential rules are put in place relating fields to their parent tables. With these technologies and consistency measure put in place it is not possible to assign a device or a task to a wrong class and define it after and with rules and constraints and integrity rules consistency is enforced at all different levels in an effort to eliminate the possibilities of conflicts or inconsistencies.

- *Untie devices from techniques and tasks and loose the hard-coded design time principle.*

Utilizing the framework web services and as shown in the case studies presented, the client application no more have to have a set of predefined set of mappings between input events and tasks. The application just sends in a mapping request for a certain user identity and given the current context of the user a set of XML mapping tree is sent back for the application to associate events to tasks with.

- *Offer on-the-fly mapping between devices and interaction tasks given large scale.*
  As explained in Section §8.4.1, the framework is not based on a desktop application nor the classical session based client/server architecture, given loose coupling and distributed web services, mapping requests could be handled in a timely manner and if a DVE scales up to a grid based level the framework can scale up to that same level to cope with that much expected load. Based on the evaluations presented earlier in the chapter it was proved that the delays incurred by integrating the framework were minimal and could be considered negligible compared to the typical DVE delays inherent int he environment already.

- *Profile users based on their interaction choices and current context.*
  The flow of the framework starts with a user accessing a certain application running on a specific machine at a certain time of the day at a certain location. The interaction picked by the user at the point would be part of the user pattern of interaction. This pattern as it build up would be basis of a set of context rules specific for that user. On an occurrence of that same context again, this set of rules would fire if the conditions are satisfied commanding reconfiguration of the interface and thus achieving the personalization we opt for.

- *Manage all knowledge base repositories with no single point of failure.*
  With the repositories being translated to and from database records, utilizing a distributed database system that can run on a grid would mean that distribution is achieved with all problems already solved and handled by the engine and totally transparent to all other levels.

- *Profile applications to afford mapping user actions to application tasks.*
  Same case as the devices, application tasks again would be part of an application class which is somewhere in a the application classes hierarchy. Application tasks however have more

attributes than what would typically describe a task and these attributes resemble similar ones that describe input devices. One example would be *stickiness*. In devices, stickiness reflects if a controller is a spring-return or something that *sticks* to its place like a mouse, slider or a knob. Tasks will have that same attribute saying that volume as an example can either be controlled in increment using a spring return device or the actual value being set using a sticky device.

8.4.3 Context

Concerning the research challenges of context-awareness, below is a list of the challenges stated previously in Chapter 1 and descriptions how the framework tackles the challenge.

- *What dimensions to consider.*
  As stated in Chapter 4, the framework does not only takes into consideration the context of the user in the real plane but the virtual plane as well. he primary dimensions, identity, location, activity and time are assessed from both planes making up the user context.

- *Assess the different dimensions chosen for the real world, integrate, filter and assimilate collected data.*
  Collecting context of the real plane for a user as discussed would consume a lot of computing power being lost in pushing and polling on data. Making the framework modular to the extent of having separate complex web services, computation resources can be distributed in a smarter way and overloaded modules would still not negatively impact others. Having different levels of granularity and the semantic location of utmost priority filtration and aggregation could be delegated to modules and the web services level and abstracted, unified semantic location would be the one of influence in making decisions.

- *Assess the virtual world context given no actual physical sensors.* As described in Chapter 4, the context of the virtual plane is assessed and assimilated through feedback from the DVE as well as the activity map concept explained in the chapter. Identity aligns with real world, time is fed through the DVE as well as the semantic location. Activity is looked up from an activity map that states that for a given location and a given time frame the activity would be something specific.

- *Manage context rules and accommodate new rules.*

  With context rules being treated as database records in contrary to classical systems that loads a set of rules and work through them on an if-then basis this give much more leverage to scalability through high performance queries and distributed databases. Adding new rules is done transparently upon a new interaction log being detected. Whenever a context occurs again the records that satisfies the condition would be fed as an XML rule set to high performance rule engine to handle that limited working set of rules.

- *Know if a rule should no longer hold.*

  Applying the concept of the client feedback loop, this reflects that the framework actually listens to the user trying to monitor his/her satisfaction. Whenever a user is unsatisfied with the decision made, an intervention to change the interface would be sent to the framework instructing that the rules already there to be updated reflecting the user new preferences.

- *Assimilate all context data collected from the real and virtual world of the concurrent users.*

  As stated in the scale challenges, it would be impossible to support context awareness for multi-million users monitoring all the rules and firing satisfied ones. However, with the modular design of the framework *and* having the context rules pulled as a database query which is fed to a rule engine makes it feasible to handle millions of users.

- *Infer through the rules of all concurrent users.*

  Two cases are possible. First is to have clients poll for any context update and this would translate to a query being executed on every request. Second, for the case of pushing changes upon context switch the rule would be queried and that working set would be fed to a rule engine that can handle a vast amount of rules (yet limited since the user base rules are in the database and not in the rule engine).

The next chapter, Chapter 9, summarizes the related work, the presented concepts, the proposed design and the developed reference implementations. The key contributions are described and a set of possible future extensions suggested.

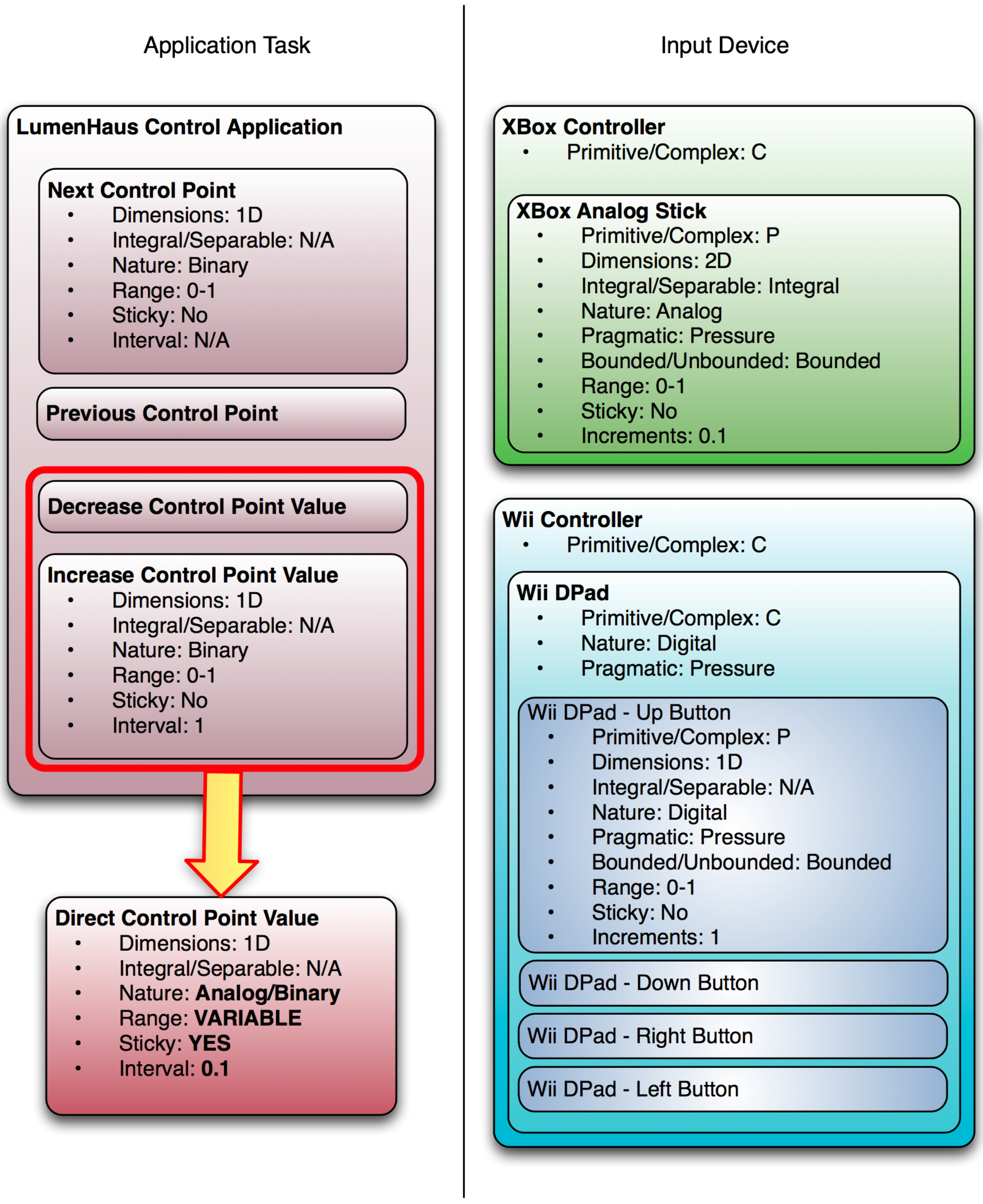
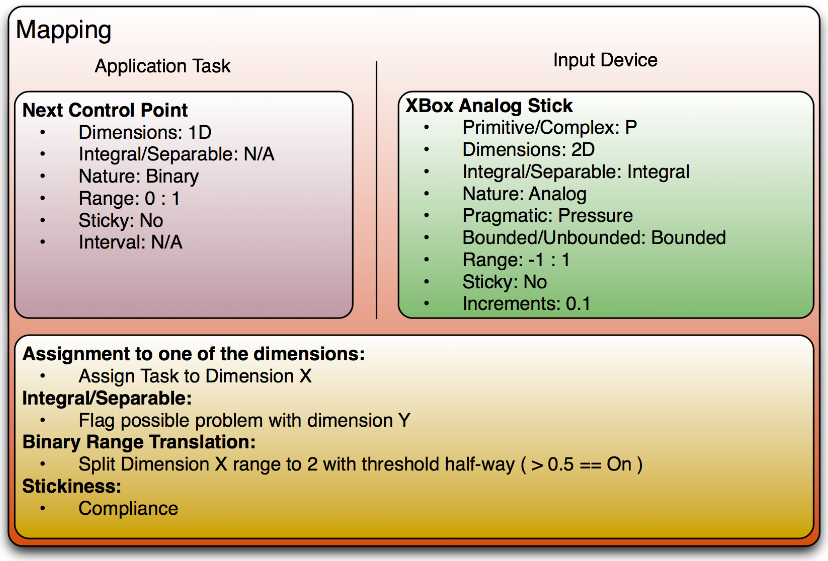


Figure 8.2: LumenHaus application tasks and input devices attributes.


Mapping

Application Task

Next Control Point
- Dimensions: 1D
- Integral/Separable: N/A
- Nature: Binary
- Range: 0 : 1
- Sticky: No
- Interval: N/A

Input Device

XBox Analog Stick
- Primitive/Complex: P
- Dimensions: 2D
- Integral/Separable: Integral
- Nature: Analog
- Pragmatic: Pressure
- Bounded/Unbounded: Bounded
- Range: -1 : 1
- Sticky: No
- Increments: 0.1

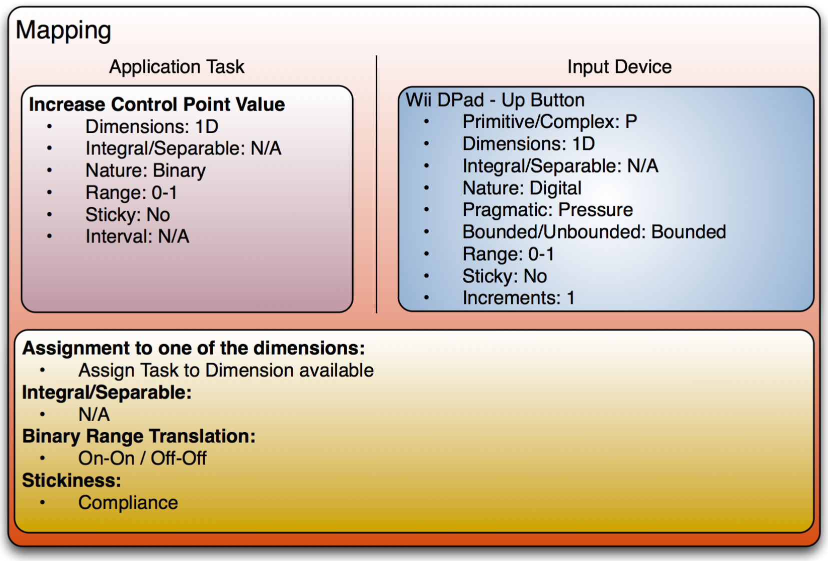
Assignment to one of the dimensions:
- Assign Task to Dimension X

Integral/Separable:
- Flag possible problem with dimension Y

Binary Range Translation:
- Split Dimension X range to 2 with threshold half-way (> 0.5 == On)

Stickiness:
- Compliance

(a) Sample mapping of control point selection using an analog input primitive.


Mapping

Application Task

Increase Control Point Value
- Dimensions: 1D
- Integral/Separable: N/A
- Nature: Binary
- Range: 0-1
- Sticky: No
- Interval: N/A

Input Device

Wii DPad - Up Button
- Primitive/Complex: P
- Dimensions: 1D
- Integral/Separable: N/A
- Nature: Digital
- Pragmatic: Pressure
- Bounded/Unbounded: Bounded
- Range: 0-1
- Sticky: No
- Increments: 1

Assignment to one of the dimensions:
- Assign Task to Dimension available

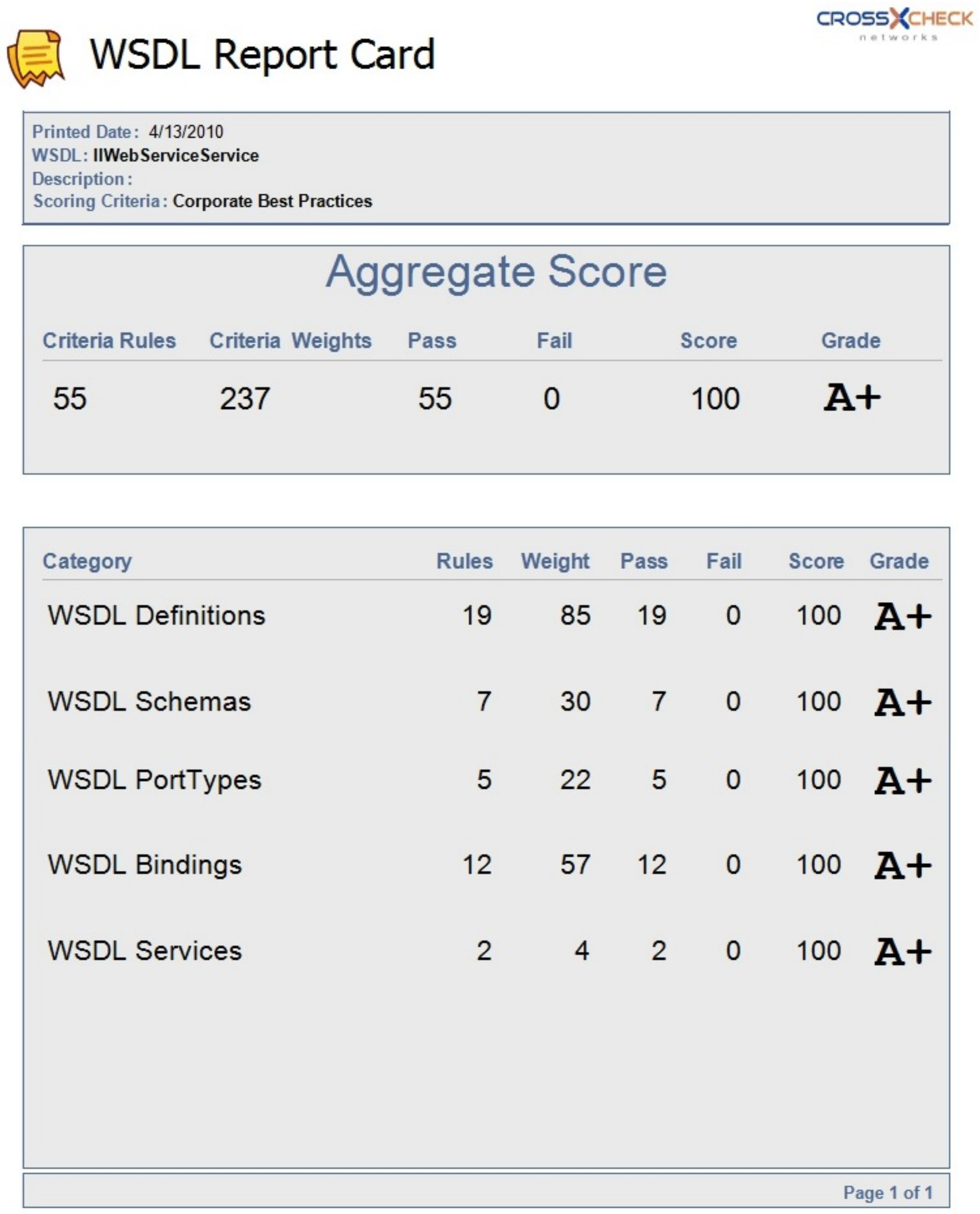
Integral/Separable:
- N/A

Binary Range Translation:
- On-On / Off-Off

Stickiness:
- Compliance

(b) Sample mapping of control point value manipulation using a digital input primitive with high resemblance.

Figure 8.3: Two sample mappings of control point selection and control point manipulation using analog and digital input primitives.



WSDL Report Card

CROSSCHECK networks

Printed Date: 4/13/2010
WSDL: IIWebServiceService
Description:
Scoring Criteria: Corporate Best Practices

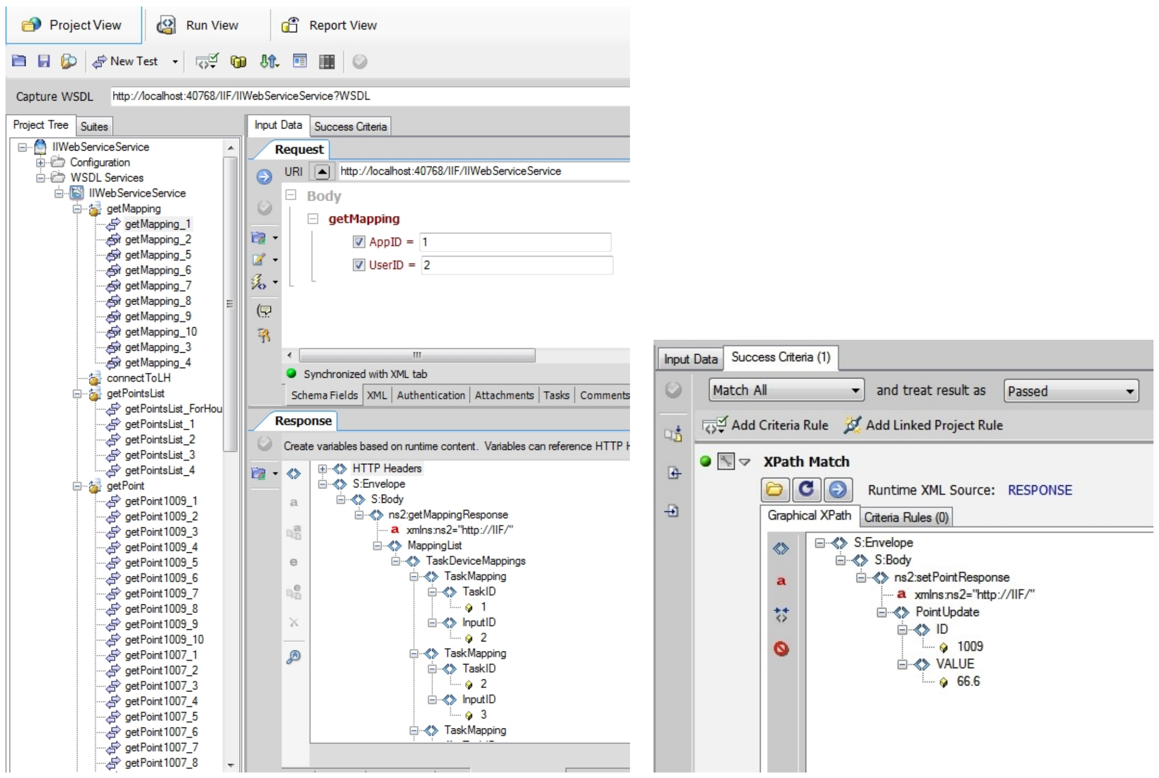
Aggregate Score

Criteria Rules	Criteria Weights	Pass	Fail	Score	Grade
55	237	55	0	100	A+

Category	Rules	Weight	Pass	Fail	Score	Grade
WSDL Definitions	19	85	19	0	100	A+
WSDL Schemas	7	30	7	0	100	A+
WSDL PortTypes	5	22	5	0	100	A+
WSDL Bindings	12	57	12	0	100	A+
WSDL Services	2	4	2	0	100	A+

Page 1 of 1

Figure 8.4: SOAPSonar WSDL Score Card for the Framework reference implementation.



Capture WSDL http://localhost:40768/IIF/IIWebServiceService?WSDL

```
IIWebServiceService
  Configuration
  WSDL Services
    IIWebServiceService
      getMapping
        getMapping_1
        getMapping_2
        getMapping_5
        getMapping_6
        getMapping_7
        getMapping_8
        getMapping_9
        getMapping_10
        getMapping_3
        getMapping_4
      connectToLH
      getPointsList
        getPointsList_ForHou
        getPointsList_1
        getPointsList_2
        getPointsList_3
        getPointsList_4
      getPoint
        getPoint1009_1
        getPoint1009_2
        getPoint1009_3
        getPoint1009_4
        getPoint1009_5
        getPoint1009_6
        getPoint1009_7
        getPoint1009_8
        getPoint1009_9
        getPoint1009_10
        getPoint1007_1
        getPoint1007_2
        getPoint1007_3
        getPoint1007_4
        getPoint1007_5
        getPoint1007_6
        getPoint1007_7
        getPoint1007_8
```

URI http://localhost:40768/IIF/IIWebServiceService

```
Body
  getMapping
    AppID = 1
    UserID = 2
```

Synchronized with XML tab

Create variables based on runtime content. Variables can reference HTTP H

```
HTTP Headers
S:Envelope
  S:Body
    ns2:getMappingResponse
      xmlns:ns2="http://IIF/"
      MappingList
        TaskDeviceMappings
          TaskMapping
            TaskID
              1
            InputID
              2
          TaskMapping
            TaskID
              2
            InputID
              3
          TaskMapping
```

Success Criteria (1)

Match All and treat result as Passed

XPath Match

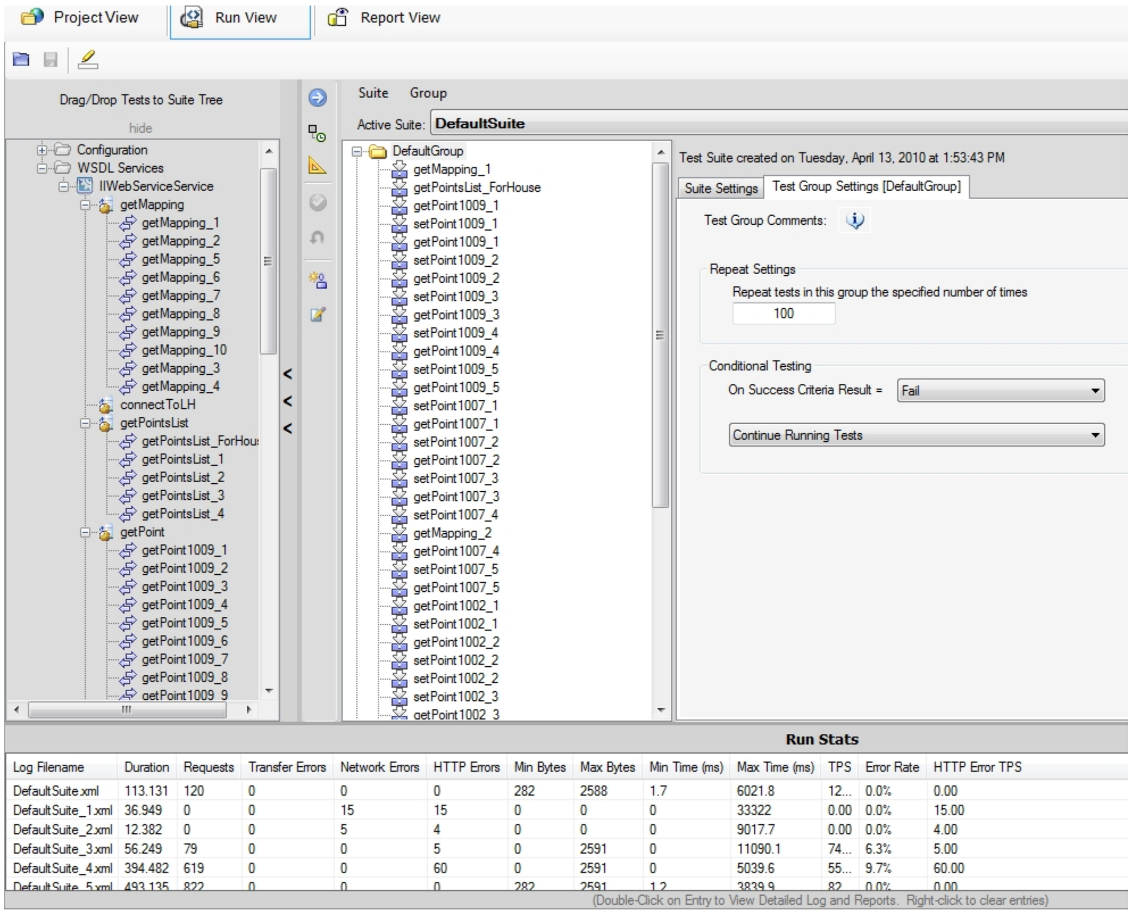
Runtime XML Source: RESPONSE

```
S:Envelope
  S:Body
    ns2:setPointResponse
      xmlns:ns2="http://IIF/"
      PointUpdate
        ID
          1009
        VALUE
          66.6
```

(a) Creating a SOAP test call in SOAPSonar. Shown is the mapping SOAP request.

(b) Test success criteria put as a XML match. Shown is the response for a Update Control Point SOAP request

Figure 8.5: A screen capture of SOAPSonar illustrating a SOAP Test and the Success Criteria to be used [SOA, 2010]. Every reasonable effort has been made to inform the copyright owner of this fair use of their image.



Active Suite: DefaultSuite

Configuration
WSDL Services
IIWebServiceService
getMapping
getMapping_1
getMapping_2
getMapping_5
getMapping_6
getMapping_7
getMapping_8
getMapping_9
getMapping_10
getMapping_3
getMapping_4
connectToLH
getPointsList
getPointsList_ForHou
getPointsList_1
getPointsList_2
getPointsList_3
getPointsList_4
getPoint
getPoint1009_1
getPoint1009_2
getPoint1009_3
getPoint1009_4
getPoint1009_5
getPoint1009_6
getPoint1009_7
getPoint1009_8
getPoint1009_9

DefaultGroup
getMapping_1
getPointsList_ForHouse
getPoint1009_1
setPoint1009_1
getPoint1009_1
setPoint1009_2
getPoint1009_2
setPoint1009_3
getPoint1009_3
setPoint1009_4
getPoint1009_4
setPoint1009_5
getPoint1009_5
setPoint1007_1
getPoint1007_1
setPoint1007_2
getPoint1007_2
setPoint1007_3
getPoint1007_3
setPoint1007_4
getMapping_2
getPoint1007_4
setPoint1007_5
getPoint1007_5
getPoint1002_1
setPoint1002_1
getPoint1002_2
setPoint1002_2
setPoint1002_2
setPoint1002_3
getPoint1002_3

Test Suite created on Tuesday, April 13, 2010 at 1:53:43 PM

Suite Settings | Test Group Settings [DefaultGroup]

Test Group Comments:

Repeat Settings
Repeat tests in this group the specified number of times
100

Conditional Testing
On Success Criteria Result = Fail
Continue Running Tests

Run Stats

Log Filename	Duration	Requests	Transfer Errors	Network Errors	HTTP Errors	Min Bytes	Max Bytes	Min Time (ms)	Max Time (ms)	TPS	Error Rate	HTTP Error TPS
DefaultSuite.xml	113.131	120	0	0	0	282	2588	1.7	6021.8	12...	0.0%	0.00
DefaultSuite_1.xml	36.949	0	0	15	15	0	0	0	33322	0.00	0.0%	15.00
DefaultSuite_2.xml	12.382	0	0	5	4	0	0	0	9017.7	0.00	0.0%	4.00
DefaultSuite_3.xml	56.249	79	0	0	5	0	2591	0	11090.1	74...	6.3%	5.00
DefaultSuite_4.xml	394.482	619	0	0	60	0	2591	0	5039.6	55...	9.7%	60.00
DefaultSuite_5.xml	493.135	822	0	0	0	282	2591	1.2	3839.9	82	0.0%	0.00

(Double-Click on Entry to View Detailed Log and Reports. Right-click to clear entries)

Figure 8.6: Building a SOAPSonar Test Suite for a complete usage scenario [SOA, 2010]. Every reasonable effort has been made to inform the copyright owner of this fair use of their image.

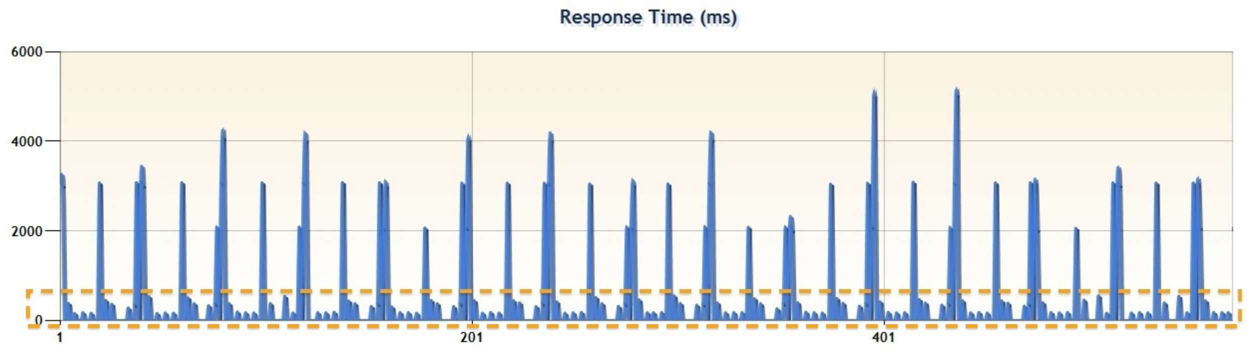
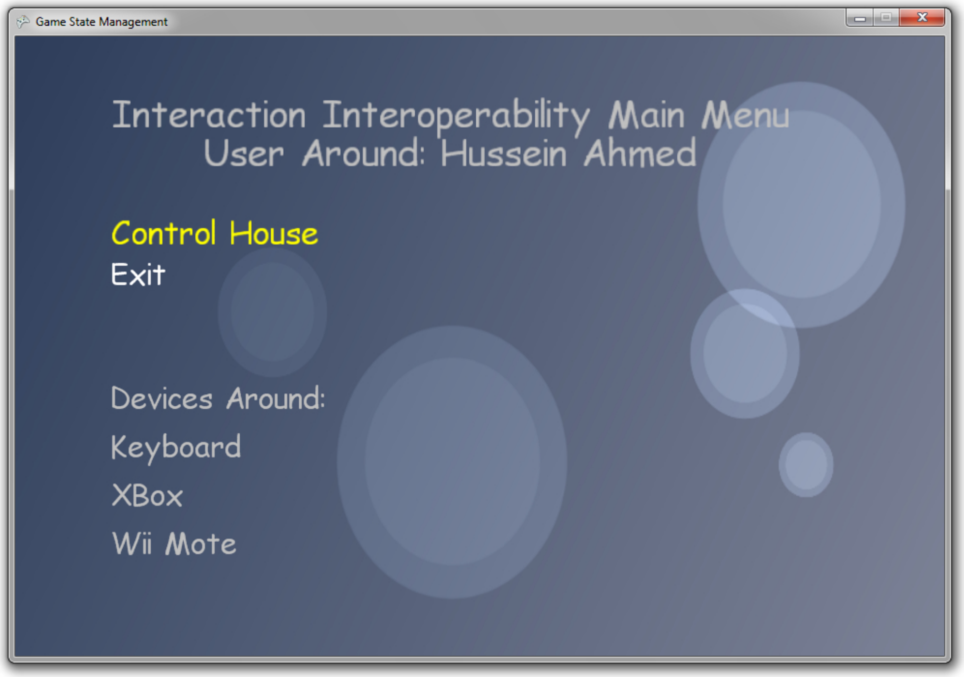
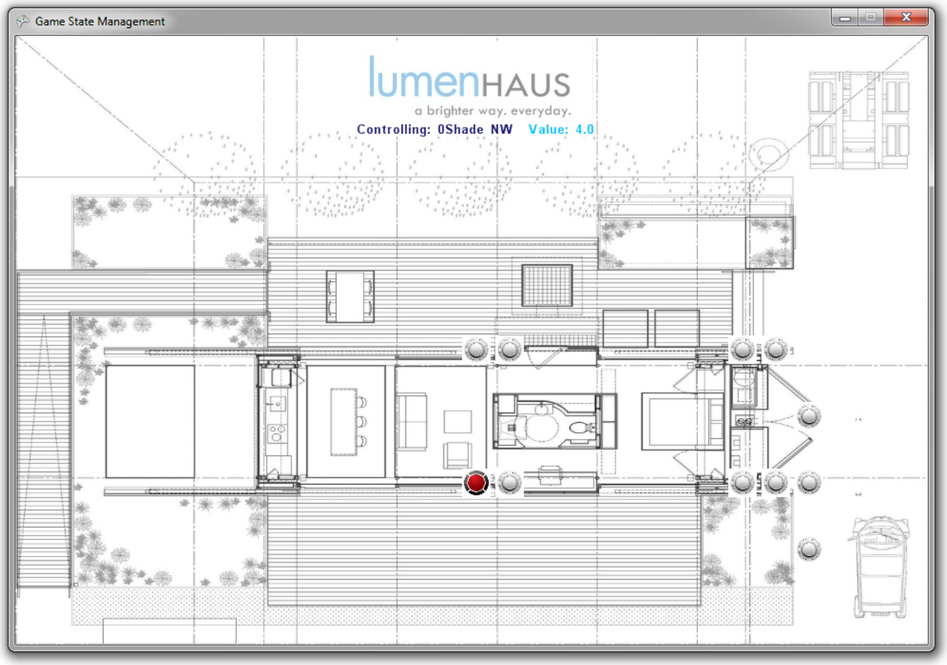


Figure 8.7: Web Service Response Time illustrating the contrast between the framework delays and the actual controller functionality.



(a) The main menu showing context awareness with the user automatically detected as well as the devices around.



(b) The LumenHaus control screen with the point selected in red. The dials shown indicate an analog control point like the sliding doors.

Figure 8.8: A screen shot of the XNA .Net application that consumes the framework web service to support interaction interoperability.

Chapter 9

Conclusion

This dissertation started with the premise stating that *it is feasible to abstract input devices and interaction tasks offering context sensitive mapping between both for large scale DVEs without compromising performance*. Given this main premise behind this dissertation and the work presented so far, this chapter concludes the dissertation and provides a summary, contributions of my work, discussion and possible future work.

In Section §9.1 a summary of the work presented in each of the previous chapters is presented. Section §9.2 lists and discusses the contributions of the work. Section §9.3 discusses possible extensions and expansions of the work.

Publications related to this dissertation, each highlighting certain parts of the work presented are [Ahmed et al., 2008b, Ahmed et al., 2008c, Ahmed et al., 2008a, Ahmed et al., 2010, Ahmed and Gracanin, 2010].

9.1 Summary

The focus of this dissertation is on affording transparent, context-aware interaction interoperability for DVEs supporting a large number of concurrent users. The motivations behind the work, as stated before, are:

- Utilize the user context in both the real world and virtual world.
- Offer a transparent, globally accessible, personalization service to applications thus elevating user satisfaction and performance.
- Offer an evaluation testbed that can be used to test new interactions (including reality-based ones) against real-life legacy applications instead of developing unrealistic mock-ups for testing.
- Make it possible for users to use any input device suitable to control any application.
- Bridge between real world and DVEs replicating interactions both ways.

Most of the existing DVEs are designed and built for a specific way of interaction using a specific input device. DVEs require all the collaborating users, with a variety of backgrounds, platforms and personal preferences, to interact with a virtual world in the same way.

Another downside of the current DVEs from the context awareness stand point is that most DVEs are concerned about the real world context of the user. For example, a user at a home has a different context then when the user is in the office. What is missing is the integration of the user context in a DVE to create the complete context.

In most of DVEs that are built to mimic a real world environment, the real and virtual worlds are not synchronized. If an office building is created in a DVE to mimic a real world building, the consistency of these two worlds should be maintained. That would reflect accuracy, consistency and further elevate the “reality ” of DVEs. As an example, opening a door of a physical office which has a replica in a DVE like Second Life or There.com then the door in the DVE is expected to be open as well and vice versa.

In Chapter 1, the importance of DVEs and CVEs was discussed highlighting how the user base is growing significantly and consistently over time. Issues, problems and research challenges were presented to highlight the research questions, objectives and motivations.

The areas of contribution span input devices classification, application profiling, user modeling, context elicitation in real and virtual environments and finally service oriented architectures for systems interoperability. Various approaches in each of these areas and overlapping work are surveyed in Chapter 2.

Chapter 3 focuses then on the concept of interaction interoperability. The chapter provides motivation to use and research interaction interoperability. Figure 1.2 shows the core concept that is based on abstracting the two sides of the interaction cycle, the input device physically manipulated by the user and at the other end the application task afforded and executed upon a specific event. The chapter also presents the conceptual model and the design requirements to achieve the goals set.

Context-awareness is then presented and discussed through Chapter 4. The chapter discusses the classical definition of context, the classical primary dimensions: identity, time, activity and location. It is highlighted that the work presented is not only concerned with the real world user context like all classical context-aware application. Instead, context is to be elicited from the two planes separately (all four dimensions with identity being common) and then an integral context is made up of all these dimensions across the planes.

Another important point in regard to context sensitivity is highlighted in Section §4.1. In most of the classical context-aware systems context adaptation is based on a *given* set of context rules that fires whenever context conditions are satisfied. It was explained that this approach of having a fixed set of rules is not feasible for an ever changing device set and application profiles. The concept used in previous work was adapted to facilitate generation of context rules on demand upon user intervention. Interventions imply that the user was not satisfied with the mapping decision and have chosen another one instead. The intervention signal received back from the client signals updating/deleting the old rules or/and creation of new rules.

Chapter 5 starts off by describing the knowledge-base backing the framework. The input devices ontology was highlighted against all significant contributions in the field of classifying and describing devices. All major attributes and classifiers have been included to ensure a fully descriptive profile that takes into account all the upsides and downsides of previous work. The sample devices were used to illustrate how can they be described using the ontology designed providing all the need attributes to facilitate mapping devices to application tasks.

In addition to the device profiles, the profiles of application tasks with detailed descriptions of expected dimensions, ranges and nature of task are presented focusing on how the breakdown of devices and application tasks would ease mapping inference to be done trying to match tasks to input devices.

Chapter 6 describes the design put for the interaction interoperability framework. The chapter presents the different options available for the architectural design pattern and discusses the decision of going with the hybrid architectural pattern given it supports the design requirements, eliminates shortcomings of others and support achieving a decoupled, distributed framework.

Figure 6.3 in Chapter 6 shows the five main building blocks of the interaction interoperability framework: input device profiling, application profiling, user modeling, context management and finally, the decision making unit. Each of these five units is described in details in separate sections highlighting the design requirements, challenges, trade offs and the optimal design approach given the module functionality, the distributed, decoupled nature of the framework as well as the whole integration of the five units together to perform as one with a set of advertised services offered for applications to consume.

Given the framework design laid out in Chapter 6, the framework can be developed in various ways. Chapter 7 describes a number of different reference implementations. Chapter 7 goes through a reference implementation of each of the five modules that makes up the interaction interoperability framework.

Through Chapter 7 it is explained how that SOA is the most appropriate architecture to impelment the framework. Other development choices were in building a suitable way to create and manipulate devices and application profiles using a wizard like interface made up of a set of questions that would eliminate conflicts and elevate ease of use. A detailed explanation of the development decisions taken for input device profiling, application profiling, user modeling, context management and the decision making unit is presented.

Chapter 8 provides evaluation results. The first two sections are devoted to evaluating and analyzing the work presented in this dissertation. Section §8.1 focuses on evaluating the extensibility and usefulness of the framework. Extensibility reflects how easily and accurately the framework can accommodate new input devices and applications while still maintaining consistency and effectiveness of mappings generation. A number of complicated input devices were accurately incorporated in the framework with precisely defined attributes. The next section evaluates the framework performance. It was stressed earlier in the dissertation that the framework motivation is to cope with large scale DVEs like Second Life which have a large number of concurrent users. Given these working load conditions, the framework was evaluated with SOAP web services testing tools using various test case scenarios and stress loading. The results from the evaluation proved that the framework should perform well despite the inherent delays. The main reason as the analysis described is that the major work done by the framework is at start up and given the delays experienced by typical DVEs (content transfer, 3D rendering, users collaborative tasks and physical control), the delays being added for using the framework are considered negligible especially in the case of real world control (like LumenHaus) which posses significantly large delay times.

Elaborating more on the LumenHaus case study, Section §8.3 highlights the case study development. The case study had all the functionalities offered by the framework in implementation and showed how the framework can be used to bridge between the real world and virtual world taking into consideration the user context in both worlds and even also replicating actions between both planes maintaining consistency between the two worlds of the user.

9.2 Main Contributions

The main key contributions of my work are as follows:

- **Abstracting Input Devices and Application Tasks.**
  Input devices, as reviewed in Chapter 2 had been classified and sorted according to certain criteria and factors. In this dissertation, input devices are rather described explicitly by

utilizing an ontology that is made up of general classes and input devices are described as instances and aggregations of these classes with individual values of attributes .

With the hierarchical position of an input device, in addition to all attributes defined and assigned values, it is possible to define any sort of simple and complicated multi-dimensional input devices and yet still have practical inference possible to take place against these definitions and the application in hand.

Application tasks have been classically classified through literature based on the major classes navigation, selection and manipulation, control and symbolic input. The contribution here is an ontology of application tasks, again with the classical classes hierarchy but still each of the classes has a set of attributes that can accurately define a task in order to facilitate matching it to a suitable device.

- **Bridging Real and Virtual Worlds in Context and Actions**

  Another contribution of this work is in not just basing context sensitive actions on the real world's user contextual dimensions but also taking into consideration the contextual dimensions in the virtual world.

  With the wide spread of *mirrored* VEs which are basically VEs built to mimic a real one, there should be consistency between the virtual and real worlds. As presented in the LumenHaus case study, opening a door in one world (real or virtual) would causes the other world to have that corresponding door opened as well.

- **A Service Oriented based Framework and an Evaluation Testbed .**

  The Interaction Interoperability framework offers all possible services through a published web service that has a WSDL that any application or requestor can use to invoke services needed. Aside also from the external interface, the framework modules are designed as well based on the service centric architecture. The front end, complex web service, should consume other services giving more flexibility, scalability and loose coupling with all what is exhibits of advantages.

  The concept of interaction interoperability and the presented framework can function as an evaluation testbed. The contribution is in offering interactions developers and researchers the ability to test any sort of input device/technique against fully fledged applications with full functionality that users do have experience with, in contrast to mock ups and prototypes that will not reflect the real effectiveness of these novel interactions when used with real life applications.

- **Transparent Context Sensitive Interaction Interoperability for Legacy Systems.** It would not have been practical or feasible to mandate developing interaction interoperability aware application in order to take advantage of the framework. The design laid out utilizing a client wrapper makes it possible to still make use of interaction interoperability even with legacy application unaware of the framework.

Given these contributions, the work presented has added to the body of knowledge in defining input devices based on a set of classes and attributes and that applies as well to application tasks. Previous work as stated did not take the approach of defining, but rather classifying interactions.

With the framework contribution of context-sensitive interaction interoperability which is offered as a set of discoverable services, this would enable any application written in any language and running on any platform to connect to and take advantage of all offered services transparently. Even legacy applications unaware of the framework, given that they run on top of the platform-specific client wrappers they can still utilize all the functionalities offered.

Basing context-awareness rules on a working set derived from database records that could be used directly or for large scale to be used to generate on-the-fly XML rules for a rule engine to work with, makes it possible to offer context-sensitivity to such large scale DVEs without compromising performance or going for expensive, cost ineffective deployments.

9.3 Future Work

The presented results can be extended in various ways. This section describes some areas of possible future work to be done and what could be done in each of these areas.

9.3.1 Knowledge Base

One direction is to start with exhaustively defining all input devices, interaction techniques and application tasks available and by that there will be comprehensive ontologies available for use by any other application or research. The presented framework would also make use of such a comprehensive knowledge base.

9.3.2 Distributed Systems

Given the framework is built based on a service-centric design, a research question would be in how to synchronize different replicas of the framework across the Internet. What is meant by synchronization in this case is basically how to take advantage of previous knowledge built through other sites and build on that instead of going from scratch at every site. This synchronization of knowledge should also happen over time, with the possibility of different sites or service providers hosting their own copy of the framework.

9.3.3 Input Devices Interfacing

The framework can through the context-management server asses what devices are around a computer and pick the appropriate one again based on *who* is on the computer and the current context. The problem was then in the operating system.

There is a possibility that there could be a device around and its powered and turned on but it is still not *recognized* by the operating system. This means that for a device to work, its driver should be installed on the machine, be properly plugged in and clearly recognized by the operating system. Future work to be done should find a way to *unify* input devices interfacing. There are standards like the USB-HID or Bluetooth HID but still these standards do not support immediate plug-and-play functionality and simply no one can walk in with a new device and start using it right away. Possible venues would be to standardize the interface between devices and the operating system. To start in this direction there should be a way to unify the description of devices and that is already doable with the framework presented. The OS can go through the classes, find the closest fit and then go through the attributes setting one by one and thus it can now make use of every primitive and every controllable dimension the device can offer.

Another direction of future work in the area of Input devices abstraction and interfacing would be creating a set of adapters for a variety of development platforms like Java, .NET and others. The idea of having such a development platform adapter would be in order to eliminate the coding needed by a developer who wants to integrate the framework into an application. The adapter will be basically shipped with a set of supported devices and the API of the adapter would support a higher level of abstraction for the application. Device events will be caught and interpreted by the adapter which in its turn consumes the framework backend accordingly.

9.3.4 Evaluation

Future work in testing and evaluation would be in building a simulator that can simulate real and virtual worlds, record and play different scenarios and then run the scenarios against a set of user profiles. That would allow to evaluate the response times, mapping delays and overall performance with simulated inherent delays to occur with physical controllers and communication to on-line DVEs like Second Life. This can be elaborated more if we can think of a browser testing recorder. Such a suite recorder is aware of the action and responses happening and thus powerful and much more meaningful and real test cases could be created.

9.4 Discussion and Reflections

Researching previous work done in input devices, interface abstraction, building context-sensitive applications, interface personalization and service centric architectures, I have realized that there is a gap in integrating all these approaches together. Going through the analysis and design, I was trying to build upon previous work instead of starting from scratch. I did not propose new interactions, I did not propose even new attributes for abstracting devices, application tasks and the service centric architectures. I am not proposing a new approach or an architectural design through my dissertation. Instead, my work was in trying to get the optimal approach in each of these domains in relation to the others, building a modular, loosely coupled distributed system that would offer what I am calling as Interaction Interoperability.

These reasons motivated to go for a full descriptive repositories of input devices and application tasks and to have these repositories distributed, accessible and consistently map to each other. Context sensitivity which is based on hard coded rules or a centralized rule engine would not be grid ready to handle large scale deployments without compromising performance. For the framework, going for a desktop application or a platform specific solution would not have been a practical way to integrate different components each working independently and at the end offer a unified interface that is discoverable, interoperable and again grid ready with stateless services.

Regarding interfaces and mainly input devices not applications, developing the reference implementation I have noticed that still there is not a standard way or protocol to communicate with input devices. Standards like the Universal Serial Bus-Human Interface Device (USB-HID) standardize the networking and connection only. There should be standards to read and communicate with devices regardless of the manufacturer, model or nature. Still though, to achieve such standardization would require having a standard to define devices and a straight forward solution for this would be my approach in abstracting devices. If a device can send in its profile as a response for a standard call, the operating system or applications can then communicate with the device based on this profile that would hold all the attributes I have presented to *describe* itself Doing so there should be no need to have platform/language specific libraries in order to interface with a non-traditional input device.

Through the user study I have done on the preliminary implementations of the framework, the users greatly appreciated how they were able to control an application using any available device. Despite the importance of personalization and a positive impact on satisfaction and performance, we do rarely find applications or websites that take advantage of that and offer a personalized experience that would even persist in accessing other applications and websites. It is shown in literature that performance and satisfaction significantly elevates with personalized interfaces and yet there is no significant presence of context sensitive applications in spite of a large number of domains that can benefit from more engaging users experience and higher group performance.

Bibliography

[Flu, 2008] (2008). Flux X3D Player. http://www.mediamachines.com/downloadplayerty.php.

[OWL, 2008] (2008). OWL Web Ontology Language. http://www.w3.org/TR/owl-guide/.

[Wii, 2008] (2008). Project WiiController4SecondLife. http://sourceforge.net/projects/wii4sl/.

[Won, 2008] (2008). Project Wonderland: Toolkit for Building 3D Virtual Worlds. https://lg3d-wonderland.dev.java.net/.

[SAI, 2008] (2008). SAI (Scene Authoring Interface / Scene Access Interface). http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/.

[SPA, 2008] (2008). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[Xj3, 2008] (2008). The Open Source Java Xj3D browser. http://www.xj3d.org/.

[Usi, 2008] (2008). UsiXML Group. http://www.usixml.org/.

[OnA, 2009] (2009). On-A-Slant Virtual Village. http://onaslant.ndsu.edu/x3d.html.

[Pro, 2009] (2009). Protege-OWL. http://protege.stanford.edu/.

[Sec, 2009] (2009). Second Life Users. https://blogs.secondlife.com/community/features/blog/2008/04/15/.

[Jav, 2009] (2009). Sun JavaSpaces. http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/.

[The, 2009] (2009). There.com Collaborative Virtual Environment. http://www.there.com.

[XMP, 2009] (2009). XMPP Standards Foundation. http://xmpp.org/xsf/.

[Div, 2010] (2010). Device independent virtual environment - an open source virtual reality toolkit. http://diverse.sourceforge.net/diverse/.

[Eka, 2010] (2010). Ekahau Wi-Fi based location tracking solution. http://www.ekahau.com/.

[eNo, 2010] (2010). eNode Project. http://www.enode.com/.

[Fre, 2010] (2010). FreeWRL, Communications Research Centre. http://www.crc.ca/FreeWRL.

[USB, 2010] (2010). Human Interface Device Usage for USB. http://www.usb.org.

[jes, 2010] (2010). Jess Rule Engine. http://herzberg.ca.sandia.gov/jess/.

[Lin, 2010] (2010). LSL Linden Scripting Language. http://wiki.secondlife.com.

[xna, 2010] (2010). Microsoft's XNA Game Studio. http://creators.xna.com/.

[MPK, 2010] (2010). MPK20: Sun's Virtual Workplace . http://labs.oracle.com/projects/mc/mpk20.html.

[dir, 2010] (2010). Project DirectFB. http://www.directfb.org/.

[Pus, 2010] (2010). PushToTest; Test, Monitor, and Automate Web Services for Reliability, Performance, Functionality, and Scalability. http://www.pushtotest.com.

[RDF, 2010] (2010). Resource Description Framework (RDF). http://www.w3.org/RDF/.

[Sec, 2010] (2010). Second Life Collaborative Virtual Environment. http://www.secondlife.com.

[Ses, 2010] (2010). Sesame the open source RDF framework. http://www.openrdf.org/.

[SOA, 2010] (2010). SOAPSonar: Service and ESB Testing available in Software, VMWare, and Cloud Image. http://www.crosschecknet.com/products/soapsonar.php.

[Jen, 2010] (2010). The Jena Semantic Web Framework. http://jena.sourceforge.net/.

[vrm, 2010] (2010). VRML97 Specification. http://www.web3d.org/x3d/specifications/vrml/.

[Web, 2010] (2010). Web3D Consortium. http://www.web3d.org. Last accessed on January 12, 2010.

[xVR, 2010] (2010). xVRML Project.

[Ahmed and Gracanin, 2010] Ahmed, H. and Gracanin, D. (2010). Context sensitive interaction interoperability for serious virtual worlds. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*, pages 159 –166.

[Ahmed et al., 2008a] Ahmed, H., Gracanin, D., and Hamid, A. A. (2008a). Poster: A framework for interaction interoperability in virtual environments. In *Proceedings of 3DUI 2008 Symposium*.

[Ahmed et al., 2010] Ahmed, H., Gracanin, D., and Radics, P. (2010). Context sensitive interaction interoperability for distributed virtual environments. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 251 –252.

[Ahmed et al., 2008b] Ahmed, H. M., Gracanin, D., Abdel-Hamid, A., and Matkovic, K. (2008b). An approach to interaction interoperability for distributed virtual environments. In *Short papers and posters proceedings of the EGVE 2008*, pages 35–38.

[Ahmed et al., 2008c] Ahmed, H. M., Gračanin, D., and Abdel-Hamid, A. (2008c). Interaction interoperability in x3d mobile collaborative virtual environments. In *Proceedings of Sixth Annual Conference INFOS 2008, Cairo University*.

[Baudisch et al., 2007] Baudisch, P., Sinclair, M., and Wilson, A. (2007). Soap: How to make a mouse work in mid-air. *CHI 2007*.

[Behr et al., 2004] Behr, J., Dähne, P., and Roth, M. (2004). Utilizing x3d for immersive environments. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, pages 71–78, New York, NY, USA. ACM Press.

[Bellur and Bondre, 2008] Bellur, U. and Bondre, S. (2008). Towards seamless user mobility in service oriented environments via context awareness. In *ICPS '08: Proceedings of the 5th international conference on Pervasive services*, pages 185–188, New York, NY, USA. ACM.

[Benford et al., 1997] Benford, S., Snowdon, D., Colebourne, A., O'Brien, J., and Rodden, T. (1997). Informing the design of collaborative virtual environments. In *GROUP '97: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 71–80, New York, NY, USA. ACM Press.

[Bierig and Göker, 2006] Bierig, R. and Göker, A. (2006). Time, location and interest: an empirical and user-centred study. In *IIiX: Proceedings of the 1st international conference on Information interaction in context*, pages 79–87, New York, NY, USA. ACM.

[Bjork et al., 2002] Bjork, S., Holopainen, J., Ljungstrand, P., and Akesson, K.-P. (2002). Designing ubiquitous computing games - a report from a workshop exploring ubiquitous computing entertainment. *Personal Ubiquitous Comput.*, 6(5-6):443–458.

[Bleser and Sibert, 1990] Bleser, T. W. and Sibert, J. (1990). Toto: a tool for selecting interaction techniques. In *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 135–142, New York, NY, USA. ACM.

[Bonino da Silva Santos et al., 2007] Bonino da Silva Santos, L. O., Vink, P., and van Wijnen, R. P. (2007). A service-oriented middleware for providing context awareness and notification. In *MC '07: Proceedings of the 2007 ACM/IFIP/USENIX international conference on Middleware companion*, pages 1–2, New York, NY, USA. ACM.

[Bosca et al., 2007] Bosca, A., Bonino, D., Comerio, M., Grega, S., and Corno, F. (2007). A Reusable 3D Visualization Component for the Semantic Web. In *Web3D '07: Proceedings of the Twelfh International Conference on 3D Web Technology*, pages 89–96, New York, NY, USA. ACM Press.

[Bowman et al., 2005] Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. (2005). *3D User Interfaces Theory and Practice*. Addison Wesley.

[Broll et al., 2006] Broll, W., Lindt, I., and Wittkämper, M. (2006). Mixed reality user interface description language. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters*, page 156, New York, NY, USA. ACM.

[Brown and Bell, 2004] Brown, B. and Bell, M. (2004). CSCW at play: 'there' as a Collaborative Virtual Environment. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 350–359, New York, NY, USA. ACM Press.

[Bullock and Fahlén, 2000] Bullock, A. and Fahlén, L. (2000). Designing for and interacting with cves. *SIGGROUP Bull.*, 21(1):26–27.

[Buxton, 1983] Buxton, W. (1983). Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph.*, 17(1):31–37.

[Cabrera et al., 2001] Cabrera, L. F., Jones, M. B., and Theimer, M. (2001). Herald: Achieving a global event notification service. In *In HotOS VIII*. IEEE Computer Society.

[Card et al., 1990] Card, S. K., Mackinlay, J. D., and Robertson, G. G. (1990). The design space of input devices. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 117–124, New York, NY, USA. ACM.

[Carriero et al., 1989] Carriero, Nicholas, Gelernter, and David (1989). Linda in context. *Commun. ACM*, 32(4):444–458.

[Carter et al., 2006] Carter, S., Hurst, A., Mankoff, J., and Li, J. (2006). Dynamically adapting GUIs to diverse input devices. In *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 63–70, New York, NY, USA. ACM.

[Carzaniga et al., 2000] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. (2000). Achieving scalability and expressiveness in an internet-scale event notification service. In *PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 219–227, New York, NY, USA. ACM.

[Carzaniga et al., 1998] Carzaniga, A., Wolf, A. L., Carzaniga, C. A., Rosenblum, D. S., Rosenblum, D. S., and Wolf, E. L. (1998). Design of a Scalable Event Notification Service: Interface and Architecture. Technical report.

[Celentano et al., 2004] Celentano, A., Nodari, M., and Pittarello, F. (2004). Adaptive interaction in web3d virtual worlds. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, pages 41–50, New York, NY, USA. ACM Press.

[Chen et al., 1997] Chen, S., Myers, R., and Pasetto, R. (1997). The out of box experience: lessons learned creating compelling VRML 2.0 content. In *VRML '97: Proceedings of the second symposium on Virtual reality modeling language*, pages 83–ff., New York, NY, USA. ACM.

[Chow, 2006] Chow, R. (2006). Modeling and simulation of context-awareness. In *ANSS '06: Proceedings of the 39th annual Symposium on Simulation*, page xi, Washington, DC, USA. IEEE Computer Society.

[Christopoulou et al., 2005] Christopoulou, E., Goumopoulos, C., and Kameas, A. (2005). An ontology-based context management and reasoning process for ubicomp applications. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 265–270, New York, NY, USA. ACM.

[Claypool, 2005] Claypool, M. (2005). On the 802.11 Turbulence of Nintendo DS and Sony PSP hand-held network games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9, New York, NY, USA. ACM Press.

[Costa et al., 2005] Costa, P. D., Pires, L. F., and van Sinderen, M. (May 2005). Architectural patterns for context-aware services architectural patterns for context-aware services platforms. In *Proceedings of the Second International Workshop on Ubiquitous Computing (IWUC 2005), Miami*, pages pp 3–19.

[da Rocha et al., 2008] da Rocha, R. C. A., Endler, M., and de Siqueira, T. S. (2008). Middleware for ubiquitous context-awareness. In *MPAC '08: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*, pages 43–48, New York, NY, USA. ACM.

[Dachselt et al., 2006] Dachselt, R., Hinz, M., and Pietschmann, S. (2006). Using the amacont architecture for flexible adaptation of 3d web applications. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, pages 75–84, New York, NY, USA. ACM Press.

[Dachselt and Rukzio, 2003] Dachselt, R. and Rukzio, E. (2003). Behavior3D: an XML-based framework for 3D graphics behavior. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology*, pages 101–ff, New York, NY, USA. ACM.

[Dargie, 2006] Dargie, W. (2006). Dynamic generation of context rules. *Self-Managed Networks, Systems, and Services*, pages 102–115.

[Dargie and Schill, 2009] Dargie, W. and Schill, A. (2009). Enabling group-awareness through context-based service provisioning. In *Casemans '09: Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, pages 25–30, New York, NY, USA. ACM.

[Davies and Peel, 2006] Davies, M. and Peel, R. (2006). Designing user interfaces for future mobile applications and devices: a user-centred interaction and visual design process walkthrough. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 287–288, New York, NY, USA. ACM Press.

[de Freitas, 2008] de Freitas, S. (2008). Serious Virtual Worlds: A scoping study. http://www.jisc.ac.uk/publications/documents/seriousvirtualworldsreport.aspx.

[de Freitas, 2009] de Freitas, S. (2009). Learning in Immersive worlds Learning in Immersive worlds, A review of game-based learning. http://www.jisc.ac.uk/eli_outcomes.html.

[de Ipia, 2001] de Ipia, D. L. (2001). An ECA Rule-Matching Service for Simpler Development of Reactive Applications. IEEE Distributed Systems Online, Vol. 2, No. 7.

[de Paula et al., 2005] de Paula, M. G., Barbosa, S. D. J., and de Lucena, C. J. P. (2005). Conveying human-computer interaction concerns to software engineers through an interaction model. In

CLIHC '05: Proceedings of the 2005 Latin American conference on Human-computer interaction, pages 109–119, New York, NY, USA. ACM Press.

[Degler et al., 2007] Degler, D., Henninger, S., and Battle, L. (2007). Semantic web HCI: discussing research implications. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 1909–1912, New York, NY, USA. ACM Press.

[Deshpande and Schultz, 1992] Deshpande, A. and Schultz, M. (1992). Efficient parallel programming with Linda. In *Supercomputing '92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pages 238–244, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Devaraju et al., 2007] Devaraju, A., Hoh, S., and Hartley, M. (2007). A context gathering framework for context-aware mobile solutions. In *Mobility '07: Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 39–46, New York, NY, USA. ACM.

[Dey, 2000] Dey, A. K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology.

[Duh et al., 2006] Duh, H. B.-L., Tan, G. C. B., and Chen, V. H.-H. (2006). Usability evaluation for mobile device: a comparison of laboratory and field tests. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 181–186, New York, NY, USA. ACM Press.

[Fernandes et al., 2007] Fernandes, V., Guerreiro, T., Araujo, B., Jorge, J., and Pereira, J. (2007). Extensible middleware framework for multimodal interfaces in distributed environments. In *ICMI '07: Proceedings of the 9th international conference on Multimodal interfaces*, pages 216–219, New York, NY, USA. ACM.

[Ferscha, 2003] Ferscha, A. (2003). Coordination in pervasive computing environments. *Enabling Technologies, IEEE International Workshops on*, 0:3.

[Ferscha et al., 2004] Ferscha, A., Holzmann, C., and Oppl, S. (2004). Context awareness for group interaction support. In *MobiWac '04: Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 88–97, New York, NY, USA. ACM.

[Figueroa et al., 2002] Figueroa, P., Green, M., and Hoover, H. J. (2002). InTml: a description language for VR applications. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology*, pages 53–58, New York, NY, USA. ACM.

[Figueroa et al., 2005] Figueroa, P., Medina, O., Jiménez, R., Martínez, J., and Albarracín, C. (2005). Extensions for interactivity and retargeting in X3D. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 103–110, New York, NY, USA. ACM Press.

[Foley et al., 1984] Foley, J. D., Wallace, V. L., and Chan, P. (1984). The human factors of computer graphics interaction techniques. *IEEE Comput. Graph. Appl.*, 4(11):13–48.

[Fritsch et al., 2006] Fritsch, T., Ritter, H., and Schiller, J. (2006). Can mobile gaming be improved? In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 44, New York, NY, USA. ACM Press.

[Furtado et al., 2004] Furtado, E., Furtado, V., Sousa, K. S., Vanderdonckt, J., and Limbourg, Q. (2004). KnowiXML: a knowledge-based system generating multiple abstract user interfaces in USIXML. In *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*, pages 121–128, New York, NY, USA. ACM Press.

[Gabbard et al., 1999] Gabbard, J. L., Hix, D., and Swan, J. E., I. (1999). User-centered design and evaluation of virtual environments. *Computer Graphics and Applications, IEEE*, 19(6):51–59.

[Gajos et al., 2006] Gajos, K. Z., Czerwinski, M., Tan, D. S., and Weld, D. S. (2006). Exploring the design space for adaptive graphical user interfaces. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 201–208, New York, NY, USA. ACM Press.

[Gelernter and Zuck, 1997] Gelernter, D. and Zuck, L. (1997). On what linda is: Formal description of linda as a reactive system. *Coordination Languages and Models*, pages 187–204.

[Goebbels et al., 2003] Goebbels, G., Lalioti, V., and Göbel, M. (2003). Design and evaluation of team work in distributed collaborative virtual environments. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 231–238, New York, NY, USA. ACM Press.

[Greenhalgh, 1999] Greenhalgh, C. (1999). *Large Scale Collaborative Virtual Environments*. Springer-Verlag, London, UK.

[Greenhalgh, 2000] Greenhalgh, C. (2000). Implementing multi-user virtual worlds: Ideologies and issues.

[Hachet et al., 2005] Hachet, M., Pouderoux, J., and Guitton, P. (2005). A camera-based interface for interaction with mobile handheld computers. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 65–72, New York, NY, USA. ACM Press.

[Haiyan Zhang, 2007] Haiyan Zhang, B. H. (2007). Building upon everyday play. *CHI 2007*.

[Heckmann et al., 2005] Heckmann, D., Schwartz, T., Brandherm, B., and Kröner, A. (2005). Decentralized user modeling with userml and gumo. In *Proceedings of DASUM*, pages 61–65.

[Henricksen et al., 2002] Henricksen, K., Indulska, J., and Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. *Pervasive Computing*, pages 79–117.

[Herrmann et al., 2005] Herrmann, C., Kawalek, J., and Stark, A. (2005). User-centred system design: addressing user needs with a mobile information system. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 335–336, New York, NY, USA. ACM Press.

[Hesselman et al., 2007] Hesselman, C., Benz, H., Pawar, P., Liu, F., Wegdam, M., Wibbels, M., Broens, T., and Brok, J. (2007). Bridging context management systems for different types of pervasive computing environments. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–8, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Hilliges et al., 2006] Hilliges, O., Sandor, C., and Klinker, G. (2006). Interactive prototyping for ubiquitous augmented reality user interfaces. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 285–287, New York, NY, USA. ACM Press.

[Holte, 1993] Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11(1):63–90.

[Hong et al., 2005] Hong, D., Chiu, D. K. W., and Shen, V. Y. (2005). Requirements elicitation for the design of context-aware applications in a ubiquitous environment. In *ICEC '05: Proceedings of the 7th international conference on Electronic commerce*, pages 590–596, New York, NY, USA. ACM.

[Hwang et al., 2006] Hwang, J., Jung, J., and Kim, G. J. (2006). Hand-held virtual reality: a feasibility study. In *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 356–363, New York, NY, USA. ACM Press.

[Irawati et al., 2005] Irawati, S., Calderón, D., and Ko, H. (2005). Semantic 3d object manipulation using object ontology in multimodal interaction framework. In *ICAT '05: Proceedings of the 2005 international conference on Augmented tele-existence*, pages 35–39, New York, NY, USA. ACM Press.

[Irawati et al., 2006] Irawati, S., Calderón, D., and Ko, H. (2006). Spatial ontology for semantic integration in 3D multimodal interaction framework. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 129–135, New York, NY, USA. ACM Press.

[Isokoski and Raisamo, 2000] Isokoski, P. and Raisamo, R. (2000). Device independent text input: a rationale and an example. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 76–83, New York, NY, USA. ACM Press.

[Jacob et al., 2008] Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., and Zigelbaum, J. (2008). Reality-based interaction: a framework for post-wimp interfaces. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 201–210, New York, NY, USA. ACM.

[Jacob, 1997] Jacob, R. J. K. (1997). Input devices and techniques. In *The Computer Science and Engineering Handbook*, pages 1494–1511.

[Jacob et al., 1999] Jacob, R. J. K., Deligiannidis, L., and Morrison, S. (1999). A software model and specification language for non-wimp user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 6(1):1–46.

[Jacob and Sibert, 1992] Jacob, R. J. K. and Sibert, L. E. (1992). The perceptual structure of multidimensional input device selection. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 211–218, New York, NY, USA. ACM.

[Jacob et al., 1994] Jacob, R. J. K., Sibert, L. E., McFarlane, D. C., and M. Preston Mullen, J. (1994). Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.*, 1(1):3–26.

[Jellinghaus and Robert, 1990] Jellinghaus and Robert (1990). Eiffel linda: an object-oriented linda dialect. *SIGPLAN Not.*, 25(12):70–84.

[Jones et al., 2004a] Jones, Q., Grandhi, S. A., Terveen, L., and Whittaker, S. (2004a). People-to-people-to-geographical-places: The p3 framework for location-based community systems. *Comput. Supported Coop. Work*, 13(3-4):249–282.

[Jones et al., 2004b] Jones, Q., Grandhi, S. A., Whittaker, S., Chivakula, K., and Terveen, L. (2004b). Putting systems into place: a qualitative study of design requirements for location-aware community systems. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 202–211, New York, NY, USA. ACM.

[Kelso et al., 2006] Kelso, J., Satterfield, S. G., Arsenault, L. E., Ketchan, P. M., and Kriz, R. D. (2006). Diverse: A framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations. *Presence: Teleoperators and Virtual Environments*, 12(1):19–36.

[Klochek and MacKenzie, 2006] Klochek, C. and MacKenzie, I. S. (2006). Performance measures of game controllers in a three-dimensional performance measures of game controllers in a three-dimensional environment.

[Koile et al., 2004] Koile, K., Tollmar, K., Demirdjian, D., Shrobe, H., and Darrell, T. (2004). Activity zones for context-aware computing.

[Koivisto et al., 2006] Koivisto, E. M. I., Suomela, R., and Koivisto, A. (2006). Ancient runes: using text input for interaction in mobile games. In *sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 159–167, New York, NY, USA. ACM Press.

[Kühme, 1993] Kühme, T. (1993). A user-centered approach to adaptive interfaces. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pages 243–245, New York, NY, USA. ACM Press.

[Kumar et al., 2006] Kumar, M., Gupta, A., and Saha, S. (2006). An approach to adaptive user interfaces using interactive media systems. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 312–314, New York, NY, USA. ACM Press.

[Leigh et al., 1997] Leigh, J., Johnson, A. E., and DeFanti, T. A. (1997). Issues in the design of a flexible distributed architecture for supporting persistence and interoperability in collaborative virtual environments. In *Supercomputing '97: Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–14, New York, NY, USA. ACM Press.

[Lieberman and Espinosa, 2006] Lieberman, H. and Espinosa, J. (2006). A goal-oriented interface to consumer electronics using planning and commonsense reasoning. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 226–233, New York, NY, USA. ACM Press.

[Limbourg and Vanderdonckt, 2004] Limbourg, Q. and Vanderdonckt, J. (2004). Addressing the mapping problem in user interface design with UsiXML. In *TAMODIA '04: Proceedings of the 3rd Annual Conference on Task Models and Diagrams*, pages 155–163, New York, NY, USA. ACM Press.

[Lin et al., 2007] Lin, Q., Neo, H. K., Zhang, L., Huang, G., and Gay, R. (2007). Grid-based large-scale Web3D collaborative virtual environment. In *Web3D '07: Proceedings of the Twelfth International Conference on 3D Web Technology*, pages 123–132, New York, NY, USA. ACM.

[Lipscomb and Pique, 1993] Lipscomb, J. S. and Pique, M. E. (1993). Analog input device physical characteristics. *SIGCHI Bull.*, 25(3):40–45.

[Locatelli and Vizzari, 2007] Locatelli, M. P. and Vizzari, G. (2007). Awareness in collaborative ubiquitous environments: The multilayered multi-agent situated system approach. *ACM Trans. Auton. Adapt. Syst.*, 2(4):13.

[Louis and Shankar, 2004] Louis, S. and Shankar, A. (2004). Context learning can improve user interaction. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 115–120.

[Luiz Lima and Calsavara, 2007] Luiz Lima, J. and Calsavara, A. (2007). A framework for CORBA interoperability in ad hoc networks. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 930–934, New York, NY, USA. ACM Press.

[Luttermann and Grauer, 1999] Luttermann, H. and Grauer, M. (1999). VRML history: storing and browsing temporal 3D-worlds. In *VRML '99: Proceedings of the fourth symposium on Virtual reality modeling language*, pages 153–160, New York, NY, USA. ACM.

[Malaka et al., 2004] Malaka, R., Haeussler, J., and Aras, H. (2004). SmartKom mobile: intelligent ubiquitous user interaction. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, pages 310–312, New York, NY, USA. ACM Press.

[Manninen, 2002] Manninen, T. (2002). Contextual Virtual Interaction as Part of Ubiquitous Game Design and Development. *Personal Ubiquitous Comput.*, 6(5-6):390–406.

[Marsden and Tip, 2005] Marsden, G. and Tip, N. (2005). Navigation control for mobile virtual environments. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 279–282, New York, NY, USA. ACM Press.

[Massó et al., 2006] Massó, J. P. M., Vanderdonckt, J., and López, P. G. (2006). Direct manipulation of user interfaces for migration. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 140–147, New York, NY, USA. ACM Press.

[Massó et al., 2005] Massó, J. P. M., Vanderdonckt, J., Simarro, F. M., and López, P. G. (2005). Towards virtualization of user interfaces based on UsiXML. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 169–178, New York, NY, USA. ACM Press.

[Merabti, 2006] Merabti, M. (2006). Networked appliances in home entertainment. In *CyberGames '06: Proceedings of the 2006 international conference on Game research and development*, pages 288–293, Murdoch University, Australia, Australia. Murdoch University.

[Mikroyannidis and Theodoulidis, 2006] Mikroyannidis, A. and Theodoulidis, B. (2006). Heraclitus II: A Framework for Ontology Management and Evolution. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 514–521, Washington, DC, USA. IEEE Computer Society.

[Mohomed et al., 2006] Mohomed, I., Cai, J. C., and de Lara, E. (2006). URICA: Usage-awaRe Interactive Content Adaptation for mobile devices. In *EuroSys '06: Proceedings of the 2006 EuroSys conference*, pages 345–358, New York, NY, USA. ACM Press.

[Mountain and Liarokapis, 2005] Mountain, D. and Liarokapis, F. (2005). Interacting with virtual reality scenes on mobile devices. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 331–332, New York, NY, USA. ACM Press.

[Mulloni et al., 2007] Mulloni, A., Nadalutti, D., and Chittaro, L. (2007). Interactive walkthrough of large 3D models of buildings on mobile devices. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 17–25, New York, NY, USA. ACM.

[Nixon et al., 2007] Nixon, L., Antonechko, O., and Tolksdorf, R. (2007). Towards semantic tuplespace computing: the semantic web spaces system. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 360–365, New York, NY, USA. ACM.

[Nurminen, 2007] Nurminen, A. (2007). Mobile, hardware-accelerated urban 3D maps in 3G networks. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 7–16, New York, NY, USA. ACM.

[O'Brien et al., 2007] O'Brien, L., Merson, P., and Bass, L. (2007). Quality Attributes for Service-Oriented Architectures. In *SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments*, page 3, Washington, DC, USA. IEEE Computer Society.

[Ohlenburg et al., 2007] Ohlenburg, J., Broll, W., and Lindt, I. (2007). DEVAL - A Device Abstraction Layer for VR/AR. *Universal Acess in Human Computer Interaction. Coping with Diversity*, pages 497–506.

[Oviatt, 2003] Oviatt, S. (2003). User-centered modeling and evaluation of multimodal interfaces. *Proceedings of the IEEE*, 91(9):1457–1468.

[Paelke et al., 2004] Paelke, V., Reimann, C., and Stichling, D. (2004). Foot-based mobile interaction with games. In *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 321–324, New York, NY, USA. ACM Press.

[Panayiotou, 2000] Panayiotou, C. (2000). Context Awareness. In *Proceedings of the Conference on Human Factors in Computing Systems*.

[Park and Han, 1997] Park, S. and Han, T. (1997). Object-Oriented VRML for multi-user environments. In *VRML '97: Proceedings of the second symposium on Virtual reality modeling language*, pages 25–32, New York, NY, USA. ACM.

[Pittarello and Faveri, 2006] Pittarello, F. and Faveri, A. D. (2006). Semantic description of 3D environments: a proposal based on web standards. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, pages 85–95, New York, NY, USA. ACM Press.

[Pousman et al., 2004] Pousman, Z., Iachello, G., Fithian, R., Moghazy, J., and Stasko, J. (2004). Design iterations for a location-aware event planner. *Personal Ubiquitous Comput.*, 8(2):117–125.

[Price et al., 2009] Price, S., Falc ao, T. P., Sheridan, J. G., and Roussos, G. (2009). The effect of representation location on interaction in a tangible learning environment. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 85–92, New York, NY, USA. ACM.

[Pulli et al., 2005] Pulli, K., Vaarala, J., Miettinen, V., Aarnio, T., and Callow, M. (2005). Developing mobile 3D applications with OpenGL ES and M3G. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 1, New York, NY, USA. ACM.

[Quillet et al., 2006] Quillet, J.-C., Thomas, G., Granier, X., Guitton, P., and Marvie, J.-E. (2006). Using expressive rendering for remote visualization of large city models. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, pages 27–35, New York, NY, USA. ACM.

[Ray, 2008] Ray, A. (2008). *The Interaction Framework For Innovation: A Method to Create Reusable Three-Dimensional Interaction Techniques*. PhD thesis, Virginia Polytechnic Institute and State University.

[Repo, 2004] Repo, P. (2004). Facilitating user interface adaptation to mobile devices. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 433–436, New York, NY, USA. ACM Press.

[Roberts et al., 2003] Roberts, D., Wolff, R., Otto, O., and Steed, A. (2003). Constructing a gazebo: supporting teamwork in a tightly coupled, distributed task in virtual reality. *Presence: Virtual Environvironments*, 12(6):644–657.

[Roibás et al., 2006] Roibás, A. C., Geerts, D., Furtado, E., and Calvi, L. (2006). Investigating new user experience challenges in iTV: mobility & sociability. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1659–1662, New York, NY, USA. ACM Press.

[Rowley, 2007] Rowley, J. E. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, pages 0165551506070706–.

[Roy et al., 2008] Roy, P., Abdulrazak, B., and Belala, Y. (2008). Approaching context-awareness for open intelligent space. In *MoMM '08: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, pages 422–426, New York, NY, USA. ACM.

[Rukzio et al., 2006] Rukzio, E., Paolucci, M., Finin, T., Wisner, P., and Payne, T. (2006). Mobile interaction with the real world. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 295–296, New York, NY, USA. ACM Press.

[Russell M. Taylor et al., 2001] Russell M. Taylor, I., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001). VRPN: a device-independent, network-transparent VR peripheral system. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61, New York, NY, USA. ACM.

[Salzberg, 1994] Salzberg, S. L. (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. *Machine Learning*, 16(3):235–240.

[Shankar and Louis, 2005] Shankar, A. and Louis, S. (2005). Learning classifier systems for user context learning. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2069–2075 Vol. 3.

[Shankar et al., 2007] Shankar, A., Louis, S. J., Dascalu, S., Hayes, L. J., and Houmanfar, R. (2007). User-context for adaptive user interfaces. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 321–324, New York, NY, USA. ACM.

[Simon et al., 2005] Simon, R., Wegscheider, F., and Tolar, K. (2005). Tool-supported single authoring for device independence and multimodality. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 91–98, New York, NY, USA. ACM Press.

[Smith et al., 2005] Smith, D., Morris, E., and Carney, D. (2005). Interoperability issues affecting autonomic computing. In *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, pages 1–3, New York, NY, USA. ACM Press.

[Stewart et al., 2007] Stewart, J. A., Dumoulin, S. J., and Noël, S. (2007). Binding external interactivity to X3D. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 109–112, New York, NY, USA. ACM.

[Stuerzlinger et al., 2006] Stuerzlinger, W., Chapuis, O., Phillips, D., and Roussel, N. (2006). User interface façades: towards fully adaptable user interfaces. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 309–318, New York, NY, USA. ACM Press.

[Suàrez et al., 2004] Suàrez, P. R., Jùnior, B. L., and de Barros, M. A. (2004). Applying knowledge management in UI design process. In *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*, pages 113–120, New York, NY, USA. ACM Press.

[Sun et al., 2006] Sun, C., Xia, S., Sun, D., Chen, D., Shen, H., and Cai, W. (2006). Transparent adaptation of single-user applications for multi-user real-time collaboration. *ACM Trans. Comput.-Hum. Interact.*, 13(4):531–582.

[Tomlinson et al., 2007] Tomlinson, B., Baumer, E., Yau, M. L., Alpine, P. M., Canales, L., Correa, A., Hornick, B., and Sharma, A. (2007). Dreaming of adaptive interface agents. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 2007–2012, New York, NY, USA. ACM Press.

[Tomlinson et al., 2005] Tomlinson, B., Yau, M. L., O'Connell, J., Williams, K., and Yamaoka, S. (2005). The Virtual Raft Project: A Mobile Interface for Interacting with Communities of Autonomous Characters. *CHI2005*.

[van Westrenen, 2004] van Westrenen, F. (2004). User centred interfaces in co-operative distributed systems. In *Proceedings of the conference on Dutch directions in HCI*, page 8, New York, NY, USA. ACM Press.

[van Wyk and de Villiers, 2008] van Wyk, E. and de Villiers, R. (2008). Usability context analysis for virtual reality training in south african mines. In *SAICSIT '08: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries*, pages 276–285, New York, NY, USA. ACM.

[Vandervelpen and Coninx, 2004] Vandervelpen, C. and Coninx, K. (2004). Towards model-based design support for distributed user interfaces. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 61–70, New York, NY, USA. ACM Press.

[Walczak and Cellary, 2002] Walczak, K. and Cellary, W. (2002). Building database applications of virtual reality with X-VRML. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology*, pages 111–120, New York, NY, USA. ACM.

[Wang and Mankoff, 2002] Wang, J. and Mankoff, J. (2002). Theoretical and architectural support for input device adaptation. *SIGCAPH Comput. Phys. Handicap.*, (73-74):85–92.

[Weber et al., 2005] Weber, M., Pfeiffer, T., and Jung, B. (2005). Pr@senZ - P@CE: mobile interaction with virtual reality. In *MobileHCI '05: Proceedings of the 7th international conference* *on Human computer interaction with mobile devices & services*, pages 351–352, New York, NY, USA. ACM Press.

[Weis et al., 2006] Weis, T., Saternus, M., Knoll, M., Brändle, A., and Combetto, M. (2006). Towards a general purpose user interface for service-oriented context-aware applications. In *CAI '06: Proceedings of the international workshop in conjunction with AVI 2006 on Context in advanced interfaces*, pages 53–55, New York, NY, USA. ACM.

[Wilson, 1995] Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.

[Witten and Frank, 2000] Witten, I. H. and Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.

[Xu et al., 2006] Xu, Y., Meng, X., Liu, W., and Xiang, H. (2006). A collaborative virtual environment for real time assembly design. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 373–376, New York, NY, USA. ACM Press.

[Yang et al., 2006] Yang, S., Huang, A., Chen, R., Tseng, S.-S., and Shen, Y.-S. (2006). Context Model and Context Acquisition for Ubiquitous Content Access in ULearning Environments. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 2, pages 78–83.

[Yee and Park, 2005] Yee, S. and Park, K. S. (2005). StudioBRIDGE: using group, location, and event information to bridge online and offline encounters for co-located learning groups. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 551–560, New York, NY, USA. ACM.

[Ying, 2006] Ying, J. (2006). An approach to Petri net based formal modeling of user interactions from X3D content. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, pages 153–157, New York, NY, USA. ACM Press.

[Zhang et al., 2006] Zhang, F., Song, Z., and Zhang, H. (2006). Web service based architecture and ontology based user model for cross-system personalization. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 849–852, Washington, DC, USA. IEEE Computer Society.

[Zhang and Gračanin, 2007] Zhang, X. and Gračanin, D. (2007). From coarse-grained components to DVE applications: a service- and component-based framework. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 113–121, New York, NY, USA. ACM.

[Zhao et al., 2006] Zhao, D., Grundy, J., and Hosking, J. (2006). Generating mobile device user interfaces for diagram-based modelling tools. In *AUIC '06: Proceedings of the 7th Australasian User interface conference*, pages 101–108, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Zhou and Houck, 2002] Zhou, M. X. and Houck, K. (2002). A semantic approach to the dynamic design of interaction controls in conversation systems. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA. ACM Press.

Appendices

Appendix A

System Analysis and Evaluation Data

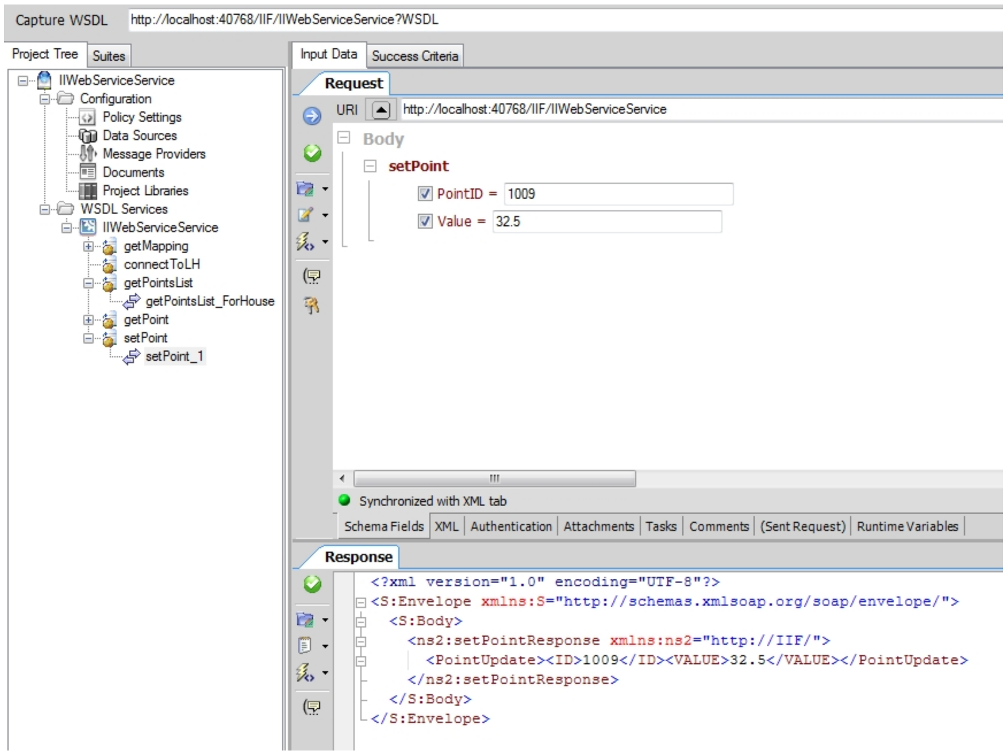
In this appendix there are several figures illustrating the web service evaluation that was done using SOAPSonar. The evaluation demonstrate that utilizing the framework services demanding procedures like context elicitation, database queries, forward chaining through the set of working rules and mapping derivation would not still be of a negative significant impact on a system.

In Figure A.1 a screen capture of a SOAPTest is shown. In creating a SOAP test, test data is supplied as input. The lower part of the screen shows the actual response that was sent by the framework web service. The response that was received will be used compare the other responses the test toll will receive during the stress test run. A perfect match would mean a success call and other mismatches means that the framework did not respond as expected implying a failed call.

In Figure A.2, the chart shows how running another daemon on the server that consumes a fixed amount of resources running a constant repetitive tasks did shift the delays but still it is noticed the framework delays are consistent even when having an overloaded machine.

Figure A.3 shows the results of having blocking calls to the LumenHaus controller. Since the calls block waiting for a response, with the controller down, this would give the perception that the web services are down as well. Another approach is to have non-blocking calls instead but this have got the risk of getting out of sync with calls being just queued and nothing actually happening.

In Figure A.4 a SOAPTest (call) of the highest computation power is stress tested, in addition to a another SOAPTest of the most computation intensive framework pure call (doesn't involve the controller). It is evident from the diagram that the peeks of the two calls are far from equal and that means that the delays incurred by integrating the framework does not have any negative significant impact over system proving the viability of integrating the framework to systems without a negative influence over performance.



http://localhost:40768/IIF/IIWebServiceService?WSDL

URI http://localhost:40768/IIF/IIWebServiceService

setPoint
PointID = 1009
Value = 32.5

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:setPointResponse xmlns:ns2="http://IIF/">
      <PointUpdate><ID>1009</ID><VALUE>32.5</VALUE></PointUpdate>
    </ns2:setPointResponse>
  </S:Body>
</S:Envelope>
```

Figure A.1: A SOAP test point created to be used in building a test suite on SOAPSonar [SOA, 2010]. Every reasonable effort has been made to inform the copyright owner of this fair use of their image.

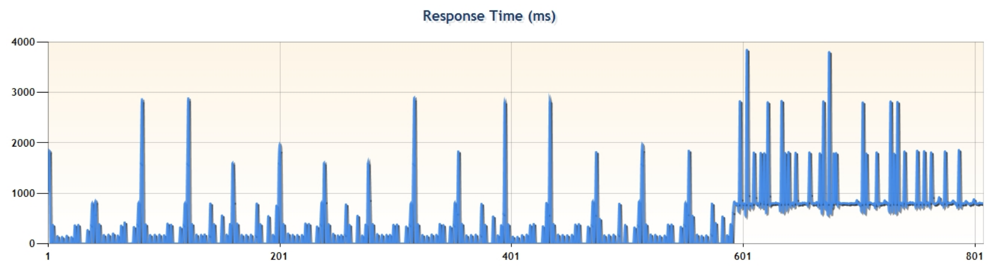


Figure A.2: Stress loading the server with other daemon tasks still shows consistency.

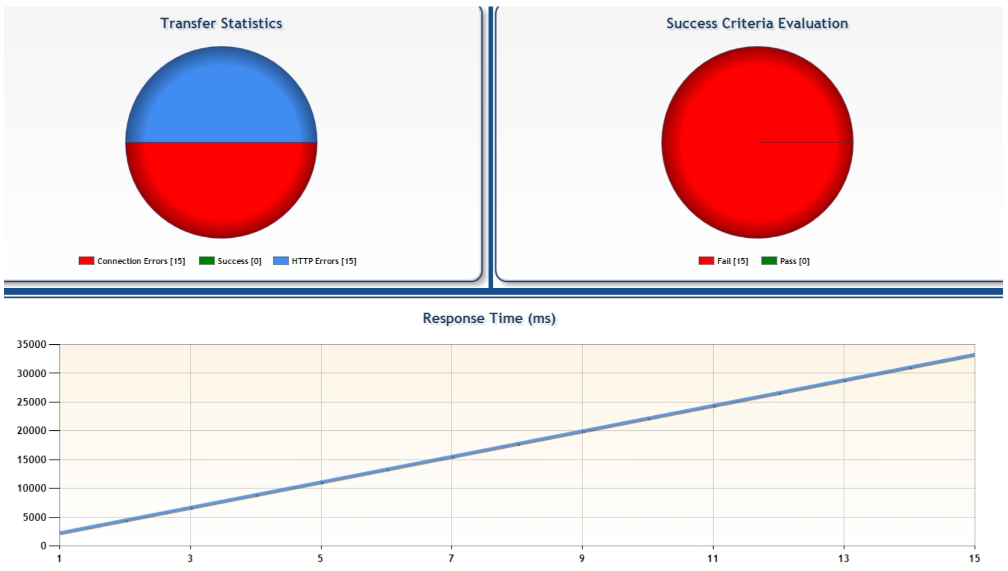


Figure A.3: Having blocking calls to the LumenHaus controller would give the perception that the web services are down if the controller failed.

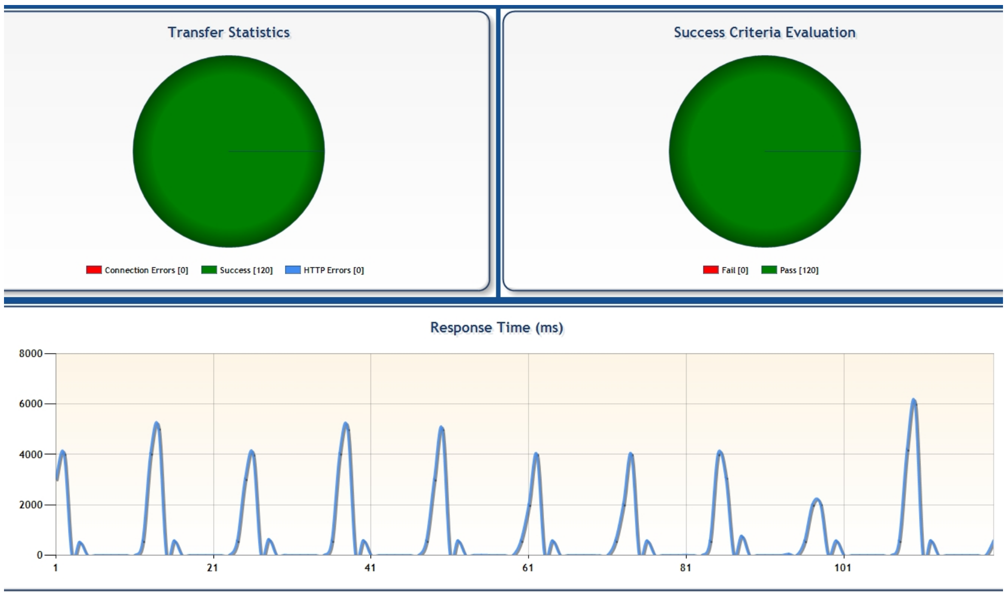


Figure A.4: Another test suite reflecting another scenario illustrating how minimal the framework delays are compared to other system inherent delays.

Appendix B

Code Snippets

In this appendix major code snippets illustrating the technicalities of implementation are listed with a brief explanation. Given that the implementation primarily includes Java web services and the .NET desktop application, there are code snippets from the Java code of the services implemented, the consuming desktop application as well as the XML SOAP messages exchanged.

B.1 Web Services

In this section some code snippets showing the functionality based on Web Services and their consumption are highlighted.

B.1.1 LumenHaus Liaison Functionality

In order to abstract the different controls available around the house, controlling the house is based upon the idea of having a set of control points. These points have some attributes that indicate the location, the type of the control point, is it analog or binary, the range of values and the current value set to the point. Listing B.1 shows how the liaison functionality between the framework and the house controller is built. Upon the initial start of the application and whenever a system wide update happens a complete list of control points are pulled by the application to know where the points should be located, what there types are and what is the current value.

```
public static String doList() {
    String ListOfPoints = new String();
    String SinglePoint= new String();
    ListOfPoints = "<PointsList>";
      for(Object o : attrMap.values()) {
          LumenHausAttribute attr = (LumenHausAttribute) o;
          SinglePoint =
            "<Point><ID>" + attr.id.trim() + "</ID>"+
            "<Name>" + attr.name.trim() + "</Name>"+
            "<Value>" + new Float(attr.value).toString().trim()+"</Value>"+
            "<Type>" + attr.type.trim()+"</Type>"+
            "<XCoord>" + new Float(attr.xCoord).toString().trim()+"</XCoord>"+
            "<YCoord>" + new Float(attr.yCoord).toString().trim()+"</YCoord>"+
            "</Point>";
           consoleLn(SinglePoint);
           ListOfPoints = ListOfPoints + SinglePoint;
        }
       ListOfPoints = ListOfPoints +  "</PointsList>";
       return ListOfPoints;
   }
```

Listing B.1: List house control points with all attributes.

Another liaison functionality is shown in listing B.2. Whenever a command is to be issued to the house the function is called with the point ID and the new value which is to be set.

```
public static String doSet(String id, String value) {
        char[] idChar = new char[4];
        id.getChars(0, 4, idChar, 0);

        Packet packet = new Packet();
        packet.setId(idChar);
        packet.setValue(new Float(value));
        String ConfirmMessage;
        try {
                out.write(packet.toDataPacket());
            ConfirmMessage = "<ID>"+id + "</ID><VALUE>" +value+"</VALUE>";
        } catch(IOException e) {
          ConfirmMessage = "<ERROR>Could␣not␣write␣value␣to␣server</ERROR>";
          consoleErr(ConfirmMessage);
        }
```

Listing B.2: Update Control Command with Confirmation or Error Message.

B.1.2 Web Service Operations

In this section the operations available by the framework liaison web services that wrap the house controller are shown in listings B.3 and B.4. In B.3 the house controller `get` operation is wrapped by this web service operation that makes the house accessible through interoperable SOAP calls. In B.4, the same idea is applied to allow controlling the house through the framework or any other

SOAP capable call.

```
@WebMethod(operationName = "getPointsList")
    @WebResult(name="PointsList")
    public String getPointsList() {
        if(connectedToLH == 0)
        {
            connectToLH("localhost",4001);
        }
      String ReturnThis = LH.doList();
      return ReturnThis;
    }
```

Listing B.3: Web Service Operation getPointsList.

```
 @WebMethod(operationName = "setPoint")
     @WebResult(name="PointUpdate")
    public String setPoint(@WebParam(name = "PointID")
    String PointID, @WebParam(name = "Value") String PointValue) {
    if(connectedToLH == 0)
        {
            connectToLH("localhost",4001);


        }
      String ReturnThis =  LH.doSet(PointID,PointValue);


      return ReturnThis;
    }
```

Listing B.4: Web Service Operation setPoint.

B.2 Desktop Application

In this section representative code is shown from the LumenHaus desktop application that was taking advantage of the context sensitive interaction interoperability functionality of the framework. Listing B.5 shows how the control points explained previously are pulled from the Web Service wrapper of the LumenHaus controller as an XML message that is automatically parsed by the .NET framework to extract the necessary parts from the XML tree retrieved.

```
IIService = new IIWebServiceService();
            responseString = IIService.getPointsList();
            xmlResponse = new XmlDocument();
            xmlResponse.LoadXml(responseString);
            nodeID = xmlResponse.GetElementsByTagName("ID");
            nodeName = xmlResponse.GetElementsByTagName("Name");
            nodeValue = xmlResponse.GetElementsByTagName("Value");
            nodeType = xmlResponse.GetElementsByTagName("Type");
            nodeXCoord = xmlResponse.GetElementsByTagName("XCoord");
            nodeYCoord = xmlResponse.GetElementsByTagName("YCoord");
            TotalPoints = nodeID.Count;
```

Listing B.5: Web Service Client Object Instantiation and a service operation call that retrieves and parses the XML response

Listing B.6 shows how the device mapping request is made. As shown in the code, an instance of the web service client object ,automatically created from the web service WSDL, is created and consumed returning back an XML message in a SOAP envelope. The code then builds two lists one for the input device components and the other is for the application tasks. Every index in these two arrays reflect the association set by the framework backend.

```
  // get device Mappings
            responseString = IIService.getMapping("1","1");
            xmlResponse = new XmlDocument();
            xmlResponse.LoadXml(responseString);
            taskID = xmlResponse.GetElementsByTagName("TaskID");
            inputID = xmlResponse.GetElementsByTagName("InputID");
            ControlMappings = taskID.Count;
```

Listing B.6: Request and Parse Device Mappings from the Framework Webservice

Listing B.7 illustrates how in the code the application does not assume any hard coding between device inputs and task execution. The function shown in listing B.7 shows how that executing a certain task will be done based on the XML associations received from the framework backend.

```
public void ExecuteTask(int TaskID)
   {
    switch(TaskID)
        {
         case 1: SelectPointControl("next"); break; // Next Point
         case 2: SelectPointControl("previous"); break; // Previous Point
         case 3: UpdatePointControl("up"); break; // Increase Value
         case 4: UpdatePointControl("down"); break; // Decrease Value
          }
   }
```

Listing B.7: Application decoupling task execution from input

Listing B.8 illustrates how device events are handled. To illustrate how nothing is hardcoded between the device events and the application execution, on the application side using the framework adapter or directly creating an event handler the handler checks for any raised events and going through the XML node list the code calls the function "*ExecuteTask*" with the task ID sent by the framework backend web service given the user, profile, current context, detected devices and the application is use.

Typically applications are built to handle certain events in certain ways and if user configuration is allowed then there would be a look up file to reflect that configuration. As shown in listing B.8 that is not the case, association and event to task execution mapping is done on the fly using the XML message sent in the SOAP envelope by the framework web service.

```
/// Go through Mappings
for (int i = 0; i < taskID.Count; i++)
   {
    int TaskNumber = int.Parse(taskID[i].InnerText);
    int DeviceNumber = int.Parse(inputID[i].InnerText);
    switch (DeviceNumber)
       {
        //// Keyboard Control
        case 1: if (keyboardState.IsKeyDown(Keys.Escape)) ExecuteTask(
           TaskNumber); break;//Right Key
        case 2: if (keyboardState.IsKeyDown(Keys.Right)) ExecuteTask(
           TaskNumber); break;//Right Key
        case 3: if (keyboardState.IsKeyDown(Keys.Left)) ExecuteTask(TaskNumber
           ); break;  //Left Key
        case 4: if (keyboardState.IsKeyDown(Keys.Up)) ExecuteTask(TaskNumber);
            break; //Up Key
        case 5: if (keyboardState.IsKeyDown(Keys.Down))ExecuteTask(TaskNumber)
           ; break; //Down Key
        case 6: if (keyboardState.IsKeyDown(Keys.Enter))ExecuteTask(TaskNumber
           ); break; //Enter Key

        ///// X-Box Controller  // Analog device to Binary task mapping
        case 7: if ( gamePadState.ThumbSticks.Left.X < -0.5) ExecuteTask(
           TaskNumber); break; //L-Analog Stick Left
        case 8: if ( gamePadState.ThumbSticks.Left.X > 0.5) ExecuteTask(
           TaskNumber); break; //L-Analog Stick Right
        case 9: if ( gamePadState.ThumbSticks.Left.Y > 0.5) ExecuteTask(
           TaskNumber); break; //L-Analog Stick Up
        case 10: if ( gamePadState.ThumbSticks.Left.Y < -0.5) ExecuteTask(
           TaskNumber); break; //L-Analog Stick Down
        case 11: if ( gamePadState.ThumbSticks.Right.X < -0.5) ExecuteTask(
           TaskNumber); break; //R-Analog Stick Left
        case 12: if ( gamePadState.ThumbSticks.Right.X > 0.5) ExecuteTask(
            TaskNumber); break; //R-Analog Stick Right
        case 13: if ( gamePadState.ThumbSticks.Right.Y < -0.5) ExecuteTask(
            TaskNumber); break; //R-Analog Stick Up
        case 14: if ( gamePadState.ThumbSticks.Right.Y > 0.5) ExecuteTask(
            TaskNumber); break; //R-Analog Stick Down
      .
      .
      .
       ////// WII MOTE //
         case 31: if ( wm.WiimoteState.ButtonState.Left) ExecuteTask(
             TaskNumber); break; //DPad  Left
         case 32: if (wm.WiimoteState.ButtonState.Right) ExecuteTask(
             TaskNumber); break; //DPad  Right
         case 33: if (wm.WiimoteState.ButtonState.Down) ExecuteTask(TaskNumber
             ); break; //DPad  Down
         case 34: if (wm.WiimoteState.ButtonState.Up) ExecuteTask(TaskNumber);
              break; //DPad  Up
         case 35: if (wm.WiimoteState.ButtonState.Minus) ExecuteTask(
             TaskNumber); break; //Minus Button
                 .
                 .
```

Listing B.8: Using the retrieved mapping to execute the respective task given event of interest

To highlight how an application can directly call other framework web services, like the context awareness web service, listing B.9 shows how the client application calls a function again of a web service client class object created automatically by the .NET framework based on the public WSDL file of the web service. The function *getDevicesAround* returns to the caller with an XML message with a list of devices (ID and Name) and this list can then be used in whatever way the application developer would like.

```
        // GET DEVICES AROUND //
    responseString = ContextWSClient.getDevicesAround(1);
```

```
xmlResponse = new XmlDocument();
xmlResponse.LoadXml(responseString);

availableDeviceID = xmlResponse.GetElementsByTagName("id");
availableDeviceName = xmlResponse.GetElementsByTagName("devicename");
TotalDevices = availableDeviceID.Count;
```

Listing B.9: Consuming web service to know what are the devices available around the machine being accessed.

In order to give more flexibility to client applications the Context Awareness web service has another publicly accessible operation named *getPriorityDevice*. The function call given the application ID and machine ID it returns back the top priority device to be used given who is the user accessing that machine, through this application and his/her current context. To be even more flexible and give application developers more control and leverage, not only the top prioritized device ID is sent, instead a complete prioritized list is sent with the most favored on top and the least near the bottom of the list. Application developers can devise this functionality in different ways and customize even the look, functionality and even theme of the application given the response received. Listing B.10 shows a sample operation call.

```
// GET USER PRIORITY DEVICES /////////////////////////////////////
    responseString = ContextWSClient.getPriorityDevice(1,1);
    xmlResponse = new XmlDocument();
    xmlResponse.LoadXml(responseString);
    priorityDeviceID = xmlResponse.GetElementsByTagName("id");
    priorityDeviceName = xmlResponse.GetElementsByTagName("devicename");
    priorityDeviceValue = xmlResponse.GetElementsByTagName("priority");
    priorityTotalDevices = priorityDeviceID.Count;
```

Listing B.10: Consuming web service to get priority device to be used given current context

B.3 Database

The database backend is an integral part of the framework. As stated in Chapter 7, the rules are basically database records that satisfies certain conditions and for larger scale, the rules working set is generated based again on the user interaction patterns and preferences retrieved through the database.

Below, listing B.11, is the SQL code that creates and fills in the tables with sample test data to have a working prototype.

```
/*
Navicat MySQL Data Transfer
Source Server         : localMySQL
Source Server Version : 50144
Source Host           : localhost:3306
Source Database       : iif_db
Target Server Type    : MYSQL
Target Server Version : 50144
File Encoding         : 65001
Date: 2010-04-18 02:08:28*/

SET FOREIGN_KEY_CHECKS=0;
-- ----------------------------
-- Table structure for `app_task`
-- ----------------------------
DROP TABLE IF EXISTS `app_task`;
CREATE TABLE `app_task` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `dimensions` int(10) DEFAULT NULL,
  `app_id` int(10) unsigned DEFAULT NULL,
 PRIMARY KEY (`id`),
 KEY `task_app_FK` (`app_id`),
 CONSTRAINT `task_app_FK` FOREIGN KEY (`app_id`) REFERENCES `application` (`
    id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of app_task
-- ----------------------------
INSERT INTO `app_task` VALUES ('1', 'Next Control Point', '1', '1');
INSERT INTO `app_task` VALUES ('2', 'Previous Control Point', '1', '1');
INSERT INTO `app_task` VALUES ('3', 'Increase Value', '1', '1');
INSERT INTO `app_task` VALUES ('4', 'Decrease Value', '1', '1');
INSERT INTO `app_task` VALUES ('5', 'Move Up in Menu', '1', '1');
INSERT INTO `app_task` VALUES ('6', 'Move Down in Menu', '1', '1');
INSERT INTO `app_task` VALUES ('7', 'Make Selection', '1', '1');
INSERT INTO `app_task` VALUES ('8', 'Exit', '1', '1');

-- ----------------------------
-- Table structure for `application`
-- ----------------------------
DROP TABLE IF EXISTS `application`;
CREATE TABLE `application` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `version` varchar(255) DEFAULT NULL,
  `apptype_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of application
-- ----------------------------
INSERT INTO `application` VALUES ('1', 'LumenHaus Desktop Controller', '1.0',
    '1');

-- ----------------------------
-- Table structure for `device`
-- ----------------------------
DROP TABLE IF EXISTS `device`;
CREATE TABLE `device` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of device
-- ----------------------------
INSERT INTO `device` VALUES ('1', 'Keyboard');
INSERT INTO `device` VALUES ('2', 'XBox');
INSERT INTO `device` VALUES ('3', 'Wii␣Mote');
INSERT INTO `device` VALUES ('4', 'Mouse␣2D␣-␣2␣Buttons');

-- ----------------------------
-- Table structure for `device_location_log`
-- ----------------------------
DROP TABLE IF EXISTS `device_location_log`;
CREATE TABLE `device_location_log` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `device_id` int(10) unsigned DEFAULT NULL,
  `loc_id` int(10) unsigned DEFAULT NULL,
  `stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
     CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `device_locationLog_FK` (`device_id`),
  CONSTRAINT `device_locationLog_FK` FOREIGN KEY (`device_id`) REFERENCES `
     device` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of device_location_log
-- ----------------------------
INSERT INTO `device_location_log` VALUES ('6', '1', '1', '2010-03-10␣04:49:36'
    );
INSERT INTO `device_location_log` VALUES ('7', '2', '1', '2010-04-17␣22:34:28'
    );
INSERT INTO `device_location_log` VALUES ('8', '3', '1', '2010-03-21␣16:28:31'
    );

-- ----------------------------
-- Table structure for `device_primitive`
-- ----------------------------
DROP TABLE IF EXISTS `device_primitive`;
CREATE TABLE `device_primitive` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `binary_continous` int(1) DEFAULT NULL,
  `dimensions` int(1) DEFAULT '1',
  `device_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `device_primitive_FK` (`device_id`),
  CONSTRAINT `device_primitive_FK` FOREIGN KEY (`device_id`) REFERENCES `
      device` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of device_primitive
-- ----------------------------
INSERT INTO `device_primitive` VALUES ('1', 'Escape', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('2', 'Right_Key', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('3', 'Left_Key', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('4', 'Up_Key', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('5', 'Down_Key', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('6', 'Enter_Key', '0', '1', '1');
INSERT INTO `device_primitive` VALUES ('7', 'Left␣Analog␣Stick␣To␣Left', '0',
    '1', '2');
INSERT INTO `device_primitive` VALUES ('8', 'Left␣Analog␣Stick␣To␣Right', '0',
     '1', '2');
INSERT INTO `device_primitive` VALUES ('9', 'Left␣Analog␣Stick␣To␣Up', '0', '1
    ', '2');
INSERT INTO `device_primitive` VALUES ('10', 'Left␣Analog␣Stick␣To␣Down', '0',
     '1', '2');
INSERT INTO `device_primitive` VALUES ('11', 'Right␣Analog␣Stick␣To␣Left', '0'
    , '1', '2');
INSERT INTO `device_primitive` VALUES ('12', 'Right␣Analog␣Stick␣To␣Right', '0
    ', '1', '2');
INSERT INTO `device_primitive` VALUES ('13', 'Right␣Analog␣Stick␣To␣Up', '0',
    '1', '2');
INSERT INTO `device_primitive` VALUES ('14', 'Right␣Analog␣Stick␣To␣Down', '0'
    , '1', '2');
INSERT INTO `device_primitive` VALUES ('15', 'Button␣A', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('16', 'Button␣B', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('17', 'Button␣X', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('18', 'Button␣Y', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('19', 'Button␣Back', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('20', 'Button␣Start', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('21', 'Button␣LB', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('22', 'Button␣RB', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('23', 'Button␣LT', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('24', 'Button␣RT', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('28', 'DPad␣Left', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('29', 'DPad␣Right', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('30', 'DPad␣Up', '0', '1', '2');
INSERT INTO `device_primitive` VALUES ('31', 'DPad␣Down', '0', '1', '2');


-- --------------------------------------------------------
-- Table structure for `ii_user`
-- --------------------------------------------------------
```

```
DROP TABLE IF EXISTS `ii_user`;
CREATE TABLE `ii_user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `userpass` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of ii_user
-- ----------------------------
INSERT INTO `ii_user` VALUES ('1', 'Hussein Ahmed', '123', 'hmahmed@vt.edu');
INSERT INTO `ii_user` VALUES ('2', 'Samah Gad', '123', 'samah@vt.edu');
INSERT INTO `ii_user` VALUES ('3', 'Denis Gracanin', '123', 'gracanin@vt.edu')
    ;

-- ----------------------------
-- Table structure for `loc_type`
-- ----------------------------
DROP TABLE IF EXISTS `loc_type`;
CREATE TABLE `loc_type` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of loc_type
-- ----------------------------
INSERT INTO `loc_type` VALUES ('1', 'Office Room');
INSERT INTO `loc_type` VALUES ('2', 'Home');
INSERT INTO `loc_type` VALUES ('3', 'Cafe');
```

```
-- ----------------------------
-- Table structure for `location`
-- ----------------------------
DROP TABLE IF EXISTS `location`;
CREATE TABLE `location` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) DEFAULT NULL,
  `normx` int(10) DEFAULT NULL,
  `normy` int(10) DEFAULT NULL,
  `type_id` int(10) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of location
-- ----------------------------
INSERT INTO `location` VALUES ('1', 'My␣Office', '2', '2', '1');
INSERT INTO `location` VALUES ('2', 'My␣Room␣at␣home', '50', '50', '2');
INSERT INTO `location` VALUES ('3', 'Starbucks', '100', '100', '3');

-- ----------------------------
-- Table structure for `machine_location`
-- ----------------------------
DROP TABLE IF EXISTS `machine_location`;
CREATE TABLE `machine_location` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `loc_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `machine_location_FK` (`loc_id`),
  CONSTRAINT `machine_location_FK` FOREIGN KEY (`loc_id`) REFERENCES `location
      ` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

```
-- ------------------------------
-- Records of machine_location
-- ------------------------------
INSERT INTO `machine_location` VALUES ('1', 'Hussein␣MacBook␣Pro', '1');

-- ------------------------------
-- Table structure for `primitive_dimension`
-- ------------------------------
DROP TABLE IF EXISTS `primitive_dimension`;
CREATE TABLE `primitive_dimension` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `min_value` int(10) DEFAULT NULL,
  `max_value` int(10) DEFAULT NULL,
  `increment` int(10) DEFAULT NULL,
  `ideal_value` int(10) DEFAULT NULL,
  `primitive_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `primitive_dimensoin_FK` (`primitive_id`),
  CONSTRAINT `primitive_dimensoin_FK` FOREIGN KEY (`primitive_id`) REFERENCES
      `device_primitive` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ------------------------------
-- Records of primitive_dimension
-- ------------------------------

-- ------------------------------
-- Table structure for `task_dimension`
-- ------------------------------
DROP TABLE IF EXISTS `task_dimension`;
CREATE TABLE `task_dimension` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT '',
  `binary_continous` int(1) unsigned DEFAULT '1',
  `min_value` int(10) DEFAULT NULL,
  `max_value` int(10) DEFAULT NULL,
  `apptask_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `task_dimension_FK` (`apptask_id`),
  CONSTRAINT `task_dimension_FK` FOREIGN KEY (`apptask_id`) REFERENCES `
      app_task` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of task_dimension
-- ----------------------------
INSERT INTO `task_dimension` VALUES ('1', 'main', '0', null, null, '1');
INSERT INTO `task_dimension` VALUES ('2', 'main', '0', null, null, '2');
INSERT INTO `task_dimension` VALUES ('3', 'main', '1', '0', '100', '3');
INSERT INTO `task_dimension` VALUES ('4', 'main', '1', '0', '100', '4');
INSERT INTO `task_dimension` VALUES ('5', 'main', '0', null, null, '5');
INSERT INTO `task_dimension` VALUES ('6', 'main', '0', null, null, '6');
INSERT INTO `task_dimension` VALUES ('7', 'main', '0', null, null, '7');
INSERT INTO `task_dimension` VALUES ('8', 'main', '0', null, null, '8');

-- ----------------------------
-- Table structure for `user_device`
-- ----------------------------
DROP TABLE IF EXISTS `user_device`;
CREATE TABLE `user_device` (
  `userid` int(10) NOT NULL DEFAULT '0',
  `deviceid` int(10) NOT NULL DEFAULT '0',
  `locationid` int(10) NOT NULL DEFAULT '0',
  `priority` int(10) DEFAULT '1',
  PRIMARY KEY (`userid`,`deviceid`,`locationid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- ----------------------------
-- Records of user_device
-- ----------------------------
INSERT INTO `user_device` VALUES ('1', '1', '1', '1');
INSERT INTO `user_device` VALUES ('1', '2', '1', '5');
INSERT INTO `user_device` VALUES ('1', '3', '1', '8');
INSERT INTO `user_device` VALUES ('2', '1', '1', '1');
INSERT INTO `user_device` VALUES ('2', '2', '1', '1');
INSERT INTO `user_device` VALUES ('2', '3', '1', '5');

-- ----------------------------
-- Table structure for `user_location_log`
-- ----------------------------
DROP TABLE IF EXISTS `user_location_log`;
CREATE TABLE `user_location_log` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` int(10) unsigned DEFAULT NULL,
  `loc_id` int(10) unsigned DEFAULT NULL,
  `stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
     CURRENT_TIMESTAMP,
 PRIMARY KEY (`id`),
 KEY `loc_user_FK` (`user_id`),
 KEY `loc_location_FK` (`loc_id`),
 CONSTRAINT `loc_location_FK` FOREIGN KEY (`loc_id`) REFERENCES `location` (`
     id`),
 CONSTRAINT `loc_user_FK` FOREIGN KEY (`user_id`) REFERENCES `ii_user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of user_location_log
-- ----------------------------
INSERT INTO `user_location_log` VALUES ('6', '1', '2', '2010-04-17␣16:24:37');
INSERT INTO `user_location_log` VALUES ('7', '2', '1', '2010-03-10␣11:54:27');
INSERT INTO `user_location_log` VALUES ('8', '3', '1', '2010-03-10␣11:53:23');
```

```
-- ----------------------------
-- Table structure for `user_machine`
-- ----------------------------
DROP TABLE IF EXISTS `user_machine`;
CREATE TABLE `user_machine` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` int(10) unsigned DEFAULT NULL,
  `machine_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `machineowner_userid_FK` (`user_id`),
  KEY `machineowner_machineid_FK` (`machine_id`),
  CONSTRAINT `machineowner_machineid_FK` FOREIGN KEY (`machine_id`) REFERENCES
      `machine_location` (`id`),
  CONSTRAINT `machineowner_userid_FK` FOREIGN KEY (`user_id`) REFERENCES `
     ii_user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of user_machine
-- ----------------------------
INSERT INTO `user_machine` VALUES ('1', '1', '1');
INSERT INTO `user_machine` VALUES ('4', '2', '1');
```

Listing B.11: Consuming web service to get priority device to be used given current context