EGVE Symposium (2008)
B. Mohler and R. van Liere (Editors)

An Approach to Interaction Interoperability for Distributed Virtual Environments

Hussein M. Ahmed[1], Denis Gračanin[1], Ayman Abdel-Hamid[2] and Krešimir Matković[3]

[1]Department of Computer Science, Virginia Tech, Blacksburg, VA 24060, USA
[2]College of Computing, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt
[3]VRVis Center, Vienna, Austria

Abstract

We present a preliminary framework for interaction interoperability in Distributed Virtual Environments (DVEs). The goal is to allow each user to use a different input devices and interaction techniques and yet collaborate seamlessly. The framework adopts Service Oriented Architecture (SOA) and use a knowledge base in the form of three ontologies. The ontologies use Web Ontology Language (OWL) to describe input devices, interaction techniques and interaction tasks. In addition, two directories contain application and user profiles. An inference engine searches for the best possible combination of input devices, interaction techniques and tasks. The resulting user's mapping file is then accessed using web services. Two proof-of-concept framework implementations demonstrate how to develop a new application and how to support a legacy application.

Categories and Subject Descriptors (according to ACM CCS): H.5.3 [Group and Organization Interfaces]: Collaborative computing I.3.6 [Methodology and Techniques]: Interaction techniques H.5.2 [User Interfaces]: User-centered design

1. Introduction

Today there are many successful Distributed Virtual Environments (DVEs) in the areas of entertainment, education and business. However, collaboration in such DVEs is still somehow hindered by compatibility issues among interaction devices and techniques. A user is obliged to use certain input devices — mostly a keyboard and a mouse — and very specific interaction techniques. Such a limitation can adversely affect different DVE characteristics such as performance, usability and even joyfulness [BSC*97].

In this paper we present a preliminary framework (reference implementation completed and tested), with the goal of facilitating *interaction interoperability*. Having this achieved, DVE users would have the freedom of using their own input device and their preferred interaction technique to collaborate with other users and yet accomplishing tasks afforded by the environment.

Our main contribution, in contrast to previous work, is that we are not concerned with how to categorize devices and techniques, but how to choose the best ones for the user. Another major difference is that we are aiming for a distributed system that could be accessed by any platform, catering not only applications aware of the framework, but also legacy applications built just for WIMP interfaces.

2. Related Work

In their taxonomy of input devices, Foley et. al. [FWC84] classified input devices based on their physical structure. Buxton [Bux83] added lexical and pragmatic considerations indicating which device is used to measure rather than how the device is built.

Card et. al. [CMR90] provide a *design space* for input devices identifying each by a six-tuple record with the possibility to represent discrete devices. Bleser and Sibert in [BS90] developed an AI application in Smalltalk holding a class hierarchy of devices defining input and output domains, physical actions afforded, and their physical packages.

Jacob and Sibert [JS92] proposed discrimination between integral devices with movement is in Euclidean space, cutting across all the dimensions of control, and separable de-

vices that constrains movement to a stair-step pattern; moving along one dimension at a time.

Concerning taxonomies of interaction techniques and interaction tasks in virtual environments, Bowman et al. [BKLP05] classified both interaction techniques and tasks. The taxonomies were indeed exhaustive and up-to-date but the question is how should we store and handle these taxonomies making it possible to accommodate newly proposed interaction techniques and tasks.

For the area dealing with adapting interfaces to input devices, there is a misconception found between the terms "retargeting" and "interoperability". Most of the work done was in trying to build toolkits that could be used in building applications to support different input devices. Figueroa et al. [FGH02] proposed InTml (Interaction Techniques Markup Language) which was built to ease the **development** of VE applications with a platform independent set of reusable components. Dachselt et al. [DR03] developed an extension specifically for X3D again to facilitate **building** applications that would take advantage of an extended set of behaviors and sensors not supported.

A survey of user modeling is provided by Zhang et. al. [ZSZ06]. They classified user models and proposed an architecture that makes use of the Service Oriented Architecture (SOA) to offer cross-system personalization in which user profiles are stored globally and accessed by all web sites rather than making the user re-enter all his data. What should be highlighted is that existing models cannot be used to infer what device a user would favor for an application nor what interaction technique to offer for a certain device in order to elevate performance and comfort.

3. Proposed Framework

Our framework is based on the idea of enabling long-term interaction interoperability using a standardized taxonomy expressed using the OWL Web Ontology Language [WWWC08]. The framework is built upon the SOA maximizing possibilities of access and interoperability.

An abstraction of the framework architecture is shown in Figure 1. One of the main components of the system is the knowledge base (discussed later). The inference engine is responsible for matching available input devices to suitable interaction techniques to required application tasks. The distributed interface is provided as a Web Service.

Two possible client types are considered. The first type includes applications aware of our framework making direct requests to the system web service. The second type includes legacy applications using just a certain input device, and in this case, the user would have to install a framework wrapper component that mediates communication with web services and emulates input devices for the legacy application.

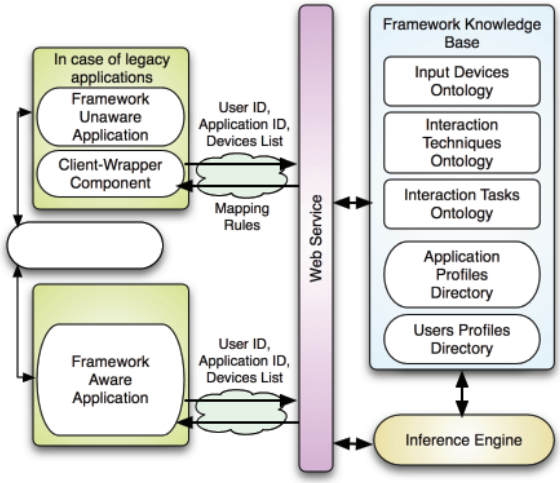
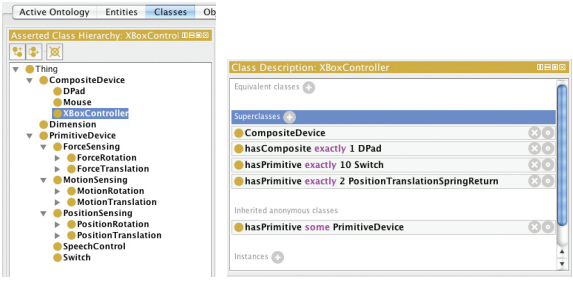


Figure 1: *Framework architecture.*

3.1. Knowledge Base

The knowledge base consists of three ontologies (input devices, interaction techiniques, interaction tasks), the application profile directory and the user profile directory. In building the taxonomies, previous taxonomies were taken into consideration including benefits and avoiding, were possible, weaknesses.

Given the three sublanguages of OWL (OWL-Lite, OWL-DL and OWL-Full) we have selected OWL-DL. The reason is that OWL-DL is much more expressive given **D**escription **L**ogics that enable automated reasoning making it possible to automatically classify and check for inconsistencies.



(a) Class Hierarchy (b) The XBox Controller Class

Figure 2: *OWL Input Devices Ontology showing the XBox Controller.*

For input devices, the taxonomy was mainly based upon the work of Jacob et al. [JSMMPM94] and Card et al. [CMR90]. Something to note however is despite the similarity to the work of Bleser and Sibert [BS90], their extended breakdown of the hierarchy would make automatic classification and inference difficult to achieve.

Figure 2 illustrates a part of the ontology showing how a composite device like the XBox 360 Controller looks like. The controller is a child of **CompositeDevice** class and has the following properties: It has **exactly one** composite device of class *DPad*, It has **exactly 10** primitive devices of class *Switch*, It has **exactly two** primitive devices of class *Position-Translation-SpringReturn* and it inherits the property — must have **at least one** primitive device.

Bowman et al. [BKLP05] proposed taxonomies for both interaction techniques and tasks decomposed by task type limiting inference capabilities. We decided to have two different taxonomies related to each other but the one for interaction techniques is divided pragmatically. This classification was also mentioned by Bowman et. al. in [BKLP05] but we extended it to include other currently available techniques and the capability of extending the taxonomy to accommodate newly created ones while maintaing consistency.

Regarding user profiles, we handle profiling in a novel way by taking record of *user interaction patterns*. These patterns show how the user preferred to play a certain game or operate a certain application. The available devices at the time are recorded, the selected device, the interaction technique favored and what tasks were tied to this combination. Recording such patterns, whenever a user accesses the framework again, his profile will be retrieved and new adaptations could be inferred.

Profiling applications involves knowing all the tasks afforded. In [ZG07], Zhang and Gračanin proposed a service and component based framework that could be used to build complete DVEs from coarse components. To enable integration, each component includes a behavior description file that defines many things including expected inputs. Other toolkits also based on the component based approach mostly offer such descriptions. Description files are parsed and a permanent, versioned profile of the application is created once and stored for later reference.

3.2. Service Oriented Architecture

The SOA architectural style promotes loose coupling between components elevating reusability and promises interoperability between heterogeneous technologies. Another point to note is that services could be discovered if registered with a global service directory.

The major service provided by the framework is the input device mapping request. The framework also provides, through the web service interface manipulation and retrieval of information stored in the ontologies and directories.

As noted previously, there are two types of service requesters. The first type includes applications aware of the framework with hard coded requests to such services. The second type includes client wrappers that run on the client side to facilitate interaction interoperability for legacy applications (Section 3.3). Information sent by the requester to the web service includes the user ID, the application ID and the list of available devices. Getting the request, the service communicates with the inference engine supplying it with the respective IDs to look up the profiles and "reasons" with the provided knowledge base using a set of inference rules to make up the most suitable mapping.

(a) Input device selection. (b) A student playing the game using the accelerometer of the Wii controller.

Figure 3: *A preliminary implementation of the proposed framework using a simple game with three atomic tasks (move left/right/forward) and three input devices (keyboard, Wii remote controller, and XBox 360 controller).*

3.3. Client-side Wrapper

The client-side wrapper provides mapping between innovative input devices and the traditional WIMP devices. The need for that is to support legacy applications unaware of the framework and its published services.

There have been several projects trying to hard code the usage of new devices in WIMP based applications. In our approach, legacy adaptation uses input devices **emulation**. Signals are read from input devices not supported by the legacy application and based on the mapping rules retrieved from the framework, events are emulated as if a keyboard key was pressed/released or the mouse has moved. Having the wrapper, literally "any" application could be controlled with any input device.

4. Implementation

As a proof-of-concept we built two implementations. The first was a game, hard coded supporting the Wii and the XBox controllers in a number of interaction techniques, in addition to the keyboard. The game was a single player game to test for the acceptance of the interaction interoperability concept.

Ten subjects were asked to play using the different input devices and once accustomed, they were asked to select one device to play a timed game. A questionnaire was given asking about their first and second favorite controllers and their

quantitative opinion of interaction interoperability. The collected results, observations, and questionnaires filled showed a significant acceptance of the concept and validated our hypothesis that no single input device can be considered the best for all users. Figure 3 shows a screen from the game and a student playing during the study.

The second implementation supports *legacy* DVEs unaware of the framework. As an example we used MPK20 [Sun08]. The client wrapper had the job of mapping between both the Wii and the XBox controller to the keyboard key presses expected by the DVE client. We tested two users collaborating in MPK20, with the wrapper running locally on each host, each user was able to use a different input device.

5. Conclusion and Future Work

The presented approach and the developed framework provide well designed OWL ontologies. User profiling, compared to the traditional user modeling, is a novel addition that saves users the effort to customize applications to devices and interaction techniques they prefer.

We are not aiming for a development toolkit to build new applications from scratch. Instead, the framework is based on the SOA maximizing interoperability to an extent that any entity could request services. Legacy applications unaware of the available services can be incorporated using the appropriate client wrappers as well.

We continue working on the framework to refine the knowledge base even more and provide additional services. We are working on a tighter integration with eXtensible 3D (X3D) standard [W3DC08] in order to enable more powerful interaction interoperability focused more on the interaction techniques than devices.

References

[BKLP05] BOWMAN D., KRUIJFF E., LAVIOLA J., POUPYREV I.: *3D User Interfaces Theory and Practice*. Addison Wesley, 2005.

[BS90] BLESER T. W., SIBERT J.: Toto: a tool for selecting interaction techniques. In *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology* (New York, NY, USA, 1990), ACM, pp. 135–142.

[BSC*97] BENFORD S., SNOWDON D., COLEBOURNE A., O'BRIEN J., RODDEN T.: Informing the design of collaborative virtual environments. In *GROUP '97: Proceedings of the international ACM SIGGROUP conference on Supporting group work* (New York, NY, USA, 1997), ACM Press, pp. 71–80.

[Bux83] BUXTON W.: Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph. 17*, 1 (1983), 31–37.

[CMR90] CARD S. K., MACKINLAY J. D., ROBERTSON G. G.: The design space of input devices. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1990), ACM, pp. 117–124.

[DR03] DACHSELT R., RUKZIO E.: Behavior3D: an XML-based framework for 3D graphics behavior. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology* (New York, NY, USA, 2003), ACM, pp. 101–ff.

[FGH02] FIGUEROA P., GREEN M., HOOVER H. J.: InTml: a description language for VR applications. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology* (New York, NY, USA, 2002), ACM, pp. 53–58.

[FWC84] FOLEY J., WALLACE V., CHAN P.: The human factors of computer graphics interaction techniques. In *IEEE Computer Graphics and Applications* (November 1984), vol. 4(11), pp. pp. 13–48.

[JS92] JACOB R. J. K., SIBERT L. E.: The perceptual structure of multidimensional input device selection. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1992), ACM, pp. 211–218.

[JSMMPM94] JACOB R. J. K., SIBERT L. E., MCFARLANE D. C., M. PRESTON MULLEN J.: Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact. 1*, 1 (1994), 3–26.

[Sun08] SUN MICROSYSTEMS, INC.: MPK20: Sun's virtual workplace. http://www.sunlabs.com/projects/mc/mpk20.html [Last accesed April 17, 2008].

[W3DC08] WEB3D CONSORTIUM: X3D and related specifications. http://www.web3d.org/ [Last accesed April 17, 2008].

[WWWC08] WORLD WIDE WEB CONSORTIUM: OWL web ontology language. http://www.w3.org/TR/owl-guide/ [Last accessed April 17, 2008].

[ZG07] ZHANG X., GRAČANIN D.: 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* (New York, NY, USA, 2007), ACM, pp. 113–121.

[ZSZ06] ZHANG F., SONG Z., ZHANG H.: 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* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 849–852.