Service-oriented Platform for Virtual Reality Application Development

Evandro César Freiberger[1,2], Ricardo Nakamura[1] and Romero Tori[1]
[1]*PCS/POLI/University of São Paulo, São Paulo-SP, Brazil*
[2]*DAI/Federal Institute of Mato Grosso, Cuiabá-MT, Brazil*

Keywords: Virtual Reality, Reuse, Interactive Content, Service-oriented Computing.

Abstract: An important contemporary fact is the advent of Web 2.0, mainly characterized by the possibility of content being produced collaboratively, empowering and enabling the concept of collective intelligence. Another important feature is the popularity of virtual communities, which allow people around the world to exchange information and experiences. In order to increase the potential for reuse and sharing of interactive content, this paper proposes an architectural model for a software platform that enables the online production and execution of virtual reality applications as well as sharing of interactive content. The remote online environment, associated with the capacity to represent and store 3D interactive content, enables the sharing and reuse of such content in the production of virtual reality applications. The paper we present the description of the main elements of the proposed architectural model and results of the research. The results point to the feasibility of the proposed model.

1 INTRODUCTION

Web 2.0 has changed the traditional model of information publishing on the Web to a model of collaborative information production. Nowadays people extensively use Web 2.0 applications such as Wikipedia, YouTube, LinkedIn, MySpace, Twitter, Facebook and Google Applications to create and share information. Web 2.0 has changed the way people consume information and knowledge into a collaborative approach (Bein et al., 2009).

Virtual reality (VR) is used in the production of complex virtual environments (VE), using non-trivial input and output devices to provide users with a sense of immersion in real-time synthetic worlds. In the particular case of software development for VR application, the need for reuse is enhanced by factors characteristic of this kind of application.

A predominant feature of VR systems is the dependence on computing resources and unconventional devices, such as special data input devices, special viewing equipment, along with the high computational power, especially with graphic processing. New techniques have alleviated this problem through the development of interactions that exploit gestures and sound commands, as an alternative control of virtual environments.

In order to reduce dependence on special input and output devices, computational resources such as graphic processing and enable the production of VR applications in a collaborative production environment, this paper proposes a software platform that combines the features of Web 2.0 and the of service-oriented computing model. The goal is to enable the production, reuse and execution of VR application elements in a distributed online environment.

2 BACKGROUND

2.1 Virtual Reality

With the development of VR and the advancement of computational resources, interactive and immersive representations become easier to obtain. The interfaces are more intuitive and break the boundaries of computer screens, keyboard and mouse, allowing users to act in a three-dimensional space.

VR is an advanced interface for computer applications will be considered. It allows the user to move (navigation) and interact in real-time in a three-dimensional environment, making use of multi-sensory devices, to act or feedback (Tori and Kirner, 2006).

With VE created from the use of VR, humans

Freiberger E., Nakamura R. and Tori R..
Service-oriented Platform for Virtual Reality Application Development.
DOI: 10.5220/0004895301960203
In *Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2014), pages 196-203
ISBN: 978-989-758-030-7

senses and abilities can be expanded, permitting to see, hear, feel, drive and travel far beyond the natural capacity. It creates alternatives for the production of applications related to entertainment, simulations, training and education.

2.2 Service-oriented Computing

Service-oriented computing (SOC) involves concepts originating from a variety of disciplines, such as distributed computing systems, computer architectures and middleware, grid computing, software engineering, programming languages, database systems, security and knowledge representation (Papazoglou and Heuvel, 2007).

Currently, the technical solution mostly adopted for the development of services-oriented computing is Web Services (Erl, 2009), (Papazoglou et al., 2008). The strong adoption of Web Services is a result of its characteristics, among which we can highlight: platform independence and programming languages, the possibility of exposing any application functionality as a service over the Internet and the use of open standards. According to Wang and Qian (Wang and Qian, 2005), implementing distributed computing via the Web Services technology, has the following advantages: increases the portability and interoperability of distributed computing; increases the reusability and scalability of distributed components; reduces the complexity of composition and deployment of components; simplifies management of the distributed system and facilitates the publication of the legacy code through distributed service interfaces.

Another important factor for the adoption of Web Services is the existence of a specification controled by a consortium which makes it less vulnerable to particular issues of either implementation (Papazoglou and Heuvel, 2007).

3 RELATED WORK

To support the objectives of this research, we conducted a literature review aimed at identifying software platforms that support the development of VR applications, with emphasis on the reuse of software components, in particular using the services-oriented computing approach.

The authors Zang and Gracanin (Zhang and Gracanin, 2008), propose a framework to build applications on a multi-user virtual environment, integrating content via distributed services. In addition, they propose the use of stream for applications feedback transfer to overcome the performance limitations of Web Services messages. The overall architecture of the framework is based on distributed components integrated through services and execution control based on events.

Shao and McGraw (Shao and McGraw, 2009) proposed a framework named Service-Oriented Embedded-Simulation Software (SOESS), which combines COS and Cloud Computing to produce military simulation, through the components composition. The goal of the framework is to ensure that applications developed from it are highly interoperable with other applications, systems and platforms, particularly with legacy software. The components communicate via Web Services, ensuring greater interoperability.

The authors Filho *et al.* (Filho et al., 2011) propose a platform for development of virtual environments called Hydra. Hydra is composed of a set of frameworks and tools with the purpose of facilitating and accelerating the applications development. The applications development is accomplished by means of plugins that allow you to customize the behavior of particular applications.

The work of Chevaillier *et al.* (Chevaillier et al., 2012) is proposed a methodology and a framework to design semantic VR environments. The development of VR applications is done through an approach based on models created with a specialization and extension of the Unified Modeling Language (UML). This modeling covers aspects of semantic representation of VE, such as: domain ontology, the structure of the environment, the behavior of entities and interactions between agents and activities.

The vast majority of studies analyzed propose the reuse of software elements for the development of VR applications through the incorporation of portions of specialized code, for example, using API components or the extent of frameworks. This paper proposes a model of representation of applications that enables the production of VR applications with high-level representation, without the need for coding and compiling. The difference lies in the ability to provide such functionality through services. This enables different technologies (programming language and execution platform) to be used for the development of applications production and running environments.

4 ARCHITECTURAL MODEL

In this work, we use the paradigm of service-oriented computing. This model is capable of being totally produced and executed in a remote environment. The

specified platform is characterized as a service bus that provides the capacity to produce and execution VR applications, in a distributed online environment.

4.1 Subsystems Architecture

The logical view of the platform elements, illustrated in the Figure 1-B, consists of four subsystems and their dependency relationships. The following subsystems are described:

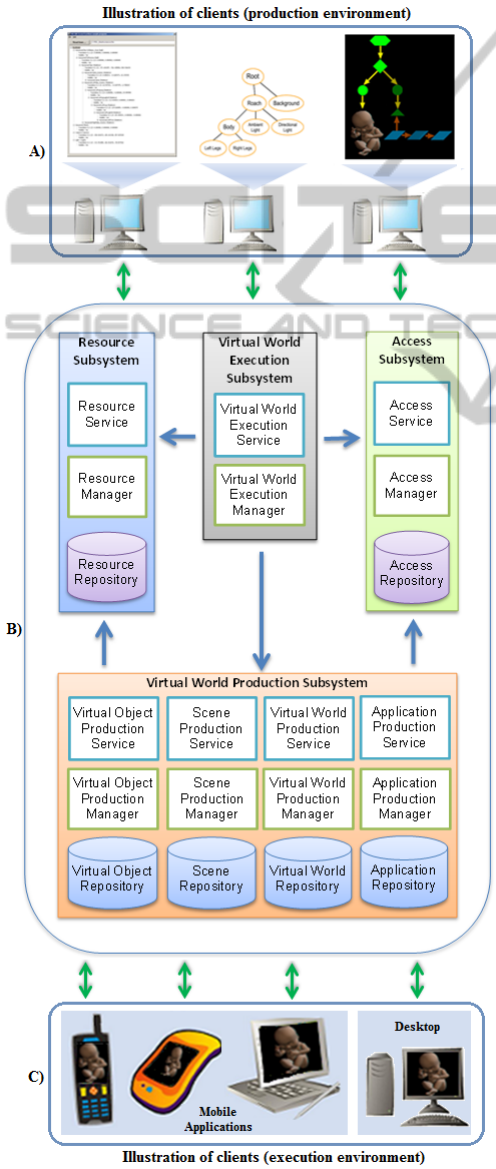

Figure 1: Platform Logical View.

- *Virtual World Production Subsystem* - has the functionality to create, update, delete and reuse elements of VR applications. A VR application is composed of one virtual world, one or more scenes, which are composed of virtual objects, behaviors and appearance characteristics;
- *Virtual World Execution Subsystem* - provides an execution context for the applications produced on the platform. An execution context consists of user context, input devices, output devices, a VR application and resources;
- *Access Subsystem* - gathers the features that enable access control and authorization for platform users. Besides the management of users, the subsystem provides authentication and verification services of authorizations to other subsystems of the platform;
- *Resource Subsystem* - gathers management features, searches for and rescues resources used in the production and execution of VR applications such as: images, sounds, videos, 3D object models and textures.

As can be seen in Figure 1-B, the production subsystem uses the services of the access and resource subsystems. It is also possible to observe that the execution subsystem uses the services of the resource, access and production subsystems. The subsystems were designed with this distribution of responsibilities to allow deployment in different servers, with adequate computing resources according to the characteristics required by each subsystem.

Figure 1-A illustrates the client-side of the platform, potentially consisting of several production environments of VR applications. The client applications use Web Services to create, seek and save elements of applications in the repositories of the platform. Each production environment can use different representations and models of interaction with the end user.

Communication between production environments (client-side) and the platform (server-side) is performed by means of Web Services, through which data (entities) that represent resources (e.g. images, sounds, textures), virtual objects, scenes or virtual worlds are sent and received.

Figure 1-C illustrates the client-side of the platform responsible for the interaction and viewing of the server-side execution. The execution environments (client-side) capture data and events from input devices and send them through Web Services to the execution context of the server-side.

From the definition of subsystems, an architectural model was produced to be used as the basis for the subsystems. The goal was to define an architectural model that could be adopted for

all subsystems to facilitate the understanding and implementation of subsystems. The next section presents the elements of this architectural model.

4.1.1 Subsystems Logical Architecture

Each subsystem consists of three components: a *Service* to expose the functionalities of the subsystem, a *Domain* that implements the functionalities and a *Repository* that stores information related to the subsystem.

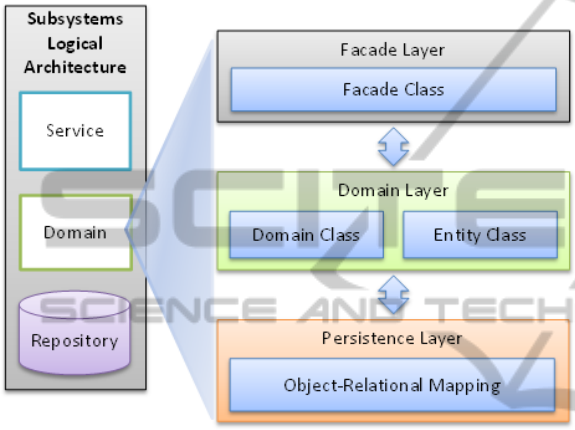

Figure 2: Domain Component Logical Architecture.

Figure 2 highlights the standard architecture defined for the subsystems *Domain* component. The subsystems' Domain component is divided into three layers of abstraction; they are:

- *Persistence* - consists of classes that are responsible for the persistence and recovery of entities of the subsystem. This layer is responsible for the implementation of the object-relational mapping (MOR) of the entities defined in the subsystem domain layer, isolating this action from the rest of the subsystem (Matic et al., 2004). The persistence layer uses the architectural pattern of data sources called Table Data Gateway (Fowler, 2003).
- *Domain* - consists of two groups of classes, domain and entity. Domain classes contain the logic of the subsystem. Entity classes represent the domain model of the subsystem, using the paradigm of object orientation. These classes represent the information and their relationships. The domain layer uses the architectural pattern known as Domain Model (Fowler, 2003).
- *Facade* - represents the subsystem interface, defined by classes that combine the public functionality of the subsystem. It aims to hide the complexity of the subsystem and to abstract class responsibilities that make up the subsystem. This layer should be implemented using the architectural pattern for domain logic named Service Layer (Fowler, 2003).

The *Repository* component consists of a Relational Database Management System (RDBMS) used for the subsystem's data persistence. Each subsystem has its own RDBMS due to the distributed architecture of the subsystems.

The *Service* component is formed by the Web Services that expose the capabilities of the subsystem. The services of each subsystem can be consumed by other subsystems of the platform or by external applications.

4.2 Application Representation

In this study we used the paradigm of object orientation associated with the paradigm of service-oriented computing to conceive a platform to build and run VR applications on a remote environment. Also proposed a model to represent VR applications. This model is capable of being totally produced and executed in a remote environment.

The representation model illustrated in Figure 3, modeled and implemented through the paradigm of object-oriented development allows instances of these objects to be remotely created, edited and persisted through service interfaces.

On the platform, a VR application consists of a virtual world (*VirtualWorldVO*), one or more scenes (*SceneVO*), which are composed of virtual objects (*VirtualObject3DVO*) and appearance characteristics (*AppearanceVO*, *MaterialVO* and *TextureVO*). Virtual worlds, scenes and virtual objects are created and maintained independent, allowing them to be reused to compose other scenes and virtual worlds.

Besides the representation of the static elements that make up the VR applications, elements of the dynamic characteristics of applications are also represented the. The behavior of the applications is represented by the elements related to the association of inputs (*InputVO*), events (*EventVO*) and behaviors (*BehaviorVO*). Events can be generated by means of data entry services (*InputEventVO*), depending on time (*TimerEventVO*) or caused by events that occur directly with objects in the virtual world (*Object3DEventVO*).

Figure 4 shows the production of applications on the platform. The aim is to discuss the potential for reuse of the key elements that can compose a VR application produced on the platform. It is possible to observe the symbolic representation of four VR

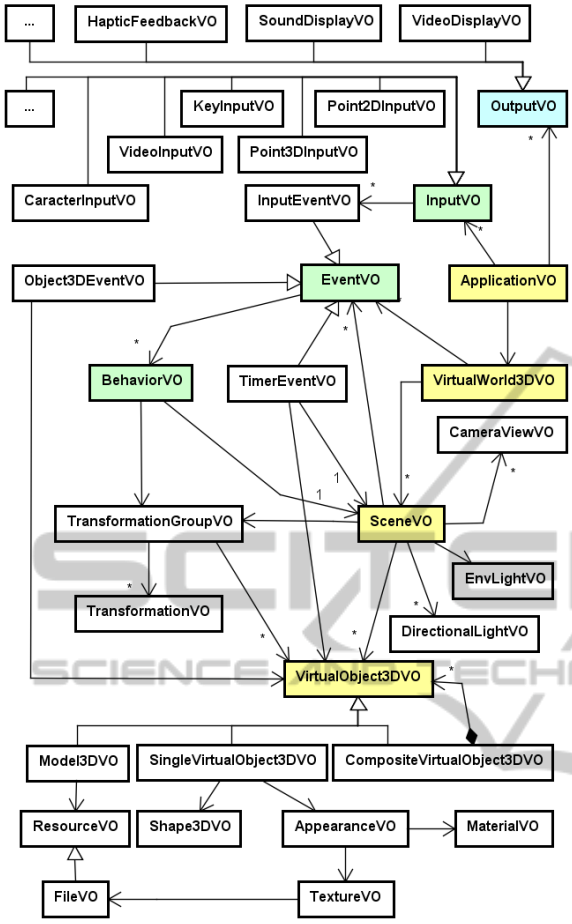

Figure 3: Application Representation Model.

applications produced by the combination of the other elements present in the repositories of the platform.

In the hierarchical representation model of the proposed application, reuse can occur at four levels:

1. the *Resource* element (3D models, images, sounds, videos) can be reused to define the virtual objects or scenes, for example, a virtual object that accesses a file with 3D representations, an image that is used as texture of a virtual object. This is illustrated in Figure 4 by the elements R3 and R4;
2. the *VirtualObject* element that can be reused to make various scenes. This is illustrated in Figure 4 by Ob2, Ob3 and Ob4 elements;
3. the *Scene* element may be reused to form different virtual worlds. This is illustrated in Figure 4 by S1, S3 and S4 elements;
4. the *VirtualWorld* member can be reused to write different applications. This is illustrated in Figure 4 by the VW2 element.

The capability to be reused is not only restricted

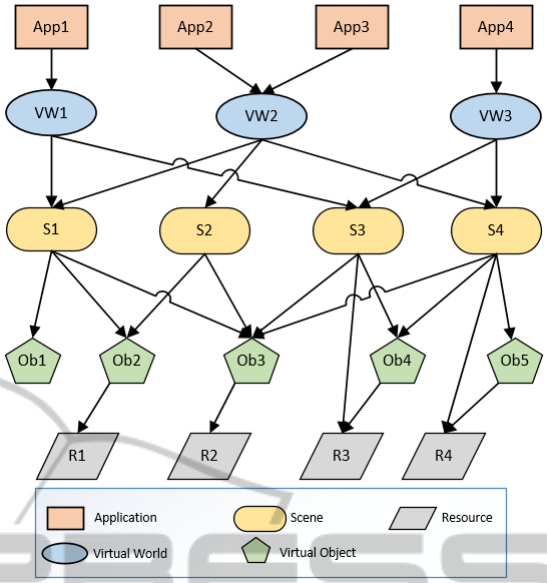

Figure 4: Levels of Reuse in Applications Production.

to the structure and appearance characteristics of the application. The behaviors associated with *VirtualObject*, *Scene* and *VirtualWorld* elements are automatically reused, as part of the definition of these elements.

4.3 Production Subsystem

In this article, the VR applications production subsystem, consisting of components such as *VirtualObjectProductionManager*, *SceneProductionManager*, *VirtualWorldProductionManager* and *ApplicationProductionManager*, has been detailed. Each component has its own service that exposes its functionality through service interface. Figure 5 illustrates the main classes of the production subsystem.

As described in the previous section, each production subsystem component has a service that exposes its functionality. Figure 5 illustrates, as an example, the application production service named *ApplicationProductionService*. The *ApplicationProductionService* service exposes the functionality of the *ApplicationProductionManager* domain class responsible for the inclusion, change, deletion and retrieval of *ApplicationVO* instances of the applications repository. The *ApplicationDAO* class is responsible for object-relational mapping of *ApplicationVO* instances.

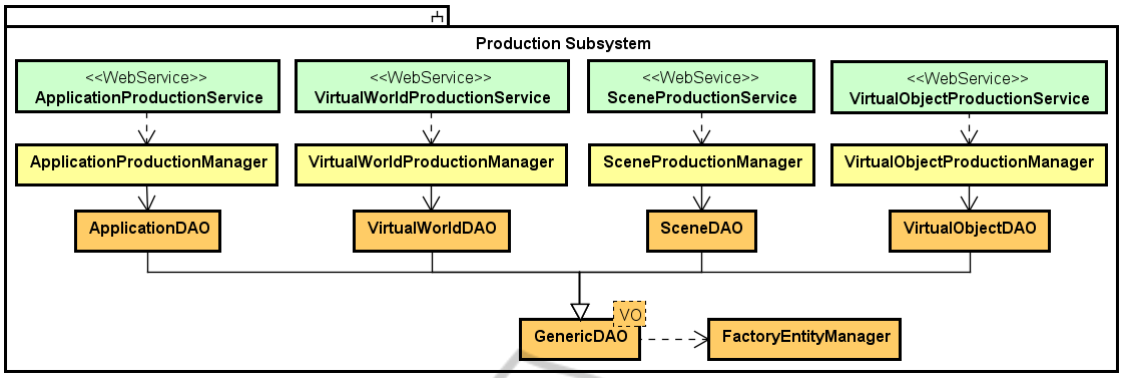

Figure 5: Production Subsystem.

5 ARCHITECTURE VALIDATION

A VR applications production environment prototype was developed, as can be seen in Figures 6, 7, 8 and 9. The development of this software prototype had two aims: 1) to evaluate the model representation, editing and reuse of elements of VR applications in the designed remote environment, and 2) to evaluate the service interface provided by the VR applications production subsystem. The production environment prototype allows to create and edit four elements that constitute a VR application as well the Application element itselfas according to the representation model illustrated in Figure 3.

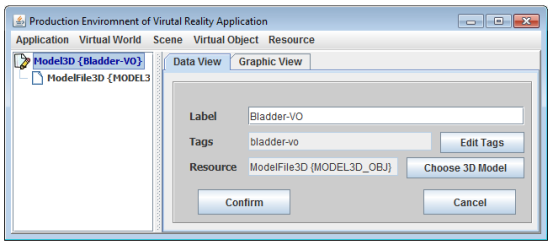

Figure 6: Prototype of a Production Client - Virtual Object.

Figure 6 illustrates the editing screen of the *VirtualObject* element. This element represents the virtual objects that can be defined into 3D models by means of reference *Resource* files, or by definition of geometric shapes or by composition of more than one virtual object. In this case study, the virtual objects were produced by reference to the *Resource* element previously inserted in the repositories. Figure 10 shows the inserted virtual objects in the case study: *Bladder-VO*, *Uterus-VO*, *LeftOvary-VO* and *RightOvary-VO*.

Figure 7 illustrates the editing screen of the *Scene* element. This element represents the scenes that are used to compose different virtual worlds. A

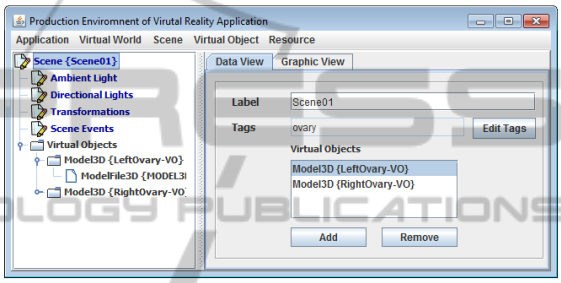

Figure 7: Prototype of a Production Client - Scene.

scene element can contain multiple *VirtualObject* elements previously inserted in the repositories of the platform. Besides the virtual objects, ambient light, directional lights, geometric transformations, events and respective behaviors are defined. Figure 10 shows scenes inserted in this case study: *Scene01*, *Scene02*, *Scene03*, *Scene04* and *Scene05*.

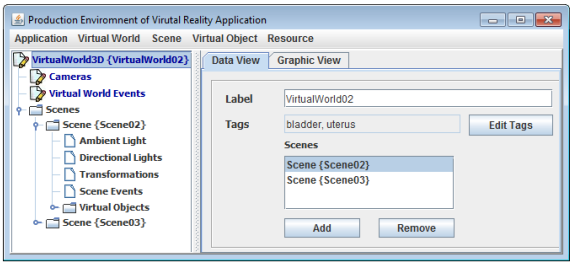

Figure 8: Prototype of a Production Client - Virtual World.

Figure 8 illustrates the editing screen of the *VirtualWorld* element. This element represents the virtual worlds that are used to compose different applications. A *VirtualWorld* element can contain multiple Scene elements previously inserted in the repositories of the platform. Figure 10 shows the inserted virtual worlds in this case study: *VirtualWorld01*, *VirtualWorld02* and *VirtualWorld03*.

Figure 9 illustrates the editing screen of the *Application* element. This element represents the

VR applications produced on the platform. An *Application* element must contain a *VirtualWorld* element previously inserted in the repositories of the platform. Figure 10 shows the inserted applications this case study: *Application01*, *Application02*, *Application03* and *Application04*.

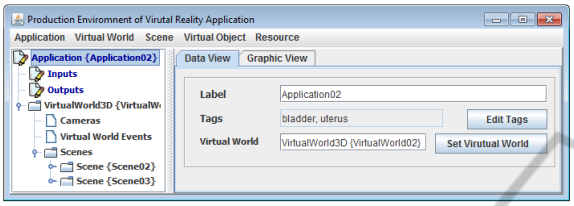

Figure 9: Prototype of a Production Client - Application.

The prototype presented consumes services provided by the platform subsystems, allowing these elements to be created, edited or deleted. Different production environments can be developed to suit different levels of users. Another important aspect is that due to the fact that the platform offers services such as Web services, different implementation technologies can be used for the development of production environments. These environments can be produced in different programming languages and execution platforms.

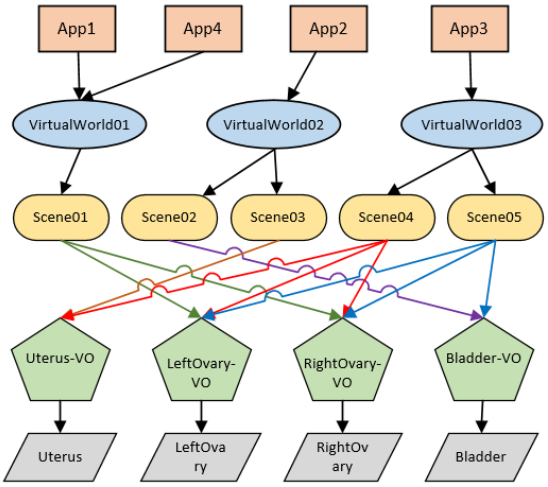

Figure 10: Case Study Elements.

Besides the application production prototype, a prototype for a VR application execution environment was developed, created through the production prototype. Figure 11 shows the three screens of the implementation prototype. The first (top) is the interface used to locate the application to be run and the second and third (middle and bottom) shows the feedback of the execution of the application on

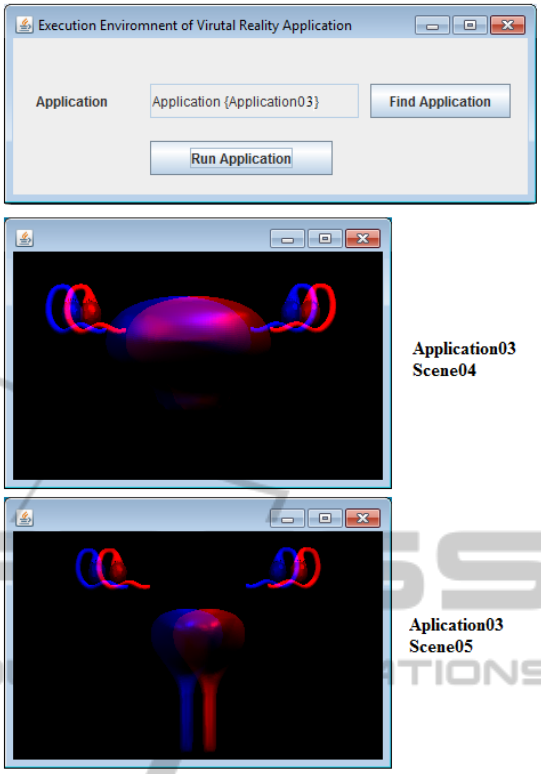

Figure 11: Prototype of a execution client.

the server side. In this first prototype, feedback is obtained by successive calls to the service which returns frames that represent the rendering that occurs on the server.

Once these critical points have been validated, the design of the platform enters a phase of development and deployment, where all the functionality of the subsystems will be implemented for deployment and delivery of the platform in a production environment. Aspects of performance and robustness of the platform will also be considered.

6 CONCLUSIONS

The proposed architectural model using the paradigm of service-oriented computing allows any application that is capable of invoking Web Services to consume the services offered by the platform. The representation model of VR applications presented allows instances of virtual objects, scenes, virtual worlds and applications to be created and edited remotely via Web Services interface.

The representation of VR applications in an online environment allows interactive content to be shared and reused to produce new VR applications.

Reducing dependence on special features such as graphic processing, high volume storage devices and specific input, extends the possibilities of platforms used as clients. With the holding of the execution of applications on the server platform, it is possible to reduce the requirements of these resources on the client-side. The use of a service-oriented model allows any computing environment (desktop, Web or mobile devices) to be used to host applications consuming the platform service.

The prototyping of the production and execution subsystems on the server-side, together with the prototyping of the production and execution environments on the client-side, made it possible to assess the viability of the proposal. Despite the limitations of the prototype, it was possible to verify the efficiency of the VR applications representation model and remote services offered by the platform, allowing the production, storage, reuse and execution of VR applications in a remote and collaborative environment.

As future work can be considered the development of production and execution environments to run in an Web platform, definition of a iconographic representation of VR application elements, should also be treated the performance and robustness aspects of the platform, expanding the possibilities of applications representations that require response low-latency.

ACKNOWLEDGEMENTS

The authors thank CAPES, the Brazilian government entity dedicated to the training of human resources and FAPEMAT, Foundation for Research Support of the State of Mato Grosso, for providing support towards the viability of the EPUSP/UFMT/IFMT agreement for the completion of the PhD on which this work is based.

REFERENCES

Bein, D., Bein, W., and Madiraju, P. (2009). The impact of cloud computing on web 2.0.

Chevaillier, P., Trinh, T.-H., Barange, M., De Loor, P., Devillers, F., Soler, J., and Querrec, R. (2012). Semantic modeling of virtual environments using MASCARET. In *2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 1–8.

Erl, T. (2009). *Soa Princípios de Design de Serviços*. PRENTICE HALL BRASIL.

Filho, R. F. d. A., Teichrieb, V., and Kelner, J. (2011). Hydra: Virtual environments development platform. In *2011 XIII Symposium on Virtual Reality (SVR)*, pages 102–111.

Fowler, M. (2003). *Patterns of enterprise application architecture*. Addison-Wesley, Boston, MA.

Matic, D., Butorac, D., and Kegalj, H. (2004). Data access architecture in object oriented applications using design patterns. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 2, pages 595–598 Vol.2.

Papazoglou, M. P. and Heuvel, W. J. V. D. (2007). Service oriented architectures: Approaches, technologies and research issues. *VLDB Journal*, 16(3):389–415.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255.

Shao, G. and McGraw, R. (2009). Service-oriented simulations for enhancing situation awareness. In *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, page 48:1–48:7, San Diego, California. Society for Computer Simulation International. ACM ID: 1639859.

Tori, R. and Kirner, C. (2006). Fundamentos de realidade virtual. In Tori, R., Kirner, C., and Siscoutto, R., editors, *Fundamentos e Tecnologia de Realidade Virtual e Aumentada*, pages 02–21. Editora SBC – Sociedade Brasileira de Computação, Porto Alegre, VIII Symposium on Virtual Reality Belém – PA.

Wang, A. J. A. and Qian, K. (2005). *Component-Oriented Programming*. Wiley-Interscience, 1 edition.

Zhang, X. and Gracanin, D. (2008). Service-oriented-architecture based framework for multi-user virtual environments. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, page 1139–1147, Miami, Florida. Winter Simulation Conference.