Service-Oriented Platform for Reuse of Interactive Content of Virtual Reality Applications

Evandro C. Freiberger, Ricardo Nakamura, and Romero Tori

***Abstract*—Virtual reality is used in the production of complex virtual environments involving input and output nontrivial devices to allow users a sense of immersion in real time synthetic worlds. Many approaches to software development are used to promote increased productivity and reuse of elements of virtual reality applications, such as routine libraries, frameworks and component platforms. However, the most approaches provide reuse by means of source code of programs or reuse of compiled components. Aiming to increase the reuse potential of elements of virtual reality applications, this paper proposes a representation model of applications and an architectural model for a remote online software platform. The service-oriented platform supports production, reuse, sharing and execution of virtual reality applications in a remote environment and online. The goal is to produce, share and reuse elements of virtual reality applications on a higher level than that of the source code of computer programs. The results obtained with architectural model validation and the developed prototypes indicate the viability of the representation model of virtual reality applications proposed.**

***Index Terms*—Interactive content, reuse, service-oriented computing, virtual reality.**

I. Introduction

Many research projects are being developed to promote gains in the production of virtual and augmented reality applications [1]-[3] and [4]. Several approaches with different levels of granularity in their components have been proposed to enable an increase in productivity and quality in the development of Virtual Reality (VR) applications. These include Application Programming Interfaces (API) which generalize specific issues, frameworks that generalize a family of applications for one problem domain and component platforms that can be reused to build different applications.

VR systems have as their predominant characteristic the demand for computational resources and unconventional devices, such as special input devices, special viewing equipment, besides the high power graphics processing devices. In some cases, users are penalized for carrying bulky uncomfortable equipment. The use of lightweight, portable equipment can facilitate interaction with such systems. Equipment such as laptops, tablets and smartphones, besides being lightweight, typically offer a set of interaction features such as keyboard, camera and compass sensors. New computing and developmental models are used to provide support for technological advances, such as service-oriented computing and cloud computing. On the other hand these new computational models cause the emergence of new challenges [5].

Aiming to increase the potential for reuse and interactive content sharing, this paper proposes a platform for service-oriented software in order to enable the production, execution and interactive content sharing of VR applications in distributed online environment. In addition to the platform, we propose a model of representation of virtual worlds composed of elements that can be recombined to produce other virtual worlds. In this new model of representation, storage and execution of VR and AR applications, clients are service consumer applications that send data originated in input devices present on the client-side. They update the respective virtual devices in the context of the application execution on the server-side, and they also receive audio streams, video and actuator update such as feedback on the application execution.

The paper is organized into eight sections, besides this introduction. Section II shows the characteristics of Service-Oriented Computing. Section III shows the characteristics of Web 2.0. Descriptions of papers with higher affinity to this research are presented in Section IV. Section V describes the proposed RV application representation model and proposed platform architectural model. Section VI describes the production process of VR applications. Section VII displays script review and validation of the proposed architecture. Section VIII presents considerations about the proposed architectural model. Finally, Section IX presents the considerations and conclusions about the proposed architectural model of the paper.

II. 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 [6].

Currently, the technical solution mostly adopted for the development of services-oriented computing is Web Services [7] and [8]. Web Services can be understood as online self-describing distributed components that expose services and functionalities through online interfaces, and can be published, located, and invoked by means of Internet communication protocols and programming [8].

Manuscript received October 20, 2013; revised December 23, 2013.

Evandro C. Freiberger is with Instituto Federal de Mato Grosso (IFMT) and Universidade de São Paulo (USP), Brazil (e-mail: evandro.freiberger@cba.ifmt.edu.br).

Ricardo Nakamura and Romero Tori are with Universidade de São Paulo (USP), Brazil (e-mail: ricardo.nakamura@poli.usp.br, romero.tori@poli.usp.br).

DOI: 10.7763/IJEEEE.2014.V4.335

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 [9], 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 and simplifies management of the distributed system.

III. Web 2.0

The Web 2.0 is an important advent, mainly characterized by the possibility of the content to be produced collaboratively, empowering and enabling the concept of collective intelligence. A result example is the popularization of virtual communities, allowing people around the world to exchange information and experiences.

The term Web 2.0 was introduced in 2004 by Tim O'Reilly to describe a growing type of web-based applications, such as wikis, tagging of media and sharing, blogs and social networks that provide the collective intelligence of users due to the content generated through collaboration and sharing. In this line, the Web is seen merely as a platform for creating systems that are connected by a set of protocols, open standards and cooperation agreements between different application platforms [10].

From the technology perspective, many Web 2.0 applications are supported by a series of web-based services with low coupling and a combination of various elements of rich media. As a practical implication of a technological approach, dynamic creation and sharing information in Web 2.0 applications maximize the collective intelligence of the user community, providing benefits for each individual user [11].

In this paper, the concept of Web 2.0 is related to the requirement that the platform proposed should enable the development of interactive content of VR applications collaboratively. The applications are produced by reusing elements previously produced by participant users of the platform. The recombination of these elements increases the potential of reuse and sharing of iterative content used in the production of new applications.

IV. 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 and AR applications, with emphasis on the reuse of software, in particular using the services-oriented computing approach.

Bauer *et al.* [12] present a framework that is based on distributed components for AR systems named Distributed Wearable Augmented Reality Framework (DWARF). In addition to the framework, they include a middleware that combines the services of components and establishes the communication between them. The applications can be deployed on mobile computers that the user carries with him. The system is capable of dynamically integrating the services offered by distributed computing in the environment and extend their own capabilities.

Oliveira and Nunes [13] describe the Virtual Medical Training (ViMeT), an object-oriented framework for the production of applications to simulate biopsy examinations. To facilitate the instantiation of the framework, a tool called ViMeT Wizard was produced. The tool allows the definition of particular parameters for each application to be produced, from which the tool generates custom code for the instantiation of the framework, reducing the learning efforts of the framework and reducing the encoding for the applications production.

Zang and Gracanin [14], 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 [15] proposed a framework named Service-Oriented Embedded-Simulation Software (SOESS), which combines SOA and Cloud 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.

Filho, Teichrieb and Kelner [16] 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.* [4] 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 framework Multiagent System for Collaborative Adaptive and Realistic Environments for Training (MASCARET) is a generic framework that provides the necessary abstractions for modeling semantics-oriented Virtual Environment content.

Realinho, Romão and Dias [17] propose a framework named Integrated Virtual Operator (IVO), designed to support the development of context-aware mobile applications, using smartphones as ubiquitous interaction devices. The framework is based on events processed by a state machine. It is designed to allow non-programmer end-users to create and run applications for mobile devices. Two construction tools (IVO Builder and IVO Outlook) are

used to enable end-users to use a visual programming environment to build and deploy context-aware applications without the need to write out any programming code. The IVO client runs the applications on the smartphone platform, through a workflow engine.

The gap observed and taken as objective of this work is the possibility to produce and execute VR applications in a fully remote environment. The work also aims to allow applications to be produced by recombining existing elements in the repositories of the remote environment.

V. Platform Architecture

The proposal platform is characterized as a service bus that provides the capacity to produce and execution VR applications, in a distributed online environment. The platform must meet two goals: the first one is to allow the production and execution of VR applications in the online environment and the second one is to provide the reuse of elements that constitute the VR applications.

To achieve the first goal, we used the paradigm of service-oriented computing, allowing all of the platform features to be distributed and exposed through services. The second goal was met using a conceptual representation model of virtual worlds (Fig. 2), produced by means of the representation of object-oriented concepts that constitute a VR application.

The platform is divided in four macro functions. The use case diagram, illustrated in Fig. 1, presents the view of the context of the platform, highlighting four functions:

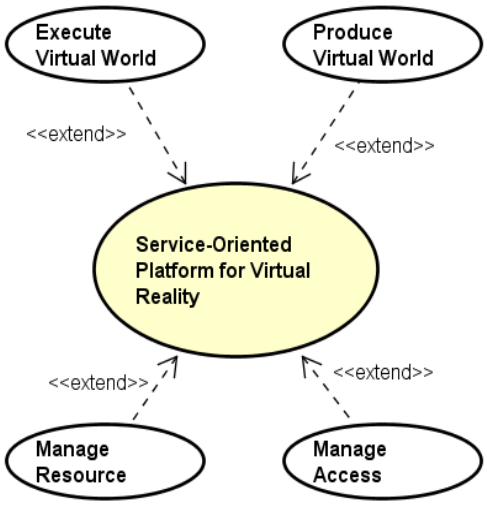

Fig. 1. Functional context diagram.

Produce Virtual Worlds - brings together functions that allow the production of VR applications constituted by a virtual world, virtual objects, scenes, input and output devices.

Execute Virtual World - brings together functions that allow the execution of VR applications produced on the platform. Must manage the input virtual devices, output streams, resources used in the execution and the events that occur during execution.

Manage Access - brings together functions related to the maintenance of users, user profiles and credentials that establish what features each user profile will be allowed to perform in the platform.

Manage Resource - gathers functions related to the maintenance of resources involved in the production and execution of applications. Examples of resources: images, sounds, textures, 3D object models.

A. Application Representation

This model is capable of being totally produced and executed in a remote environment.

This model allows the representation of the elements of a VR application. The applications are produced by means of a hierarchy of objects that represent the static and dynamic elements of the VR applications. The representation model of VR applications allows editing and execution to be performed remotely through calls to the platform.

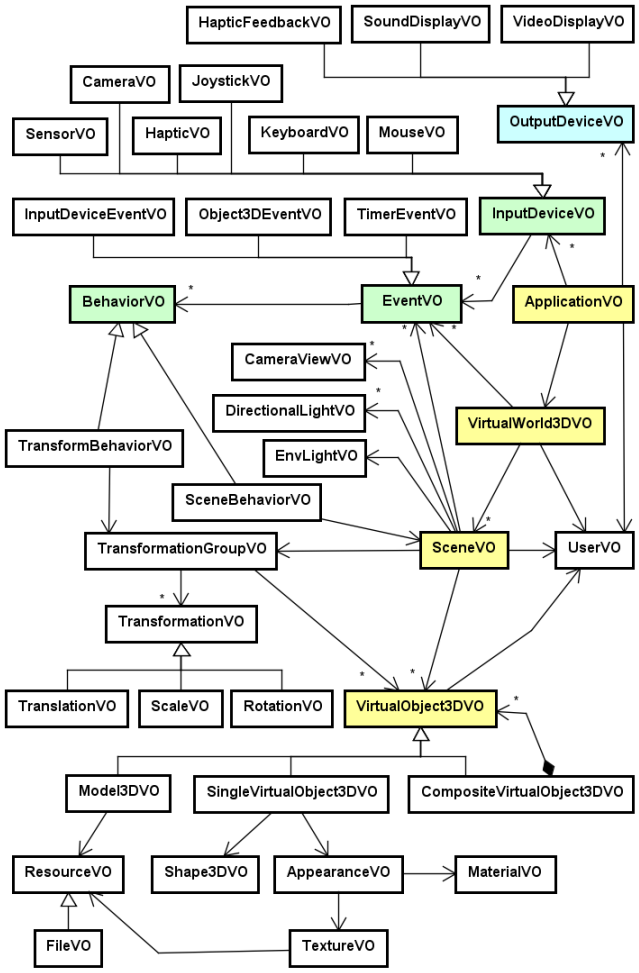

Fig. 2. Application representation model.

On the platform, a VR application consists of a virtual world (*Virtual World VO*), one or more scenes (*Scene VO*), which are composed of virtual objects (*VirtualObject3DVO*) and appearance characteristics (*Appearance VO*, *Material VO* and *Texture VO*). 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 input devices (*Input Device VO*), events (*Event VO*) and behaviors (*Behavior VO*).

B. Platform Logic Architecture

The logical view of the platform elements, illustrated in the Fig. 3, consists of four subsystems and their dependency relationships.

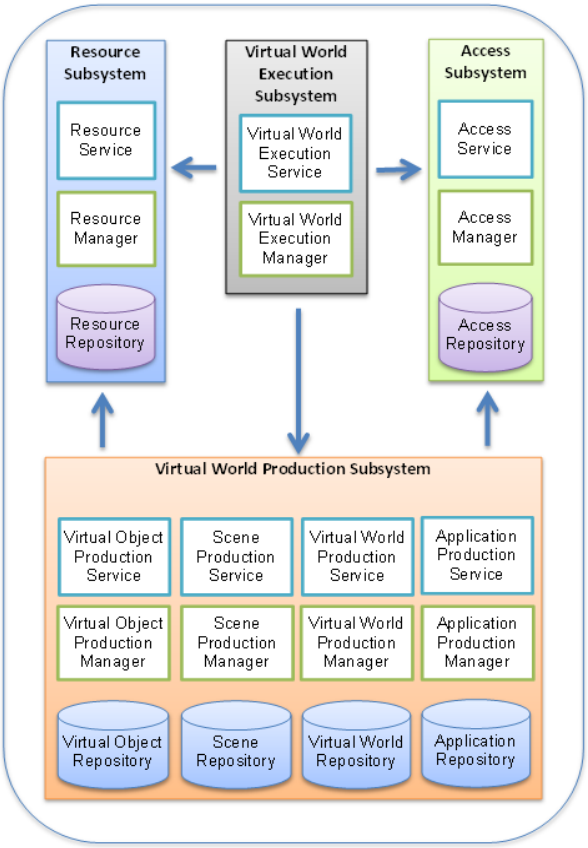

Fig. 3. Platform logical view.

The Resource subsystem, illustrated in the Fig. 3, gathers management features, searches for resources and recovery used in the production and execution of virtual worlds such as: images, sounds, videos, 3D object models and textures.

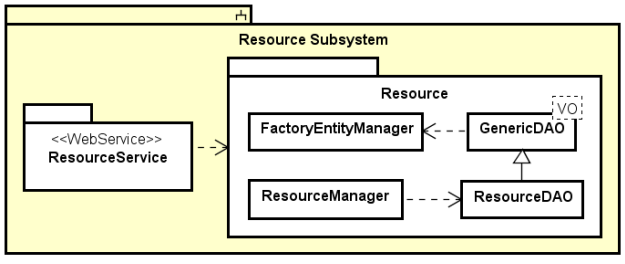

Fig. 4. Resource subsystem.

Fig. 4 shows the logical view of the architecture of the Resource subsystem. Not all software elements that make up the subsystem have been presented, but just the main logical elements of the subsystem. The subsystem consists of the *Resource* package that represents the functionality of the subsystem. This *Resource* package consists of the *Resource Manager* class that represents all the operations of domain subsystem and the *Resource DAO* class that represents all the persistence operations of the subsystem information.

The Resource subsystem is also constituted by the *Resource Service* package, representing all services used to expose the functionality of the subsystem to its customer.

The Access subsystem, illustrated in the Fig. 3, 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.

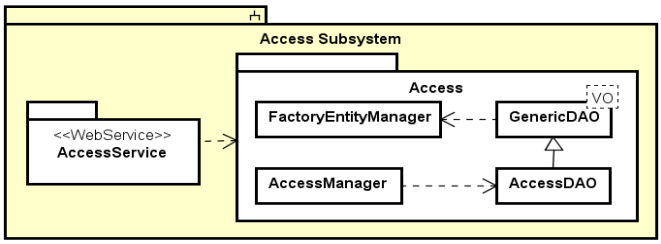

Fig. 5. Access subsystem.

Fig. 5 shows the logical view of the architecture of the access subsystem. Details of software elements that constitute the subsystem were omitted, presenting the main logical elements of the subsystem. The subsystem is formed by the *Access* package that represents the features that allow the management of users and their roles in the platform as well as the operations of authentication and authorization, represented by the *Access Manager* class.

The Access subsystem is also constituted by the *Access Service* package, representing all services used to expose the functionality of the subsystem to its customers, who are the other subsystems of the platform or client applications.

The production subsystem, illustrated in the Fig. 3, has the functionality to create, change, delete and reuse elements of VR applications. Virtual worlds, virtual objects and scenes are maintained in repositories in order to be reused in the composition of other VR applications.

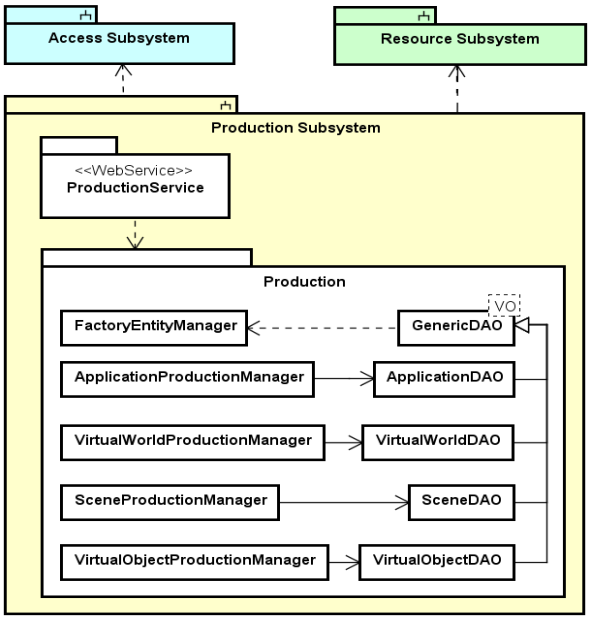

Fig. 6. Production subsystem.

Fig. 6 illustrates a subset of classes that make up the Production subsystem. Classes *Application Production Manager*, *Virtual World Production Manager*, Scene

Production Manager and *Virtual Object Production Manager* are responsible for the operations of addition, change, deletion and recovery of instances that make up the VR applications on the platform. Classes *Application DAO*, *Virtual World DAO*, *Scene DAO* and *Virtual Object DAO* are responsible for object-relational mapping of the instances that make up the VR applications.

The Production subsystem provides its functionality through service interfaces implemented with Web Services technology. It also consumes the services of the Resource and Access management subsystem.

The execution subsystem, illustrated in the Fig. 3, provides an execution context for the applications produced on the platform. The execution context consists of user context, input devices, output streams, resources and an instance of VR application. The flow control of execution is previously established through relationships between event and behaviors.

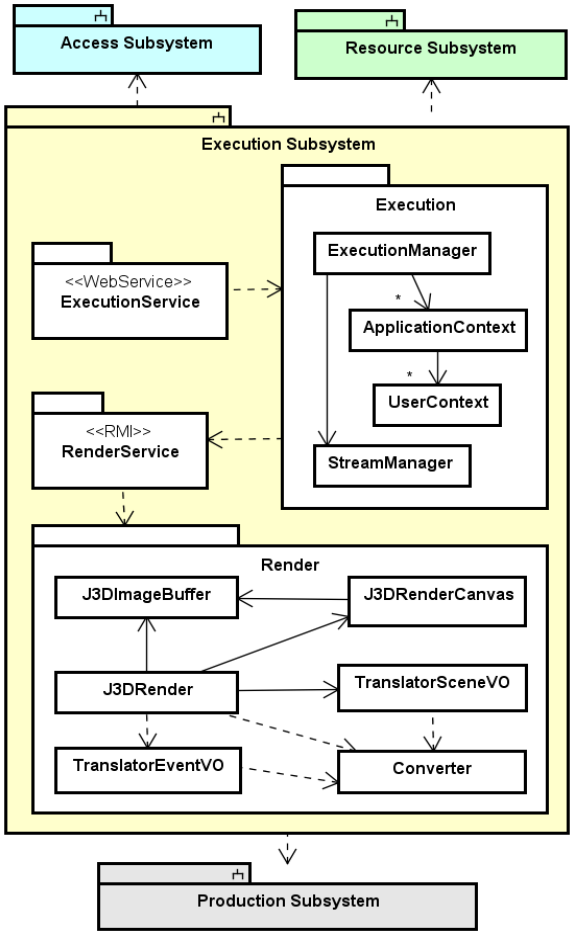

Fig. 7. Execution subsystem.

Fig. 7 illustrates a subset of classes that make up the production subsystem. The execution subsystem is formed by components *Execution Manager*, *Application Context*, *User Context* and *Stream Manager*, along with the services that publish the subsystem functions. It is the task of the *Execution Manager* component to start an application context for each instance of the running application. The *Application Context* component is responsible for receiving events from remote clients, delegating the rendering of scenes to the rendering component and providing the feedback resulting from the execution of the virtual world.

VI. Production Process of Virtual Reality Applications

The development of VR applications in the proposed platform is performed by instantiating the objects that compose the representation model of applications (Fig. 2). In this section, we present a set of possible actions to produce a VR application, considering that the application can be produced with new elements or by the reuse of existing elements. Fig. 8 illustrates the production process of applications by means of an activity diagram.

The process of application production starts with the recognition of requirements to be met. From the requirements, the first action is to seek an existing virtual world in the platform repository that meets the requirements fully or partially. If a virtual world that fully meets the requirements of the application is located, the next steps are: define the application behavior, set the execution parameters and persist the application in the repository. If a virtual world that partially meets the requirements of the application is located, the flow follows a path that will allow editing of scenes and objects that compose the virtual world. If a virtual world that meets the requirements of the application is not located, the flow follows a path that allows creating a new virtual world. The same logic sequence is used for the flows that allow editing or creating virtual objects and scenes.

The production process illustrated in Fig. 8 establishes a sequence of actions to produce applications considering the complete cycle of the elements that compose a VR application in the proposed platform. However, the production of virtual objects, scenes and virtual worlds can be carried out independently. This makes the specialization of the production task possible, for example, a specialized producer to produce virtual objects, another one specialized in producing scenes based on the virtual objects and a third one specialized in producing virtual worlds from existing scenes in the repository.

VII. Architecture Validation

The architecture validation activity was divided into two stages. The first is the architectural review which aims to carry out a verification of candidate architecture in order to identify the risks and the proposed solution. The second is the architectural evaluation, which takes into consideration the result of the first step to establish the validation techniques of the cases identified as risk points for the architecture.

We analyzed several methods of architectural review, among which Architecture Trade-off Analysis Method (ATAM), Tiny Architectural Review Approach (TARA), Software Architecture Analysis Method (SAAM) and Active Review for Intermediate Designs (ARID) stand out [18]-[21].

ATAM and SAAM are methods with strong stakeholder participation and are aimed at evaluating completed architectures. TARA is a simplification of the ATAM while maintaining most of its features, but it diminishes the participation of stakeholders [19]. ARID was chosen for this phase of architectural development since it can be applied to incomplete architectures, combining with the characteristics of research projects that require multiple revisions [21].

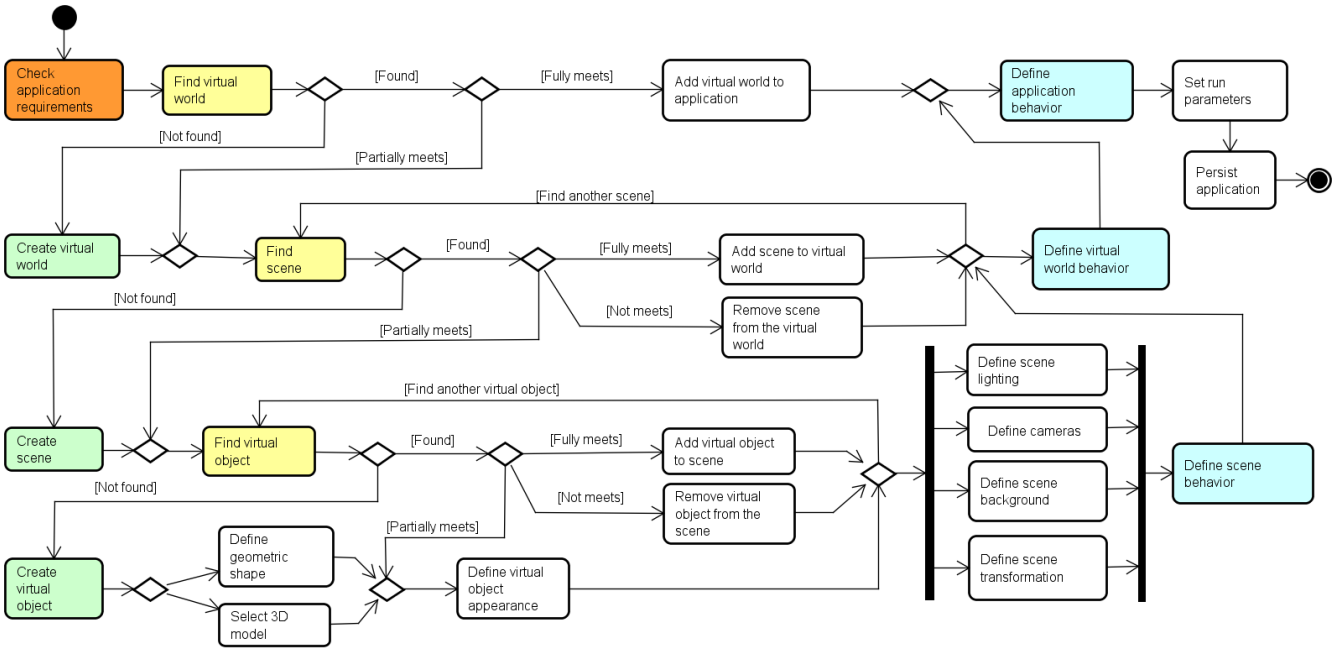

Fig. 8. Production process of VR applications.

The execution of the ARID method has detected four critical points in the architecture: conceptual representation of VR applications, VR applications production in remote environment, running control in remote environment and virtual worlds rendering in remote environment.

The Web Services technology was chosen to implement the platform services proposed in this paper, on the basis of considerations presented by the authors Erl [7], Papazoglou *et al.* [8] and, Wang and Qian [9]. According to Zhang and Gracanin [22], the use of streaming is the way to overcome the performance limitations of Web Service technology, especially in real-time applications with the need for low latency. Thus, the flow of video and sound are carried out by streaming periods triggered by a service call.

The validation of the critical points of the architecture was performed by means of implementations produced by following a list of risks and criticality pointed out by applying the review method. The implementations and tests were conducted using the following implementation and execution technologies: Java development kit (JDK 7), Java Runtime (JRE 7), Netbeans IDE 7.2.1, 1.5.1 Java 3D API, API object-relational mapping with EclipseLink JPA 2.0 engine, JAX-WS for the definition, implementation and generation of Web Service, PostgreSQL 9 database, Glassfish 3.1.2 application server.

A VR applications production environment prototype was developed, as can be seen in Fig. 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 itself as according to the representation model illustrated in Fig. 2. In order of increasing granularity the elements are: Resource, Virtual Object, Scene, Virtual World and Application. Each element of lower granularity can be used to form higher granularity elements, thus allowing reuse. For example, an instance of *Virtual Object VO* can be used to compose several instances of *Scene VO*. *Scene VO* can be reused in different *Virtual World VO*.

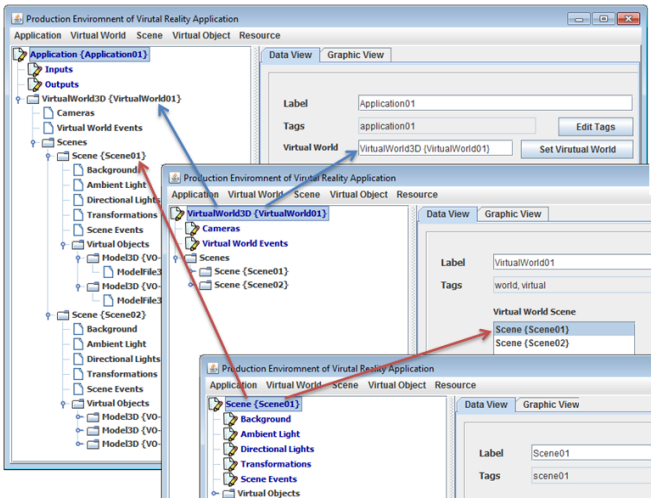

Fig. 9. Prototype of a production client.

Besides the application production prototype, a prototype for a VR application execution environment was developed, created through the production prototype. This prototype had two objectives: 1) to evaluate the capability of the proposed VR applications, especially the ability to represent behavior and 2) to evaluate the application execution services provided by the execution subsystem. Fig. 10 shows the two screens of the implementation prototype. The first (top) is the interface used to locate the application to be run and the second (bottom) shows the feedback of the execution of the application on 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. In upcoming versions streams will be used both for sending data from input devices, as well as for obtaining

feedback from the rendering performed on the server side.

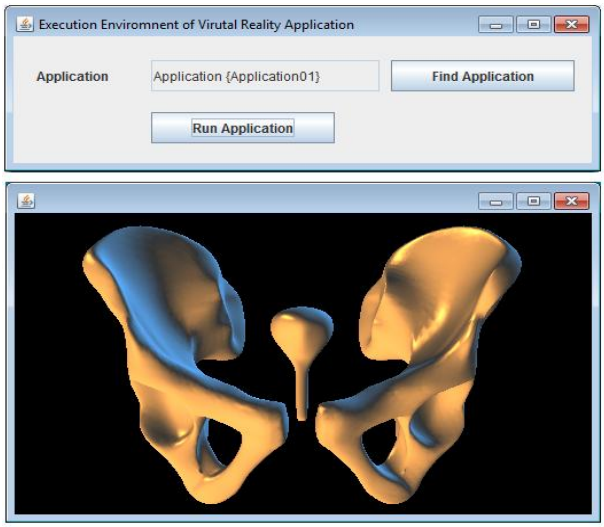

Fig. 10. Prototype of an execution client.

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.

VIII. Results Discussion

The results obtained so far show favorable aspects of the proposed platform. The first results obtained with the platform were on the ability of representing RV applications through instances and relations of objects, as has been partially laid out in Fig. 2.

The production subsystem, associated with the characteristics of representation of VR application elements, enables the reuse of application elements at four levels: resource, virtual world, scene and virtual object. The production and editing of resources, virtual objects and scenes is carried out independently, allowing the elements to be reused in virtual worlds and scenarios not provided at the time of their creation.

The remote environment of production and execution of the platform is accessible through services available through Web Services. This feature enables any client application to embed a code to consume the platform service. It also allows many production and execution environments to be produced, each of them with features and tools aimed at a desired application domain. The graphical environments of production and implementation of VR applications can be manipulated by users who have no technical knowledge of software development, especially in the development of VR applications.

The current state of research aims to demonstrate the feasibility of production and execution of VR applications in a distributed environment through the conceptual representation of the elements that compose applications. These findings emphasize the need to expand the studies to ensure that issues related to performance and robustness are optimized. It is also necessary to extend the capacity of behavior representation and specification for applications produced on the platform.

IX. Conclusion

The proposed architectural model using the paradigm of service-oriented computing, allows any application able to invoke Web Services to consume the services offered by the platform. This model also allows the platform components extension and modification in a transparent manner, with low coupling.

Because of the application representation model and the remote storage of VR applications, the production of new applications can take place through the composition of existing elements in four levels: resource, virtual world, scene and virtual object. Virtual worlds, scenes and virtual objects stored in the repository can be reused to compose different applications. The remote environment also promotes the sharing of resources, such as: images, sounds and videos.

By keeping the storage and execution of applications on the server-side, the platform client does not need to provide a high potential for processing and storage. The virtualization of input devices allows customers to update their states through service calls, avoiding the need for the existence of specific devices on the client-side. It also allows devices not present on the client-side to be simulated by software or replaced by similar devices, only requiring that the data be passed to the execution context through service calls.

The development of production and execution environments for RV applications may be considered in future studies. Those environments could offer user interfaces with a higher level of abstraction, enabling users with less technical knowledge to produce and share VR applications.

Acknowledgment

The authors thank Coordination for Higher Education Staff Development (CAPES), Foundation for Research Support of the State of Mato Grosso (FAPEMAT) and Federal Institute of Mato Grosso (IFMT).

References

[1] T. Guo, "Virtual-environment-based instrument development technology," in *Proc. 16th International Conference on Artificial Reality and Telexistence–Workshops, 2006, ICAT '06*, 2006, pp. 232–235.

[2] C ́. Fleury and A. Chauffaut, "A generic model for embedding users' physical workspaces into multi-scale collaborative virtual environments," presented at Apresentado em 20th International Conference on Artificial Reality and Telexistence, Adelaide, Australia, 2010.

[3] H. Liu, M. Bowman, W. A. Hunt, and A. M. Duffy, "Enabling behavior reuse in development of virtual environment applications," in *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 2012, pp. 1–12.

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

[5] C. Taylor and J. Pasquale, "Towards a proximal resource-based architecture to support augmented reality applications," Virtual Reality

Workshop (CMCVR), 2010 Cloud-Mobile Convergence for, 2010, pp. 5–9.
[6] M. P. Papazoglou and W. J. V. D. Heuvel, "Service oriented architectures: Approaches, technologies and research issues," *VLDB J.*, vol. 16, no. 3, pp. 389–415, 2007.
[7] T. Erl, *Soa Princ ́ipios de Design de Servi ̧cos*, Brasil: Prentice Hall, 2009.
[8] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: A research roadmap," *Int. J. Coop. Inf. Syst.*, vol. 17, no. 2, pp. 223–255, 2008.
[9] A. J. A. Wang and K. Qian, *Component-Oriented Programming*, ed. Wiley-Interscience, 2005.
[10] T. O'Reilly, *What is web 2.0?*, O'Reilly, 2005.
[11] S. Kurkovsky, D. Strimple, E. Nuzzi, and K. Verdecchia, "Convergence of web 2.0 and SOA: taking advantage of web services to implement a multimodal social networking system," in *Proc. 11th IEEE International Conference on Computational Science and Engineering Workshops*, 2008, pp. 227 –232.
[12] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Rill, C. Sandor, and E. M. Wagner, "Design of a component-based augmented reality framework," pp. 45–54, 2001.
[13] A. C. M. T. G. Oliveira, L. Pavarini, F. L. S. Nunes, L. C. Botega, D. J. Rossatto, and A. Bezerra, "Virtual reality framework for medical training: implementation of a deformation class using Java," in *Proc. the 2006 ACM international conference on Virtual reality continuum and its applications*, New York, NY, USA, 2006, pp. 347–351.
[14] X. Zhang and D. Gracanin, "Service-oriented-architecture based framework for multi-user virtual environments," in *Proc. the 40th Conference on Winter Simulation*, Miami, Florida, 2008, pp. 1139–1147.
[15] G. Shao and R. McGraw, "Service-oriented simulations for enhancing situation awareness," in *Proc. the 2009 Spring Simulation Multiconference*, San Diego, California, 2009, pp. 48:1–48:7.
[16] R. F. dos A. Filho, V. Teichrieb, and J. Kelner, "Hydra: virtual environments development platform," *2011 XIII Symposium on Virtual Reality (SVR)*, 2011, pp. 102–111.
[17] V. Realinho, T. Rom ão, and A. E. Dias, "An event-driven workflow framework to develop context-aware mobile applications," in *Proc. the 11th International Conference on Mobile and Ubiquitous Multimedia*, New York, NY, USA, 2012, pp. 22:1–22:10.
[18] M. Mattsson, H. Grahn, and F. Mårtensson, "Software architecture evaluation methods for performance, maintainability, testability, and portability," 2006.
[19] E. Woods, "Industrial architectural assessment using TARA," in *Proc. 2011 9th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2011, pp. 56–65.
[20] M. A. Babar, L. Zhu, and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," in *Proc. Software Engineering Conference, 2004 Australian*, 2004, pp. 309–318.
[21] P. C. Clements, "Active reviews for intermediate designs," Norma T écnica CMU/SEI-2000-TN-009, 2000.
[22] X. Zhang and D. Gracanin, "Streaming web services for 3D portal applications," in *Proc. the 13th international symposium on 3D web technology*, New York, NY, USA, 2008, pp. 23–26.

Evandro Freiberger was born in Mato Grosso do Sul, Brazil, in 1972. He graduated in Computer Science (Universidade Federal de Mato Grosso, 1997). He has a Master degree in Computer Science (Universidade Federal de Santa Catarina, 2003), and is currently a Ph.D. candidate in computer engineering at the Universidade de S ão Paulo, Brazil. His research interests include augmented reality, virtual reality, service-oriented computing and software reuse. He is currently is a professor at the Instituto Federal de Mato Grosso, Brazil. Prof. Freiberger is a member of the Brazilian Computer Society (SBC).

Ricardo Nakamura was born in S ão Paulo, Brazil, in 1976. He graduated in mechatronic engineering (Universidade de S ão Paulo, 1998), obtained a master's degree in electrical engineering (Universidade de S ão Paulo, 2002) and a doctorate in electrical engineering (Universidade de S ão Paulo, 2008). He has worked as a Software Engineer and Lecturer and currently is an Assistant Professor at Universidade de S ão Paulo. Currently, his research interests include augmented reality, human-computer interaction and digital games. Prof. Nakamura is a member of the Association for Computing Machinery (ACM) and the Brazilian Computer Society (SBC).

Romero Tori was born in S ão Paulo, Brazil, in 1959. He graduated in electronic engineering (Universidade de S ão Paulo, 1982), obtained a master's degree in electrical engineering (Universidade de S ão Paulo, 1988) and a doctorate in electrical engineering (Universidade de S ão Paulo, 1994). He has worked as a Computer Engineer and Advisor and currently is an Associate Professor at Universidade de S ão Paulo and a Full Professor at Centro Universit ário Senac. His research interests include augmented reality, human-computer interaction and computer in education. Dr. Tori is a member of the Association for Computing Machinery (ACM) and the Brazilian Computer Society (SBC).