Pós-Graduação em Ciência da Computação

RODRIGO DA CRUZ FUJIOKA

UMA ARQUITETURA DE REFERÊNCIA PARA GERENCIAMENTO E REUSO DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2015

Rodrigo da Cruz Fujioka

Uma Arquitetura de Referência para Gerenciamento e Reuso de Ambientes Virtuais Tridimensionais.

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR

ORIENTADOR(A): **Dr. Fernando da Fonseca**

RECIFE
2015

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

F961 Fujioka, Rodrigo da Cruz
Uma arquitetura de referência para gerenciamento e reuso de ambientes virtuais tridmensionais / Rodrigo da Cruz Fujioka . – 2015.
162 f.: il., fig.

Orientador: Fernando da Fonseca de Souza.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2015.
Inclui referências e apêndices.

1. Engenharia de software. 2. Realidade virtual. I. Souza, Fernando da Fonseca (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2017-89

Tese de Doutorado apresentada por **Rodrigo da Cruz Fujioka** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Uma Arquitetura de Referência para Gerenciamento e Reuso de Ambientes Virtuais Tridimensionais"**, orientada pelo **Prof. Fernando da Fonseca de Souza**e co-orientada pelo **Prof. Marcus Salerno de Aquin**o aprovada pela Banca Examinadora formada pelos professores:

_____

Profa. Veronica Teichrieb
Centro de Informática / UFPE

_____

Profa. João Marcelo Xavier Natario Teixeira
Departamento de Estatística e Informática / UFRPE

_____

Prof. Marckson Roberto Ferreira de Sousa
Departamento de Ciência da Informação / UFPB

_____

Prof. José Josemar de Oliveira Junior
Escola de Ciência e Tecnologia / UFRN

_____

Prof. Alvaro Francisco de Castro Medeiros
Departamento de Informática / UFPB

Visto e permitida a impressão.
Recife, 13 de março de 2015.

_____

Profa. Edna Natividade da Silva Barros
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

AGRADECIMENTOS

Ao professor Fernando da Fonseca de Souza, pela confiança, compreensão e pela colaboração durante a execução deste trabalho e ao professor Marcus Salerno de Aquino, pela contribuição e motivação para caminhar a jornada acadêmica, assim como acompanhamento durante todo este percurso.

Agradeço a Deus, pois ele é meu pai, meu guia e meu caminho, que, porventura, é através dele que tenho a oportunidade de estar aqui hoje, executando este projeto de Doutorado.

Aos meus pais, Hiroshi Fujioka e Inalda Silva da Cruz Fujioka, também a Rafaela, irmã e amiga, a minha tia Ivana que sempre falou palavras belas, as minhas avós Neuza e Tisuko, ao meu avô José Emanuel que, mesmo não sabendo ler e escrever, partiu deixando a todos os filhos e netos a educação e a humildade que trazemos no coração.

A minha esposa Ana Karla A. R. Fujioka e as nossas filhas Yasmin Fujioka, Yumi Fujioka e Nicole Akemi, de quem sou pai com muito orgulho. Elas que me auxiliaram após o pré-AVC ao qual fui acometido, e me compreenderam nos meus momentos de ansiedade e estresse, com carinho e muito discernimento para me dar suporte em todos os momentos.

Aos meus amigos professores e alunos da Unipê e IESP, em especial aos professores Luiz Martins, Josemary Araújo, Thatyana Dias e todos aqueles que certamente me auxiliaram no caminho que percorri.

Também aos membros da banca que, mesmo tendo um tempo reduzido, concordaram em avaliar meu trabalho, permitindo que eu apresentasse o resultado de uma árdua caminhada realizada ao longo dos últimos cinco anos.

Por fim, aos meus amigos, Eduardo Otávio, Lobão, César Rocha, Daniel Abella, Roberta, Venilton, Lurdeneide, Gracilene e José Carlos.

E a todos aqueles que colaboraram direta e indiretamente para a conclusão deste trabalho.

“A mente de um homem expandida por uma nova ideia não consegue nunca voltar às suas dimensões originais”.

Oliver Wendell Holmes (1809 -1894).

RESUMO

Os Ambientes Virtuais Tridimensionais (3D) vêm se tornando uma alternativa para o desenvolvimento de aplicações mais realistas e intuitivas para os usuários, pois permitem a navegação e interação com elementos próximos das suas realidades. Técnicas para geração de Ambientes Virtuais que se adaptam ao perfil do usuário têm sido desenvolvidas, incorporando procedimentos de acompanhamento de ações e modificando o ambiente em função deste comportamento. Tais técnicas necessitam trabalhar constantemente com o reaproveitamento de artefatos desenvolvidos para Ambientes Virtuais Tridimensionais (AV). O presente trabalho analisa a problemática envolvida na utilização de técnicas de reuso de artefatos em AV, bem como a sua disponibilização por meio de uma Arquitetura Orientada a Serviços (SOA). Trata-se de uma abordagem multidisciplinar, envolvendo a agregação de componentes relacionados à Engenharia de Software, Reuso de Software e Realidade Virtual. Esta abordagem levou em consideração os seguintes aspectos: (i) a disponibilização e gerenciamento destes ambientes na Web; (ii) os variados tipos de artefatos existentes para descrição de mundos em 3D; e (iii) os diferentes domínios que esta arquitetura irá gerenciar. Para alcançar tais objetivos, foi definida uma arquitetura de referência baseada em SOA que gerencia um repositório de objetos virtuais tridimensionais e um framework para ser utilizado no desenvolvimento de aplicações para recuperação dos artefatos necessários à construção de ambientes virtuais. As principais contribuições elencadas nesta tese são: (i) especificação de uma arquitetura de referência baseada em SOA para distribuição e compartilhamento de ambientes virtuais; (ii) avaliação do estado da arte sobre reuso no contexto dos ambientes de Realidade Virtual; e (iii) um arcabouço de aplicações Web utilizando a tecnologia Java como prova de conceito da solução do problema investigado.

Palavras-chave: Realidade Virtual. Engenharia de Software. Reuso de Software. SOA. *Web Services*.

ABSTRACT

Virtual Three-dimensional (3D) environments are becoming an alternative to the development of more realistic and intuitive applications for users because they allow browsing and interacting with elements close to their realities. Techniques for generating virtual environments that fit user profile have been developed, including tasks of actions follow-up and modifying the environment due to this behavior. Such techniques need to work constantly with the reuse of artifacts developed for AV. This work analyzes the issues involved in the use of AV in recycling techniques, as well as, their provision through a service-oriented architecture (SOA). It is a multidisciplinary approach that includes components related to Software Engineering, Software Reuse and Virtual Reality. This approach takes into account the following: (i) the availability and management of those environments on the Web; (ii) several kinds of existing artifacts descriptions of 3D environments; and (iii) the different areas that architecture will manage. To achieve those goals, a SOA-based reference architecture was defined. It manages a repository of three-dimensional virtual objects and a framework to be used in the development of applications for recovery of the artifacts needed to build virtual environments. The main contributions listed in this doctoral thesis are: (i) specification of a SOA-based reference architecture for distribution and sharing of virtual environments; (ii) evaluation of the state of the art for reuse in the context of a virtual reality environment; and (iii) a web application framework using Java technology as proof of concept of the proposed solution.

Keywords: Virtual Reality. Software Engineering. Software reuse. SOA. Web Services.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AOP	Aspect Oriented Programming
AV	Ambientes virtuais tridimensionais
AVI	Ambientes virtuais inteligentes
BLOB	Binary Large Objects
CDF	CAD Distillation Format
COM	Component Model
CORBA	Common Object Request Brokered Architecture
COTS	Commercial Off-The-Shelf Systems
CSS	Cascading Style Sheets
DAO	Data Access Object
DBC	Desenvolvimento Baseado em Componentes
DCOM	Distributed Component Model
DI	Dependency Injection
DOM	Document Object Model
ELPS	Engenharia de Linha de Produtos de Software
GPU	Graphics Processor Unit
HTML	HyperText Markup Language
HTTP	The Hypertext Transfer Protocol
IA	Inteligência Artificial
IAAS	Infrastructure as a Service
ISO	International Organization for Standardization
JPA	Java Persistence API
JSF	Java server faces
LPS	Linhas de produto de software
MOM	Message Oriented Middleware
MVC	Model View Controller
ORM	Object Relational Mapping
PaaS	Platform as a Service
RA	Realidade Aumentada
REST	Representational State Transfer
RM	Realidade Misturada
RMI	Remote Method Invocation

RV	Realidade Virtual
SaaS	Software as a Service
SAI	Scene Access Interface
SEI	Software Engineering Institute
SGBD	Sistema gerenciador de banco de dados
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator
UML	Unified Modeling Language
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
X3D	Extensible 3D
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de recursos de computação nas últimas décadas resultou em uma evolução significativa nas mais diversas áreas. A produção de hardware, com maior poder de processamento e baixo custo, vem tornando viável a utilização de modelos sofisticados para análise e solução de problemas complexos. O progresso tecnológico no campo da computação torna possível a obtenção de recursos gráficos, ambientes virtuais, e novas possibilidades de aplicações práticas (FERREIRA, 2013; LAGERGREN, 2002).

A Internet, um dos recursos que mais evoluíram neste período, vem atingindo um número cada vez maior de usuários, tanto no Brasil quanto no restante do mundo, em função da modernização das tecnologias da informação e de comunicação, além do aumento das conexões de banda larga (BAGLIONI; SOUZA, 2007). Além disso, pode-se considerar a interatividade, independência quanto à localização geográfica e a conectividade global como as três características que fazem da Internet um dos meios de acesso a informações mais interessantes de serem utilizados em aplicações (BARILLI et al., 2003; BARBOSA et al., 2012).

Assim, um dos pilares mais antigos da sociedade – a educação – vem tendo uma relevante influência, passando cada vez mais a utilizar os recursos que a Internet oferece para potencializar e reforçar o ensino. Estas tecnologias são conhecidas como aplicações de *e-learning*, e estão atualmente entrelaçadas com o dia a dia das pessoas, apresentando melhorias como: um acesso mais rápido e global das informações, conteúdo personalizado, distribuição e democratização do ensino, entre outras (RUIZ, 2006; SANTOS,;TOCZEK; GIMENES; 2014).

Seguindo essa tendência, diversificam-se aplicações desenvolvidas para Internet voltadas para educação e simulação, destacando-se entre elas os Ambientes Virtuais Tridimensionais (AV), que por sua vez podem ser descritos como ambientes interativos, compostos por objetos tridimensionais e gerados em tempo real por um sistema computacional. Os AV têm como objetivo a simulação de ambientes reais ou construção de ambientes imaginários, permitindo a um ou mais usuários interagirem através da visualização e manipulação de objetos ampliando o sistema sensorial (BRANDÃO et al., 2014; KIRNER; KIRNER, 2011;.KIRNER; SISCOUTTO, 2007).

Esses ambientes vêm utilizando a técnica conhecida como Realidade Virtual (RV), que é contextualizada como sendo uma avançada técnica de interface, na qual um ambiente virtual sintético tridimensional é gerado por computador, permitindo ao usuário interagir e

navegar, estando parcialmente ou completamente imerso. Um dos principais diferenciais dessa tecnologia é a sensação de presença no ambiente (CARDOSO et al.; 2014; RIVA; MANTOVANI, 2012). Tal navegação é assistida por canais multissensoriais com ou sem a utilização de ferramentas, estando em consenso entre os diversos pesquisadores que o principal canal para interação nesses ambientes é a visão (BURDEA; COIFFET, 1994; KIRNER; KIRNER, 2011).

Neste contexto, os AV que são executados de forma distribuída – fora de uma arquitetura cliente-servidor – ou seja, em qualquer local, utilizando a comunicação de dados provida pela rede mundial de computadores, tornam-se uma alternativa para o desenvolvimento de interfaces mais realistas e intuitivas para o usuário, permitindo a navegação e interação com elementos próximos das suas realidades, independente de sua localização geográfica.

Além disso, as tecnologias de RV estão evoluindo, tornando possível a construção de AV personalizados de acordo com o perfil cognitivo do usuário, incorporando procedimentos de acompanhamento das suas ações e das modificações do ambiente, bem como a simulação de eventos com o intuito de simular comportamentos específicos (AQUINO et al., 2007; CARDOSO et al., 2014; SILVA, 2014).

No entanto, apesar dessa evolução, as ferramentas que utilizam mundos virtuais têm, no processo de criação dos seus ambientes, um problema. De fato, a dificuldade encontrada em tal processo é significativa, sendo necessária a utilização de ferramentas complexas e especializadas para operá-las. Também pode ser considerado um ponto negativo o fato de os artefatos gerados não serem reutilizados em outras ferramentas, ficando destinados à plataforma para a qual foram criados.

A utilização de técnicas de reuso pode auxiliar a reduzir essas dificuldades do ponto de vista de que um componente de RV pode ser reutilizado em outra ferramenta, auxiliando na redução do custo e do tempo, aumentando a qualidade do sistema. Em geral, todas as técnicas de reuso favorecem a redução de tempo na construção de novos artefatos, bem como na melhoria de qualidade. No entanto, a reutilização em um nível mais alto de abstração é mais eficiente comparada a técnicas focadas na codificação (ALMEIDA et al., 2007; QUEIROZ, 2009).

Assim, nos últimos anos, foram concebidas diversas técnicas para apoiar a de software. Tais técnicas se embasam no fato de que sistemas correlatos construídos para um domínio semelhante possuem alto potencial de reuso, sendo possível que a reutilização seja realizada em níveis que vão desde a utilização de simples funções à composição de sistemas

completos (ALMEIDA et al., 2007; SOMMERVILLE, 2011). Dentre as técnicas existentes, Sommerville (2011) elenca como sendo as principais: Frameworks de Aplicação, Desenvolvimento Baseado em Componentes (DBC), Padrões de Projeto, Sistema Orientado a Serviços, Linhas de Produto de Software, Integração de COTS (*Commercial Off-The Shelf Systems*), Geradores de Programa, Bibliotecas de Programa e Desenvolvimento Baseado em Aspectos.

Entre as técnicas citadas anteriormente, o DBC tem destaque como uma abordagem promissora na área de Realidade Virtual (Oliveira et al., 2009; Souza, 2014), uma vez que componentes são reutilizáveis, intercambiáveis, de uso geral ou específico, podendo interagir ainda com outros componentes.

1.1 Motivação

A incorporação de técnicas de reuso para distribuição e geração de AV de forma distribuída, utilizando o conceito de computação em nuvem e arquitetura orientada a serviços (*Service Oriented Architecture* – SOA) surge como uma oportunidade de investigação, pois se trata de uma abordagem recente e com poucas pesquisas relacionadas, principalmente em relação à aplicação de técnicas de reuso (FREIBERGER et al., 2012; SOUZA, 2014).

A combinação desses elementos é proposta com o objetivo de potencializar a construção de AV para diversas plataformas e ferramentas de forma mais simples, a partir de uma base de componentes de Realidade Virtual.

Diante dos itens elencados, este trabalho tem como motivação investigar a conjunção do conceito de reuso utilizando uma arquitetura SOA aplicada ao contexto da RV na Web, uma vez que não se encontra evidências definidas na literatura sobre a combinação de tais técnicas. A exploração desses itens vai permitir investigar a viabilidade de tais abordagens na geração de AV.

1.2 Descrição do problema e justificativas

Desde algum tempo, pode-se encontrar aplicações sofisticadas, apoiadas em recursos tecnológicos avançados. A visualização da simulação, tais como os trabalhos de Balci (1997) e De Oliveira (1999), vem sendo pesquisada por mais de uma década. O desenvolvimento de tecnologias e plataformas visuais interativas conduziu ao estudo de métodos voltados para a solução dos mais variados tipos de problemas e domínios (BRANDÃO et al., 2014; CHAMOVITZ; SABBADINI; OLIVEIRA, 2008; GRIMM, P.; NAGL, F., 2010; VALKOV et al., 2011;).

No sentido de tornar os ambientes mais interessantes ao usuário, diversas técnicas para geração de ambientes virtuais têm sido desenvolvidas (AQUINO et al., 2007; CHITTARO; RANON, 2003; SANTOS; OSÓRIO, 2004, SILVA, 2014). Contudo, no processo de criação de mundos virtuais, faz-se premente a atuação de profissionais devidamente capacitados e que possuam um amplo conhecimento sobre alguma ferramenta de autoria de objetos tridimensionais.

Diante do que foi exposto, é relevante a construção de um modelo de referência para gerenciamento de Ambientes 3D que permita a reutilização de componentes virtuais, o que vai auxiliar a minimizar o viés entre o desenvolvimento e a utilização de AV em ferramentas distintas.

Desta forma, este trabalho se propõe a investigar o modelo de desenvolvimento baseado em reutilização para Web. Assim, esta investigação leva em consideração (i) a disponibilização e gerenciamento destes ambientes na Web; (ii) os variados tipos de artefato para descrição de mundos em 3D na Web; e (iii) os diferentes domínios em que o modelo dessa arquitetura poderá ser adaptado e utilizado para gerenciamento de tais objetos e mundos.

1.3 Objetivos Gerais

O principal objetivo desta tese é apresentar uma Arquitetura de Referência para gerenciamento e disponibilização de Ambientes Virtuais Tridimensionais combinando Realidade Virtual (RV), Reuso de Software e SOA. Para isso, pretende-se investigar, modelar e implementar duas aplicações, uma representando uma aplicação cliente e outra representando uma infraestrutura de gerenciamento e disponibilização baseada na Arquitetura de referência definida.

Assim, esse trabalho tem como objetivos gerais:

I. Definir uma Arquitetura de Referência que possa ser utilizada como modelo na Engenharia de Software para reuso em RV com SOA independentemente da plataforma utilizada;
II. Responder as seguintes perguntas:
    a. Como explorar o reuso de componentes virtuais em ambientes tridimensionais, utilizando o conceito de reuso mantendo a interoperabilidade entre sistemas?
    b. É viável utilizar técnicas de reutilização dentro de uma infraestrutura de reuso de componentes virtuais que funcione de forma distribuída

utilizando uma Arquitetura Orientada a Serviços?

c. É factível que a interoperabilidade entre sistemas de RV seja alcançada com disponibilização destes componentes utilizando *Web Services*?

1.3.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Contextualizar as tecnologias e conceitos importantes para o entendimento desta tese;
2. Classificar as técnicas de reuso encontradas de acordo com o protocolo (Apêndice A);
3. Realizar um estudo de caso implementando duas aplicações:
    a. Uma representando a infraestrutura de reuso de Componentes Virtuais Tridimensionais (CVT) implementada com base na Arquitetura de referência.
    b. Outro sistema representando um cliente que consome os CVT disponibilizados pelo item a.

1.4 Estrutura do documento

Este trabalho está estruturado conforme descrito a seguir. Além deste capítulo introdutório, o Capítulo 2 apresenta a primeira parte da investigação bibliográfica para embasamento conceitual desta tese; o Capítulo 3 conclui o embasamento sobre o desenvolvimento orientado ao reuso, orientação a serviços e computação em Nuvem; o Capítulo 4 apresenta o trabalho de tese descrevendo as arquiteturas, o Framework empregado no desenvolvimento, bem como o método que foi utilizado para validar o trabalho, além dos fluxos, testes e modelos definidos para implementação. Já o Capítulo 5 descreve as conclusões, resultados, limitações e contribuições obtidas. Por fim, a bibliografia referenciada no trabalho é listada, seguida dos apêndices deste trabalho.

2 AMBIENTES VIRTUAIS E TECNOLOGIAS PARA DESENVOLVIMENTO DE CONTEÚDO 3D

O escopo de investigação consiste na busca por evidências sobre a reutilização combinada com uma arquitetura baseada em SOA para disponibilização e gerenciamento de AV aplicada a sistemas de realidade virtual (RV). Nesta linha, o presente capítulo contextualiza conceitos fundamentais relacionados ao trabalho desenvolvido nesta tese de doutorado. Inicialmente, é realizada uma breve introdução sobre ambientes virtuais. Na sequência são considerados conceitos básicos sobre RV, sendo tratados os tipos de abordagem em relação ao grau de interação, bem como as principais tecnologias utilizadas na construção de tais ambientes.

2.1 Ambientes virtuais

Os AV são locais nos quais o usuário incorpora algum avatar (isto é, o usuário assume o papel de um personagem) e a partir dessa personificação passa a atuar e interagir como se aquele avatar tivesse vida, ou em outras palavras, o usuário faz de conta que é um personagem daquele ambiente (CHAMOVITZ; SABBADINI; OLIVEIRA, 2009). Também é possível que ele colabore e realize interações com outros participantes, transmitindo assim um maior sentimento de presença no ambiente (KIRNNER; SISCOUTO, 2007). As abordagens relacionadas a AV, tratadas e discutidas neste trabalho estão intrinsecamente ligadas a ambientes tridimensionais.

Em AV, o usuário torna-se uma parte do sistema, no qual ele pode ser caracterizado como uma presença autônoma no ambiente, com liberdade para navegar, interagir e explorar objetos e o próprio ambiente, seja manipulando-o, resolvendo atividades ou respondendo perguntas de variados pontos de vista (AVRADINIS et al., 2000).

Esses ambientes podem ser compostos por muitas entidades (ex. prédios, casas, veículos, pessoas, entre outros), podendo permitir a participação e interação de diversos usuários entre si e em tempo real. Tais ambientes são conectados por meio de uma rede de comunicação, permitindo, desta forma, que os usuários colaborem e interajam entre si. Se esta rede estiver conectada à Internet, a colaboração entre os membros do ambiente é possível mesmo que fisicamente os indivíduos se encontrem em locais geograficamente diferentes (PESSIN; OSÓRIO; MUSSE, 2008; SINGHAL, ZYDA, 1999).

Os AV também são conhecidos como ambientes de realidade virtual ou mundos virtuais, que podem ser descritos como uma metáfora computacional do mundo real, contendo lugares, objetos, pessoas e recursos multimídia com os quais se é possível interagir (AQUINO et al., 2005).

A utilização de técnicas de RV possibilita o desenvolvimento de AV complexos, cujos componentes são capazes de representar qualquer tipo de informação. É comum a utilização de objetos e recursos multimídia como imagens, vídeo e som, deixando-os mais próximos da realidade do usuário e, neste sentido, tornando-os mais realistas e ao mesmo tempo mais interessantes e interativos (AQUINO et al, 2007; WALCZAK; CELLARY, 2002).

Em sistemas de RV, a interatividade está relacionada às respostas do sistema em função das ações do usuário, envolvendo navegação e capacidade de manipular objetos do mundo virtual. Entretanto, para que esta interatividade seja sentida de forma mais realista é essencial que a geração de imagens seja em tempo real e direcionada às necessidades do usuário, ou que o ambiente seja dinâmico e seja alterado conforme operações do usuário com o referido ambiente.

De acordo com Aquino et al. (2007) e Silva (2014), técnicas de adaptação vêm sendo utilizadas e desenvolvidas para gerar AV personalizados, construídos de acordo com as preferências de cada usuário, seu nível de conhecimento, bem como suas necessidades.

Algumas dessas técnicas estão baseadas na ajuda ao usuário durante a navegação, bem como na definição de prioridades para apresentação de conteúdo (e.g. os produtos mais acessados ou consultados em comércio eletrônico), como por exemplo os trabalhos de Rickel et al. (2002); Chitaro e Randon (2003); e Walczak e Cellary (2002). Como visto, diversos fatores têm motivado a construção de AV. Mais recentemente, a integração de técnicas de Inteligência Artificial aos AV tem permitido a representação de ambientes que exploram o uso de entidades com certo grau de inteligência e diferentes formas de interações, provendo maior dinamicidade, realismo e usabilidade aos ambientes (BRANDÃO et al., 2014; SANTOS; OSÓRIO, 2004; SILVA, 2014).

Ambientes virtuais que empregam técnicas de Inteligência Artificial por meio do uso de entidades autônomas e inteligentes são conhecidos como Ambientes Virtuais Inteligentes (AVI). Estas entidades estão presentes no ambiente por meio de representações virtuais de formas de vida (seres humanos, animais, entre outros) e de avatares (representação do usuário no ambiente), possuindo um alto grau de interação.

Conforme Anastassakis et al. (2001) e Guo e Luqi (2000), aplicações de AVI têm sido empregadas em diversas áreas, principalmente para simulação, entretenimento e educação.

Em simulação, por exemplo, diferentes tipos de ambientes podem ser gerados para projetos arquitetônicos, manufatura ou controle de tráfego. Também tem sido explorada a simulação de humanos virtuais em situações de emergência (MUSSE, 2000), comportamentos de grupos de animais (REYNOLDS, 2001), bem como em outras áreas (BRANDÃO et al., 2014; CAVAZZA et al., 2001; LI et al., 2010; MUELLER-WITTIG et al., 2002; NIJHOLT; HULSTIJN, 2000; RICKEL et al., 2002; RIZZO et al., 2002; SECONDLIFE, 2015; SILVA, 2014).

Neste contexto, existem os sistemas adaptativos que têm a expectativa de oferecer interfaces customizadas de acordo com as necessidades e características específicas de cada usuário (AQUINO, 2007; PALAZZO, 2000; SILVA, 2014; ZORZAL, 2007). Para isso, um AV Adaptativo deve ser capaz de identificar o perfil do usuário no início de uma sessão e determinar qual o ambiente mais apropriado a ser gerado. Portanto, a representação dos mundos com os quais este ambiente vai trabalhar tem que prover meios de manipulação apor meio da inserção e remoção de objetos, além de permitir atualizações de forma eficiente (AQUINO et al., 2007).

Para Celetano (2004), Ambientes Virtuais Adaptativos devem ser dotados de inteligência a fim de atualizar o ambiente 3D e desta forma reduzir a necessidade cognitiva de interação. Neste sentido, o sistema pode auxiliar o usuário durante a interação, alterando a complexidade do ambiente ou facilitando no processo de navegação. Essas técnicas têm sido exploradas em vários trabalhos (AQUINO, 2007; AQUINO et al, 2005; CHITTARO et al, 2003; CHITTARO; RANON, 2002; ZORZAL, 2007).

Esta seção discorreu sobre algumas definições associadas a AV, bem como à utilização das mesmas para tornar os ambientes mais interessantes e instigantes a quem os utiliza, aplicando-se técnicas de adaptabilidade em função de regras programadas, seja no ambiente ou associadas à aprendizagem do usuário. A respeito da adaptabilidade, pode-se observar uma oportunidade de reuso, uma vez que a mudança de ambiente nesses AV está mais relacionada a detalhes do mundo que são exibidos ou omitidos em função das regras, configurações ou IA, programados para o AV.

Levando esses pontos em consideração é possível explorar no campo do reuso a hipótese de que o núcleo pode ser mantido alterando-se algumas das características dos componentes. Ou seja, é possível gerar ambientes virtuais personalizados ou com diferentes proporções ou características, a exemplo de uma esfera ou cubo onde se pode manter a forma estrutural e alterar características tais como cores, tamanhos e luminosidade em relação ao modelo original adaptando o artefato a outros ambientes.

2.2 Realidade Virtual

Realidade Virtual pode ser descrita como uma interface avançada do usuário para acessar aplicações dotadas de objetos tridimensionais sintéticos, gerados por computador, na qual é possível realizar algum tipo de interação. Ainda segundo Tori e Kirner (2006), nesses ambientes

> Os sentidos e as capacidades das pessoas podem ser ampliados em intensidade, no tempo e no espaço. É possível ver, ouvir, sentir, acionar e viajar muito além das capacidades humanas... Pode-se, assim, ser tão grande (em nível das galáxias) ou tão pequeno (em nível das estruturas atômicas) quanto se queira, viajando a velocidades muito superiores a da luz e aplicando forças descomunais. Ao mesmo tempo, pode-se ampliar a medida do tempo, para que as pessoas possam observar ocorrências muito rápidas em frações de segundos, implementando o conceito de câmera lenta, ou reduzir a medida do tempo, acelerando-o, para observar ocorrências e fenômenos muito lentos, que poderiam demorar séculos. Para isto, são utilizadas técnicas de modelagem tridimensional na elaboração dos objetos e mon*tag*em do cenário virtual, por onde o usuário poderá navegar. (TORI, KIRNER, 2006, p. 3).

Em RV, a forma de interação mais simples ocorre quando o usuário navega no ambiente tridimensional utilizando algum dispositivo como, por exemplo, um *mouse*. Outra definição de RV é: “realidade virtual é uma interface computacional avançada que envolve simulação em tempo real e interações, através de canais multissensoriais” (BURDEA e COIFFET apud KIRNER, KIRNER, 2011, p. 10). Ou ainda: “realidade virtual é uma interface computacional que permite ao usuário interagir em tempo real, em um espaço tridimensional gerado por computador, usando seus sentidos, através de dispositivos especiais” (KIRNER, 2011, p. 14).

Tori et al. (2006) destacam a existência de variadas definições associadas a RV, que de forma geral envolvem características de cunho tecnológico ou aspectos gerais (BURDEA, 1994; KIRNER, 1996; SHERMAN, 2003; VINCE, 1995, 2004) e descrevem outra definição para RV: “[...] é uma interface avançada para aplicações computacionais, que permite ao usuário a movimentação (navegação) e interação em tempo real, em um ambiente tridimensional, podendo fazer uso de dispositivos multissensoriais, para atuação ou feedback” (TORI et al., 2006. p. 7).

Tecnologias que utilizam a RV para realização de experimentos e simulações estão sendo favorecidas com a evolução dos computadores pessoais, os quais estão com poder de processamento cada vez mais elevado e também a redução dos custos de aquisição dos equipamentos eletrônicos. Essa tendência tem feito a RV deixar de ser apenas um objeto de

estudo dos grandes centros de investigação para se tornar uma tecnologia mais acessível para pessoas comuns (KUROSE et al., 2010; ROCHA et al.; 2013). Isto ocorre, principalmente, em ambientes de Educação a Distância (EAD) que utilizam a Internet e navegadores Web para disponibilizar o seu conteúdo (ABULRUB et al., 2011; AQUINO, 2007; BARILLI et al., 2011; MARTINS et al.,2013; SOUZA et al; 2008; TAO et al., 2010).

Sobre a experiência de imersão do usuário no ambiente, a RV pode ser separada em duas categorias, cada uma com três tipos de classificação. A combinação entre as possibilidades oferece um total de 3^2, ou seja, nove (9) opções de imersão. Esta separação é dada em função do senso de imersão do usuário no ambiente, como apresentado no Quadro 2.1 , o qual descreve os níveis, exemplificando os tipos de ambiente e a classificação destes segundo o grau de imersão.

Quadro 2.1 – Tipos de realidade virtual

	Imersivos U ←→A	Semi-Imersivos U←A	Não Imersivos U←|→A
Real	A chamada telepresença. **Ex: Recuperação de artefatos arqueológicos submarinos a alta profundidade com robôs Telecomandado.**	Reproduções navegáveis de ambientes reais de difícil ou impossível acesso. **Ex: Treino de pilotagem de um veículo.**	Reproduções tridimensionais de ambientes reais de difícil ou impossível acesso. **Ex: TAC (Tomografia Axial Computadorizada).**
Misto	A chamada Realidade Aumentada. **Ex: Operação médica robotizada.**	Modelos navegáveis de ambientes reais alterados. **Ex:Tour virtual pelo sistema solar.**	Teste virtual de elementos a ser introduzidos num ambiente real. **Ex: Visualização 3D do projeto de um edifício.**
Virtual	Realidade Virtual no sentido mais puro da palavra. **Ex: Espaço Cibernético (Cyberspace)**	Reproduções navegáveis de ambientes imaginários ou inacessíveis. **Ex: Passeio virtual na Terra Média do Senhor dos Anéis**	Reproduções tridimensionais De ambientes imaginários ou inacessíveis. **Ex: Aspecto da Terra Pré-Histórica.**

Fonte: KILNER et al. (2011, p.12).

De modo geral os pesquisadores classificam seus ambientes, em sua maioria, basicamente em dois tipos: o Imersivo e o Não Imersivo, entendendo que o Semi-Imersivo é um meio termo entre ambos. Na RV Imersiva o usuário é literalmente transportado para o ambiente da aplicação, no qual o mesmo navega no cenário utilizando dispositivos multissensoriais (capacete, luvas, cavernas e seus dispositivos) que reagem aos estímulos inferidos pelo usuário no ambiente. Já na RV Não Imersiva o usuário é projetado parcialmente

para o ambiente virtual por meio de periféricos baseados em *desktop*, *notebooks*, *tablets* e outros dispositivos 2D, nos quais a sensação de imersão geralmente é reduzida.

Este segundo tipo de imersão, embora seja mais simples em relação às sensações extrassensoriais, é aquele que traz mais oportunidades de possibilidades práticas, também sendo o mais investigado. O custo reduzido e o fato de ter sua disponibilização sendo feita no nível global por meio da Internet tornam esta opção mais atraente para os pesquisadores.

Corroborando com esta linha, Guimarães et al. (2013. P7) destacam que o baixo custo associado aos ambientes virtuais não imersivos baseados em monitores convencionais apesar de não utilizarem toda a potencialidade da RV são suficientes para diversas situações relacionados ao ensino (OLIVEIRA, MARTINS, 2010; MARTINS, OLIVEIRA, GUIAMARÃES, 2013).

Ainda conforme Aquino,

> Embora os AV imersivos tenham evoluído e apresentem aplicações muito realistas, os sistemas não-imersivos são os mais populares por serem mais baratos e mais simples de implementar. Além disso, novas tecnologias proporcionam uma visualização de melhor qualidade para os ambientes não-imersivos, aumentando assim a sensação tridimensional. (AQUINO, 2007, p. 10).

Quando se fala em RV também é importante conceituar outro termo que é comumente citado ao se referenciar AV tridimensional – a Realidade Aumentada (RA). De acordo com Azuma (1997), a RA é um complemento à RV, não sendo considerada uma substituta. O mesmo autor descreve a RA como um sistema caracterizado por três aspectos principais: combina o real com o virtual; é interativa em tempo real; os objetos virtuais são ajustados no ambiente tridimensional. Ainda segundo Tori et al,

> [...] a realidade virtual e a realidade aumentada permitem ao usuário retratar e interagir com situações imaginárias, como os cenários de ficção, envolvendo objetos reais e virtuais estáticos e em movimento. Permitem também reproduzir, com fidelidade, ambientes da vida real como a casa virtual, a universidade virtual, o banco virtual, a cidade virtual, etc., de forma que o usuário possa entrar nesses ambientes e interagir com seus recursos de forma natural, usando as mãos (com ou sem aparatos tecnológicos, como a luva) e eventualmente comandos de voz. Com isto, o usuário pode visitar salas de aula e laboratórios de universidades virtuais, interagindo com professores e colegas e realizando experimentos científicos; pode entrar no banco virtual e manusear o terminal de atendimento virtual, da mesma maneira que o faz com o equipamento real, e mesmo conversar com o gerente, representado no ambiente por um humanoide virtual. (TORI et al, 2006, p. 23).

A Figura 2.1 mostra ambientes cujos conceitos foram descritos anteriormente: a) Um Sistema de RV imersivo no qual o usuário interage com a cena utilizando dispositivos que dão

a sensação de imersão; e b) São incluídos na cena real objetos tridimensionais.

Figura 2.1 – Ambiente de Realidade Virtual e Aumentada

a- Realidade Virtual **b-** Realidade Aumentada

Fonte: a – Adaptado de Info (2009) e b – Rodello e Brega (2011).

A utilização de RV e RA é um segmento promissor que está em ascensão, muito devido ao avanço tecnológico, computadores com maior poder de processamento e um custo reduzido. Além disso, tem-se o aumento da velocidade em relação às conexões de banda larga, que estão colaborando para que ambientes virtuais tridimensionais distribuídos possam ser carregados em qualquer local do planeta (FUJIOKA, 2015). Além disso, dispositivos como o Google Glass (GGLASS, 2015) e o Oculus Rift (OCULUS, 2015) que adicionam capacidades de RA a óculos especiais, apresentam oportunidades de investigação acadêmica e industrial.

Conforme abordado, tais ambientes proporcionam maior interação do usuário com as cenas permitindo um aprendizado ou imersão mais eficientes. No entanto, a construção dos mesmos não é uma tarefa trivial. É preciso dominar uma ou algumas das tecnologias que são empregadas para tal finalidade e, além disso, quando estes são modelados, geralmente são fortemente conectados com a ferramenta para o qual foram desenvolvidos, não abrindo margens para reutilização ou customização em outros sistemas fora da própria arquitetura interna. Existem diversas tecnologias para desenvolvimento de conteúdo em três dimensões e aquelas que são aderentes a esta pesquisa são abordadas na Seção 0.

2.3 Tecnologias para desenvolvimento de conteúdo 3D

A criação de imagens 3D em AV necessita de linguagens de descrição específicas capazes de expressar de forma precisa características do objeto que se deseja modelar, tais como cor, textura, forma e posicionamento. Além disso, os ambientes virtuais tridimensionais estão cada vez mais presentes na Web, permitindo que um grupo maior de usuários tenha acesso a este tipo de recurso de uma forma mais simples e eficiente, portanto, sendo necessário que o usuário tenha apenas acesso à Internet.

Na concepção desses ambientes existem diversas linguagens utilizadas, tais como JAVA3D (JAVA3D, 2014), VRML (W3C-d, 2014), X3D (WEB3D-a, 2014), X3DOM (X3DOM, 2014), WEBGL (KHRONOS, 2014), UNITY 3D (UNITY3D, 2014), TREEJS (TREEJS, 2015), entre outras. Para selecionar as tecnologias utilizadas nesta investigação foram utilizados três critérios:

A. Ser um padrão Web aberto – Por ser padronizada permite que se tenha uma previsibilidade maior em relação aos resultados, bem como na longevidade da aplicação pelo fato do padrão ser aberto e não proprietário;
B. Permitam a disponibilização e visualização de seu conteúdo por meio de browsers e uma conexão com a Internet - Esse ponto se faz muito importante pois a pesquisa aqui realizada tem uma abordagem de distribuição baseada em serviços da web e não em ferramentas de renderização específicas; e
C. Seja utilizado pela comunidade ou indústria - permitindo exploração e testes em uma maior variedade de ambientes.

Com base nos critérios expostos foram selecionadas as tecnologias X3D, X3D X3D (WEB3D-a, 2014), X3DOM (X3DOM, 2014) e WEBGL (KHRONOS, 2014), abordadas nas subseções 2.3.1 a 2.4.

2.3.2 X3D (*Extensible 3D*)

X3D é um padrão ISO baseado em XML utilizado para representação de informações, formas e comportamentos em ambientes 3D complexos na Web, sendo referenciada como a sucessora de VRML[1] (WEB3D-a, 2014). Ela possui diversas características que podem ser combinadas para descrever AV em três dimensões para uso na engenharia e visualização

[1] Para um maior aprofundamento em relação a VRML existem diversos trabalhos que elencam suas características e estrutura (ZIVIANI, 1998; GONÇALVES, 2002; AQUINO, 2005; LIGHTHOUSE3D, 2014). Em web3d-c (2014) pode ser encontrada toda a especificação VRML bem como a descrição de todos os seus nós.

científica, CAD e arquitetura, visualização médica, treinamento e simulação, multimídia, entretenimento e educação, entre outros (AQUINO, 2007; FUJIOKA, 2008; WEB3D-a, 2014).

Mesmo sendo uma nova especificação de VRML, X3D foi separada em três especificações ISO em função do suporte dado a linguagens de programação e formatos de arquivo: as **ISO 19775:200x (**WEB3D-d, 2014**)** – que descrevem de forma abstrata as funcionalidades de sistema referente aos modelos estruturais e de execução, incluindo também aquelas relacionadas à programação externa; já as **ISO 19776:200x (**WEB3D-d, 2014**)** – apresentam um conjunto de descrições referente a formatos de arquivos texto (VRML e XML) e um binário comprimido; e por fim o grupo das **ISO 19777:200x (**WEB3D-d, 2014**)** apresenta um conjunto de mapeamentos para linguagens de programação que pode ser utilizado para realizar acesso externo às cenas do ambiente, como JAVA (JAVA, 2014) e *ECMAScript*[2] também conhecida como *JavaScript* (WEB3D-d, 2014).

Para interação entre as linguagens de programação e as cenas é utilizada uma interface conhecida como SAI (*Scene Access Interface*) (WEB3D-e, 2014), a qual permite que sejam aplicadas alterações no grafo da cena em tempo de execução sem a necessidade de recarregamento da cena, possibilitando a construção de ambientes dinâmicos ou que se adaptem em tempo real de acordo com regras estabelecidas pelo ambiente (WEB3D-e, 2014; WEB3D-f, 2014).

A SAI pode ser escrita no próprio arquivo X3D utilizando-se linguagem de *script* como a já mencionada *ECMAScript* ou externamente com a linguagem de programação JAVA (WEB3D-e, 2014; WEB3D-f, 2014). Caso não exista uma API implementada, as especificações da SAI definem regras que permitem a codificação de ferramentas que possam utilizar a API (FALCÃO et al., 2010; WEB3D-e, 2014).

A X3D, assim como sua antecessora, é baseada em uma estrutura hierárquica de nós predefinidos. Uma das principais diferenças entre a VRML e a X3D está no formato adotado pela segunda, XML[3], que é um formato comum na troca de mensagens e integração entre aplicações na web. Também foram adicionadas a ela novas características em relação à VRML. Além destas melhorias, outro grande avanço foi a implementação de uma arquitetura modular que promove uma maior flexibilidade e extensibilidade. Com X3D é possível

---

[2] ECMASCRIPT: É uma linguagem de programação baseada em scripts, padronizada pela Ecma International na especificação ECMA-262. A linguagem é bastante usada em tecnologias para Internet, sendo esta base para a criação do JavaScript/JScript (ECMASCRIPT, 2015).

[3] W3C. **Extensible Markup Language (XML).** Disponível em http://www.w3.org/XML/. Acesso em: 15 jul. 2014.

desenvolver aplicações sem a necessidade de implementar todas as funcionalidades definidas pela especificação de uma só vez (WEB3D-a, 2014; FALCÃO et al, 2010).

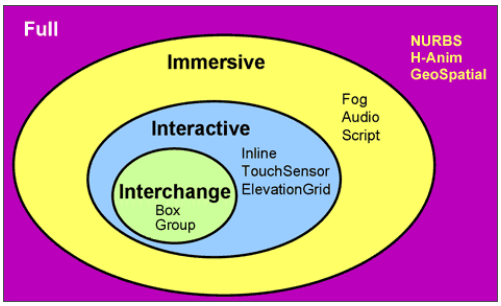
Observando-se que a maioria dos domínios das aplicações que trabalham com X3D não precisa implementar todos os recursos da linguagem e muito menos existe um suporte garantido em todas as plataformas em função da variedade de funcionalidades definidas pela especificação, dois conceitos foram introduzidos: o componente (*Component*) e o perfil (*Profile*) (AQUINO, 2007; FUJIOKA, 2008).

Um componente é um conjunto de nós que tipicamente possui em comum as funções. Tais artefatos estão estruturados em relação ao nível de suporte dado pelo perfil às funcionalidades. Estes podem ser agrupados de acordo com os objetivos ou necessidades requeridas pelo conteúdo. Um Browser X3D (ferramentas utilizadas para renderizar as cenas X3D) pode utilizar as informações do cabeçalho presente no arquivo, ou as opções de execução de uma API, para otimizar o processamento das cenas carregando apenas os componentes e os perfis requeridos, como também ir carregando os módulos de acordo com as necessidades.

O perfil pode especificar o nível de restrição a determinados recursos com o objetivo de simplificar a implementação ou mesmo reduzir consumo de memória, impondo limitações ou campos sem suporte em um dado nó. Os perfis foram concebidos com o objetivo de agrupar funcionalidades para determinados tipos de cena. Em julho de 2014 encontravam-se descritos na especificação um total de quarenta e um (41) componentes (Web3D-g, 2014).

Um perfil pode não conter outro perfil, embora ele possa necessitar de componentes e níveis iguais a de outros perfis. Esta categorização permite que os desenvolvedores de *browsers X3D* não sejam obrigados a implementar grande parte da especificação de uma só vez. Assim, alguns desses ambientes ou objetos podem ser processados em um tempo mais curto de acordo com nível especificado pelo autor na cena (AQUINO, 2007; BRUTZMAN; DALY, 2007; FALCÃO et al, 2010).

Figura 2.2 – Perfis base do X3D



Fonte: WEB3D-h (2014).

A Figura 2.2 ilustra os perfis base disponibilizados por X3D de acordo com a especificação da Web3D (WEB3D-h, 2014), quais sejam: o *Interchange*, *Interactive*, *Immersive e Full*. Tais perfis possuem nível ascendente e incremental conforme as funcionalidades apoiadas.

- *Interchange* – Oferece suporte às funcionalidades para representações geométricas providas pelos nós *Geometry* e *Appearance* (*material e texture*). Não existe tempo real para o modelo de renderização, o que torna este perfil mais simples de ser utilizado e integrado em qualquer aplicação (WEB3D-h, 2014; AQUINO, 2007).
- *Interactive* – Habilita a interação básica ao perfil *Interchange* incluindo vários nós de interatividade para navegação e interação com o ambiente *(PlaneSensor, TouchSensor, entre outros)*, temporização e iluminação adicional (*PlaneSensor e PointLight*) (AQUINO, 2007; FUJIOKA, 2008; WEB3D-h, 2014).
- *Immersive* – Adiciona ao perfil *Interative*, suporte completo a interatividade(*Route*) e todos os grafos 3D, bem como geometria 2D, áudio, colisão sombreamento de cenas (fog) e *scripts* (WEB3D-h, 2014).
- *Full* – Este perfil inclui todos os nós definidos pela especificação (WEB3D-i, 2014).

Além dos citados, também existem adicionalmente os seguintes perfis:

- *CDF (CAD Distillation Format)* – Este perfil oferece suporte à importação de modelos de ferramentas CAD (WEB3D-h, 2014).
- *Core Profile* – Define o mínimo requerido pela especificação X3D. Basicamente é composto de nós Metadata. Descreve cenas com pouca complexidade (WEB3D-h, 2014).

- *MPEG-4 Interactive* – É uma versão reduzida do perfil *Interactive*, projetada para dispositivos móveis (WEB3D-h, 2014).
- *MedicalInterchange* – Este perfil provê formato aberto para troca de geometrias poligonais, dados volumétricos e documentação de acompanhamento entre imagens de sistemas médicos. No entanto, não é padronizado, passando ainda por processamento da ISO (WEB3D-h, 2014).

O desenvolvedor que utiliza X3D se beneficia dos componentes e perfis expostos. Além disso, é possível a estes criarem suas próprias extensões utilizando protótipos[4]. No entanto, para isso é preciso seguir um rigoroso conjunto de regras criadas para evitar a diversidade de extensões que surgiram em VRML, muitas vezes alterando completamente a especificação definida pela linguagem (AQUINO, 2007).

Além do suporte às diversas mídias, formas de interação e a rica API disponibilizada pela especificação X3D, de modo geral, pode-se destacar como principais van*tag*ens desta: não ser obrigatória a implementação de todas as funções de visualização em aplicativos de edição; a flexibilidade e extensibilidade da especificação permite que as comunidades construam suas próprias funções (GEOVRML, 2014); funções evoluídas (NURBS[5], reflexão, entre outras) podem ser incorporadas e integradas a *plug-ins* existentes; ela é base para gráficos 3D interativos de MPEG-4 utilizado no padrão brasileiro de televisão digital (GINGA, 2014); utilizada para aplicações gráficas interativas (WEB3D-h, 2014); e é compatível com VRML, sendo possível a conversão de cenas em VRML para o formato X3D utilizando ferramentas adequadas.

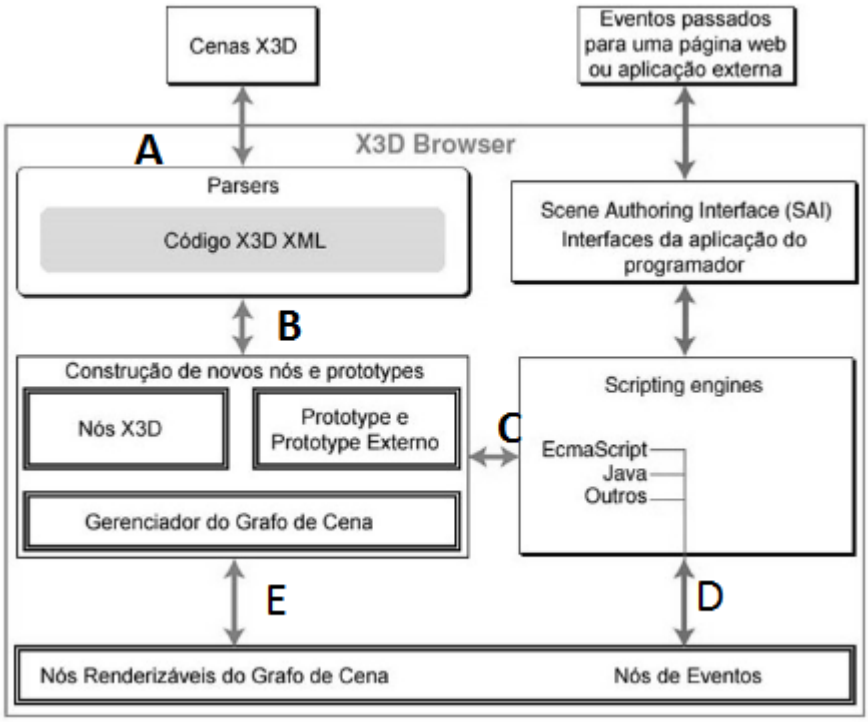
Na Figura 2.3 é apresentada a arquitetura de uma aplicação X3D na qual se verificam as seguintes etapas: (**A)** cena X3D é passada para o browser que identifica o formato de codificação do arquivo utilizando interpretadores (*parsers*); **(B)** Após o processamento realizado pelos *parsers*, os nós são criados e enviados para o *gerenciador de grafo de cena*, que é encarregado pela geração da árvore do grafo e também responsável pela atualização da mesma em função de regras ou interações com o ambiente. Conforme elencado, podem ser utilizados scripts (*Scripts Engines)* e linguagens de programação externas para realizar estas modificações com a SAI; **(C)** Os eventos então podem ser habilitados pelas ferramentas de script ou podem iniciar uma animação na cena; **(D)** pode-se invocar o gerenciador do grafo na

[4] Protótipos: a mesma ideia dos protótipos em VRML, os desenvolvedores podem customizar novos nós a partir de outros nós X3D e/ou outros protótipos (FALCÃO et al, 2010).

[5] NURBS "(non-uniform rational b-splines) – componente de x3d utilizado para construir superfícies tridimensionais definidas por funções paramétricas bivariantes (rosto de uma pessoa, por exemplo)" (AQUINO, 2007, p. 33).

cena para realizar alteração no ambiente e, por fim, **(E)** a atualização da cena que leva em consideração aspectos como a localização do usuário na cena bem como as animações que foram ativadas.

Figura 2.3 – Arquitetura de uma aplicação X3D



Fonte: Adaptado de BRUTZMAN; DALY (2007).

Conforme exposto, X3D possui um rico e variado conjunto de características e funcionalidades. Além disso, outro ponto positivo no contexto da investigação realizada neste trabalho está no formato utilizado para codificar as cenas tridimensionais, o XML, que é um padrão bem utilizado na troca de informações entre serviços da Web e também na integração de sistemas, independentemente de tecnologia.

2.3.3 WEBGL

WebGL é uma API multiplataforma para desenvolvimento de gráficos 3D que permite a renderização na Web com HTML5 (W3Ce, 2014) diretamente no browser sem a necessidade de *plug-in* ou acessórios especiais instalados. Ela é mantida pelo Khronos Group (KHRONOS, 2014), responsável pela especificação do OpenGL (*Open Graphics Library*). O

WebGL é uma biblioteca baseada no OPENGL ES 2.0[6] (*OpenGL for Embedded Systems*), que estende o suporte a *JavaScript*, tornando possível a geração de gráficos 3D iterativos, acelerados por GPU (*Graphics Processor Unit*) (KHRONOS, 2014; HEINZLE, 2013; JUNIOR, 2012).

Para poder desenhar os objetos 3D nas páginas é necessário utilizar o elemento CANVAS[7] do HTML5, o qual é acessado por meio de Interfaces DOM (*Document Object Model*) (W3Cf, 2014). A WebGL é uma tecnologia que permite a concepção de conteúdos complexos em três dimensões. Diversas bibliotecas vêm sendo desenvolvidas para WebGL, tendo como objetivo simplificar a implementação de aplicações 3D em alto nível (LEMOS, MACHADO, 2012).

Em conjunto com as tecnologias HTML5 e *JavaScript*, conforme relatado anteriormente, a WebGL torna factível que aplicativos complexos que demandam um desempenho considerável de placas gráficas sejam implementados para ser executados em navegadores Web.

A comunicação entre a camada HTML e o código *JavaScript* é provida pela API WebGL. Esta permite a execução de *shaders*[8] que efetuam a renderização dos objetos modelados em Aplicações Web. O contexto WebGL é acessado a partir da *tag* canvas e operações da API invocadas a partir desse contexto.

---

[6] OPENGL ES – é um padrão para gráficos 3D em sistemas embarcados como tablets, Smartphones e consoles de videogame, esta API é baseada na *OPENGL* (KHRONOS, 2014).

[7] CANVAS – "É uma área bitmap de modo imediato que pode ser manipulada pelo javascrip. Este modo imediato se refere à maneira com que o canvas renderiza os pixels na tela. A HTML5 canvas repinta completamente a área bitmap a cada frame através das chamadas da api canvas do javascript" (SLIVINSKI, 2011, p. 25).

[8] OPENGL. **Shader.** Disponível em http://www.opengl.org/wiki/shader. Acesso em: 24 JUL. 2014

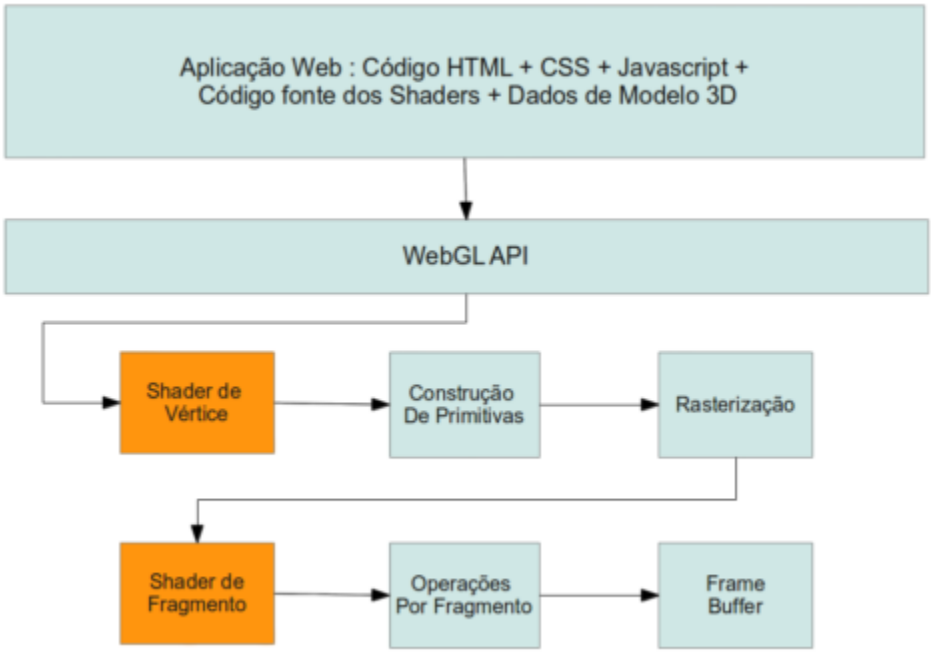
Figura 2.4 – Imagem de vídeo do Jogo Quake 2

Fonte: GOOGLECODE (2014)

Em Anttonen et al (2011) é mencionado o jogo Quake 2 ilustrado pela Figura 2.4 , sobre o qual Junior (2012) reforça a percepção de que com a utilização dessas tecnologias combinadas, recursos sofisticados podem ser aproveitados inclusive por jogos.

A Figura 2.5 ilustra o ciclo de execução de uma aplicação desenvolvida utilizando a tecnologia WebGL, numa tentativa de reproduzir graficamente a comunicação entre o *JavaScript* e os principais elementos que conduzem ao processamento dos objetos pela API WebGL.

Figura 2.5 – Pipeline de Aplicação Desenvolvida com WEBGL



Fonte: JUNIOR (2012).

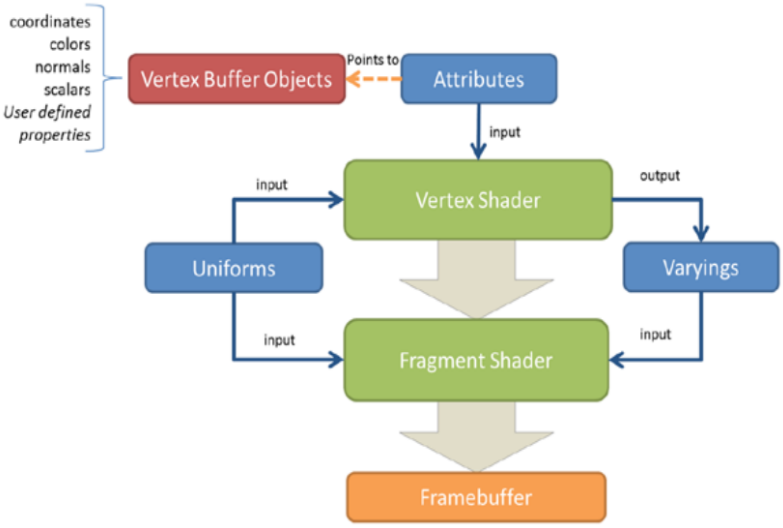
Conforme a estrutura exposta na Figura 2.5, a WebGL encontra-se adaptada ao arranjo de uma página web com HTML5 que comumente é implementada utilizando folhas de estilo CSS[9] para definição da aparência dos elementos e textos, bem como formatação do layout. O arranjo de página HTML é realizado pela definição de *tags* que podem conter elementos *canvas* nos quais o ambiente WebGL é definido. *JavaScript* é utilizado para aplicação de eventos conforme as solicitações realizadas e os códigos fontes dos *shaders* ficam embutidos em *tags*.

Vertex Shader ou *Shader* de vértice é o *shader* responsável pela execução de códigos de finalidade geral inerentes à manipulação de vértices. A sua invocação é realizada para cada vértice e manipula informações do mesmo (CANTOR; JONES, 2012; JUNIOR, 2012; MUNSHI; GINSBURG; SHREINER, 2009). O esquema utilizado para a passagem de dados ao Vertex Shader é ilustrado pela Figura 2.5 , na qual se tem um código fonte *shader* codificado em OpenGL, JavaScript fazendo utilização de matrizes de vértices (*vertex arrays*) obtém os dados VBO10 (*Vertex Buffer Objects*), que são disponibilizados pelos vértices, para isso faz utilização dos *Attributes*. Nas *tags* <script> existe o atributo type contendo o valor

[9] W3C. **Cascading Style Sheets – CSS.** Disponível em http://www.w3.org/Style/CSS/. Acesso em: 14 jun. 2014.
[10] OPENGL. **Vertex Especifications**. Disponível em http://www.opengl.org/wiki/Vertex_Specification. Acesso em: 20 jul. 2014

type = ‘x-shader/x-vertex’ que indica a existência de código fonte *shader*. (CANTOR; JONES, 2012; JUNIOR, 2012)

Figura 2.6 – Detalhes da API WebGL



Fonte: JUNIOR (2012, p. 22).

No esquema da Figura 2.6, os *Atributtes* são tipos de variáveis utilizados pela WebGL para ter acesso aos dados presentes nos VBO. O código *Vertex Shader* é executado em paralelo pela GPU utilizando o valor dos atributos em cada ciclo de renderização. Tais valores são dinâmicos não se repetindo em cada ciclo (CANTOR; JONES, 2012; JUNIOR, 2012; OPENGL, 2014).

Os *Vertex Shaders* e *Fragment* utilizam os parâmetros *Uniforms* como variáveis de entrada, sendo que diferente dos *Attributes*, os *Uniforms* são mantidos constantes durante todo o ciclo de renderização (CANTOR; JONES, 2012; JUNIOR, 2012; OPENGL, 2014). Quando é preciso trocar dados entre os *Shaders Vertex* e *Fragment*, variáveis do tipo *Varyngs* são utilizadas. Elas funcionam como tipos estáticos na aplicação: uma vez definidas em um dos *shaders* podem ser acessadas/manipuladas por ambos utilizando a mesma nomenclatura (CANTOR e JONES, 2012; JÚNIOR, 2012).

Esta seção fez uma breve introdução a WebGL e sua API, uma tecnologia bem recente e interessante. Com ela é possível construir ambientes tridimensionais, jogos, dentre as coisas possíveis com a computação gráfica, e tem como vantagem poder ser renderizada nos principais navegadores Web sem a necessidade de instalação de ferramentas ou *plug-in* (KHRONOS, 2014).

A WebGL vem sendo investigada por diversos autores (HEINZLE; MONTIBELER; HOGREFE, 2013; JUNIOR, 2012; LEMOS, MACHADO, 2012), e além disso tem apoio de vários *players* de mercado incluindo a Google (GOOGLE, 2015). De acordo com Junior (2012), em contrapartida a essas vantagens está o fato de ela ter que ser codificada em *javascript*, o que pode comprometer a mesma, uma vez que o navegador pode estar com o suporte a esta linguagem desabilitado.

2.3.4 X3DOM

X3DOM é um *framework open source*[11] para execução de gráficos 3D na web. Foi desenvolvido como esforço para integrar o HTML5 a ambientes 3D declarativos dando suporte a discussões da Web3D e os grupos da W3C que debatem o assunto (X3DOM, 2014). Com ele pode-se incluir cenas X3D em páginas HTML5 que reproduzem estas como elementos de uma árvore DOM sem a necessidade de *plug-in* ou *browsers* X3D específicos. Isto é possível devido à renderização do conteúdo a partir da WebGL (SOURCEFORGE, 2014; X3DOM, 2014).

O código da Figura 2.7 ilustra uma página HTML na qual se utiliza X3DOM para embutir uma cena X3D. Pode ser observado também que a cena está presente a integração entre o conteúdo X3D, HTML e folhas de estilo CSS responsáveis pelo carregamento de uma imagem como plano de fundo na cena e um cubo vermelho.

Figura 2.7 – Página HTML com X3DOM

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Adaptação do CSS Integration Test da documentação x3DOM</title>
<link rel="stylesheet" href="http://www.x3dom.org/download/x3dom.css">
<script src="http://www.x3dom.org/download/x3dom.js"></script>

<style>
#fujioka_element {
    width: 20%;
    height: 5%;
    background: #000
    url(http://cin.ufpe.br/~rcf4/images/logo.jpg);
}
</style>

</head>
<body>
    <h1>Fujioka Element</h1>
    <x3d width="400px" height="300px" id="fujioka_element">
       <scene>
         <shape>
           <appearance>
            <material diffuseColor='red'></material>
           </appearance>
           <box> </box>
         </shape>
       </scene>
    </x3d>
</body>
</html>
```

[11] OPEN SOURCE. **Open Source**. Disponível em: <http://opensource.org/<. Acesso em: 20 jul. 2014

Fonte: O Autor

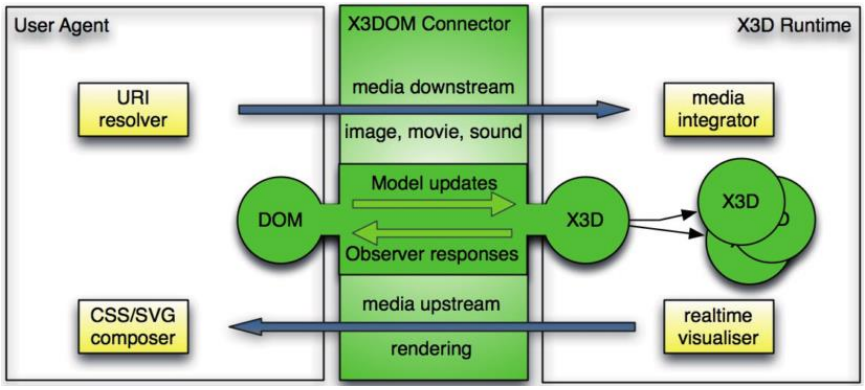
A Figura 2.8 contém o resultado redenderizado do código apresentado na Figura 2.7. Neste exemplo está habilitada a manipulação do cubo presente na cena.

Figura 2.8 – Renderização de HTML utilizando X3DOM para inclusão de X3D

Fonte: O Autor.

A Figura 2.9 exibe a arquitetura proposta para o sistema X3DOM (BEHR et al, 2010), que tem como objetivo renderizar cenas X3D em HTML DOM, permitindo aos desenvolvedores de aplicação manipular o conteúdo 3D, realizando operações de adição, remoção ou alterando os elementos DOM na cena sem utilizar nenhum *plug-in* específico ou muito menos utilizar interfaces como a SAI da Web3D (WEB3D-j, 2014). Eventos de HTML podem ser utilizados nos objetos 3D, como o "*onclick*", por exemplo. Esta operação é a execução de um clique do botão de um mouse, dispositivo utilizado para interação com o sistema operacional.

Figura 2.9 – Arquitetura proposta para sistemas utilizando a arquitetura X3DOM incluindo o UA (*User Agent - Browser*), o *runtime* X3D e o conector X3DOM.



Fonte: BEHR et al (2010).

O navegador Web dá suporte à estrutura de árvore DOM, integrando e fazendo composição no processamento da cena final. Também pode disponibilizar algum endereço URI (*Uniform Resource Indentifier*) para que imagens, vídeos, áudios e cenas X3D possam ser baixados. O fornecimento de serviços para gerar e atualizar a cena X3D pode ser invocado sempre que houver algum tipo de interação com a cena, seja durante a navegação ou acionados temporalmente, estes serviços são disponibilizados pela X3D *runtime*.

Um componente fundamental nessa estrutura é o X3DOM *Connector*, pois ele é a base do núcleo interno da arquitetura, sendo responsável pela conexão entre a árvore DOM com o X3D *runtime*. As alterações, tanto no DOM como na representação X3D, são distribuídas pelo X3DOM *Connector*, que atualiza o ambiente quando os nós são manipulados ou quando há alteração em atributos. Também manipula o *upstream*[12] e o *downstream*[13] de mídias tais como a visualização da cena, imagens, vídeos e áudios.

X3DOM traz todas as vantagens de X3D, incluindo a de não necessitar de ferramenta especial para ser renderizada na web, uma vez que é compatível com os principais browsers existentes. Além disso, ela é renderizada utilizando a WebGL, deste modo, se o browser for capaz de executar objetos em WebGL também vai ser capaz de executar X3DOM. Entretanto, assim como WebGL, trata-se de uma tecnologia recente que ainda está sendo investigada e melhorada.

---

[12] *Upstream* – refere-se à medição de banda em relação ao envio de informações do computador para Internet.

[13] *Downstream* – refere-se à medição da banda de download, a velocidade em que se pode baixar dados da Internet.

2.4 Considerações

Existem diversas linguagens para descrição de mundos virtuais tridimensionais, como exposto anteriormente. No entanto, este trabalho descreveu aquelas que se enquadravam dentro dos critérios estabelecidos: (a) Ser um padrão web aberto e padronizado; (b) Permita a disponibilização e visualização de seu conteúdo por meio de browsers e uma conexão com a Internet; e (c) Seja utilizado pela comunidade ou indústria. No entanto, a utilização de outras tecnologias tais como o Java3D (JAVA3D, 2015), UNITY (UNITY, 2015), TREEJS (TREEJS, 2015) e as outras citadas na Seção 0 podem ser exploradas adaptando-se a arquitetura de referência proposta nesse trabalho. O objetivo deste trabalho não é verificar dentre estas alternativas qual a mais eficiente ou a mais simples de ser utilizada, pois a aderência de cada uma delas vai depender do propósito do projeto no qual ela é empregada. Um comparativo entre diversas tecnologias é discutido e investigado nos trabalhos de (BARBOSA et al., 2012; LEMOS; MACHADO, 2012; SILVA; SILVA, 2011).

Esse trabalho investiga dois segmentos da literatura (AV e Reuso) com o intuito de gerar evidências de que a combinação de ambas otimiza o processo de geração de AV. Desta forma, no que se refere à contextualização bibliográfica, este capítulo apresentou e descreveu as tecnologias relacionadas à construção de mundos virtuais em três dimensões utilizadas na validação da arquitetura apresentada nesse trabalho. O próximo capítulo discorre sobre aspectos essenciais para o entendimento deste trabalho, relativo à construção de sistemas utilizando a reutilização de software, bem como a Arquitetura Orientada a Serviços (SOA).

3 ENGENHARIA DE SOFTWARE, ARQUITETURA ORIENTADA A SERVIÇOS (SOA) E COMPUTAÇÃO EM NUVEM

Este capítulo apresenta embasamento teórico referente ao desenvolvimento de sistemas baseados em reuso, elencando as técnicas investigadas neste trabalho. Assim, é apresentado referencial teórico sobre reuso de software e algumas definições no que se refere às abordagens e estratégias pesquisadas, seguido de um detalhamento mais amplo em relação a Reuso, Computação em Nuvem, Arquitetura Orientada a Serviços (SOA) e *Web Services*.

3.1 Engenharia de Software

Engenharia de Software (ES) é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de forma econômica, que seja confiável e funcione de forma eficiente em máquinas reais (MAHONEY, 2004).

Em outra visão, Musa (1975) define ES como a aplicação da abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção do Software.

Em 1968, diante da crise do software, em uma conferência na Organização do Tratado do Atlântico Norte (OTAN), McIlroy (1968) apresenta as primeiras ideias e os benefícios referentes ao conceito que existe de reuso de software. Em seu artigo, McIlroy discorre sobre a produção de software em massa comparando com a indústria e tomando como exemplo a reutilização de um catálogo de rotinas.

Complementarmente, Almeida et al. (2007) e Krueger (1992) concordam que durante esta conferência foi apresentada uma definição simples, porém poderosa, de reuso de software, momento em que, segundo Krueger (1992), deveria ter se tornado um padrão em Engenharia de Software.

3.1.1 Reuso de Software

A reutilização é uma estratégia de resolução de problemas utilizada na maioria das atividades do ser humano (ALEXANDER et al., 1977; ALMEIDA et al., 2007; PRESSMAN, 2011). O reuso nasceu da observação de que software frequentemente segue padrões similares, sendo possível explorá-los em diferentes níveis e, assim, empregar menos esforços em soluções para problemas já resolvidos (MEYER, 2000).

Além disso, o reuso propõe um conjunto sistemático de processos, técnicas e ferramentas que auxiliam na melhoria da produtividade, manutenção e da qualidade, tanto do software quanto do processo de desenvolvimento, podendo ser obtido por meio da reutilização de código-fonte (trechos de código, *templates*, procedimentos ou bibliotecas), modelagem (padrões de projeto) ou por uma combinação de ambos (CARROMEU, 2007; GAMMA et al., 1995; WESCHTER; TOURINE, 2008).

Ainda conforme Frakes e Terry (1996), a reutilização de software pressupõe o uso de artefatos existentes do software ou o conhecimento para criação de novo software. Nos últimos vinte anos, foram desenvolvidas muitas técnicas para apoiar o reuso de software (ALMEIDA et al., 2007; SOMMERVILLE, 2011). Estas têm trazido uma série de benefícios no desenvolvimento de projetos com maior produtividade, maior confiabilidade e melhores estimativas de custos (ALMEIDA et al., 2007; GAMMA et al., 1995).

Assim, diante do exposto, várias abordagens e plataformas de reuso têm sido propostas na literatura com o objetivo de tratar e otimizar as vantagens elencadas para reuso no desenvolvimento de sistemas, tais como engenharia de domínio (SOMMERVILLE, 2011), classes abstratas (PREE, 1995), Frameworks (JOHNSON, 1992; FAYAD; SCHMIDT, 1997; FAYAD; JOHNSON; SCHMIDT, 1996; PREE 1999; BRAGA; GERMANO; MASIERO, 1999; BRUGALI; SYCARA, 1999; BRAGA, 2003), padrões de projeto (GAMMA et al., 2000; COALLIER, 2007), padrões arquiteturais (BUSCHMANN et al. 1996; BASS et al. 2002; GOAER et al., 2008), linguagem de padrões (GAMMA et al., 1993; GAMMA et al., 1995; FAYAD, 1999; BRAGA, 2003; ARAGON, 2004), componentes (SAMETINGER, 1997; LAU, 2006), geradores de aplicação (FRANCA; STAA, 2001; SMARAGDAKIS; BATORY, 2002; BATORY, 2005; WANG et al., 2008), linha de produtos de software (LPS) (CZARNECKI et al., 2005; CLEMENTS; NORTHROP, 2002; BATORY et al., 2002; GILL, 2006; BOSCH 2001), desenvolvimento baseado em componentes (DBC) (DSOUZA; WILLS, 1999; CRNKOVIC et al., 2002) e aspectos (KICZALES, 1997; TARR, 2000), empacotamento de sistemas legados, sistemas orientados a serviços, aplicações verticais configuráveis e biblioteca de programas (SOMMERVILLE, 2011) nas quais os artefatos são organizados de modo a compor novos artefatos, que, por sua vez, podem ser reutilizados em outros projetos (WESCHTER, 2008; CARROMEU, 2007).

Na sequência são contextualizadas as técnicas de reuso comuns na implementação de aplicações (FUJIOKA, 2011; SOMMERVILLE, 2011) e para a investigação realizada neste trabalho.

3.1.2 Padrões de Software

Um padrão é uma entidade que documenta um problema recorrente, uma solução e a situação em que deve ser aplicado. Os padrões podem ser utilizados em diversas etapas do desenvolvimento de software. Um dos benefícios apresentados por Vlissides (1998) é a possibilidade de armazenar experiências, tornando-as acessíveis aos não experientes.

Conforme Gamma et al. (2000), um padrão apresenta quatro elementos essenciais: o **nome** que é utilizado para referenciá-lo; o **problema** que descreve quando aplicá-lo; a **solução** que especifica os elementos que compõem o projeto, seus relacionamentos, suas responsabilidades e colaborações e; por fim, as **consequências,** que são os resultados e análises das vantagens e desvantagens da aplicação do padrão, sendo críticas para a avaliação e compreensão dos custos e benefícios do seu uso.

3.1.3 Desenvolvimento baseado em componentes

O Desenvolvimento Baseado em Componentes (DBC) surgiu como uma nova perspectiva para o desenvolvimento de software, caracterizada pela composição de partes já existentes. Ela consiste no princípio de dividir para conquistar, de forma que quebra blocos monolíticos em componentes interoperáveis gerenciando a complexidade, isto é, dividindo problemas complexos em partes menores. Desta forma, soluções maiores e mais complexas são implementadas fazendo uso de partes menores e mais simples (CHEESMAN; DANIELS, 2001; SAMETINGER, 1997).

Em visão complementar, Singhal e Zyda (1999) definem componentes como uma unidade que possui interfaces especificadas previamente definidas, por meio de contratos e dependências de contexto explícitas.

A baixa eficiência do modelo orientado a objetos foi um dos fatores que motivaram o DBC, pois as classes desenvolvidas com OO são muito especializadas e dependentes de ligações que ocorrem geralmente em tempo de compilação. Tais características terminam restringindo as possibilidades de reutilização e distribuição (NAKAMURA, 2012).

Ainda conforme Sommerville (2011), existem quatro características essenciais a este tipo de abordagem: (i) devem ser especificados por suas interfaces e com garantia de independência; (ii) deve seguir padrões para garantir as integrações com outros componentes;

(iii) deve existir um *middleware*[14] para apoiar a integração entre componentes; e (iv) deve existir um processo de desenvolvimento voltado à engenharia de software baseada em componentes (CBSE).

Além disso, Henninger (1997) elenca como sendo um ponto chave para obtenção de melhores resultados com DBC a estruturação de um repositório com acesso e disponibilização dos componentes. Diante disto, o Quadro 3.1 exibe os quatro requisitos elencados por Guo e Luqui (2000) como sendo essenciais para estruturação de um repositório de componentes. A definição de interfaces básicas está relacionada às operações que podem ser executadas pelo componente, seguido da documentação que descreve a forma, como e quais parâmetros devem ser utilizados para invocar cada operação. Além disso, devem possuir um padrão para facilitar a manutenção e extensão como também um modelo de classificação para facilitar a localização dos componentes.

Quadro 3.1 – Requisitos para estrutura de um repositório de componentes

Interface para as operações básicas no manuseio de componentes (uso, pesquisa e disponibilização).
Disponibilização de documentação sobre cada componente.
Estrutura padrão para os componentes.
Esquema para classificação nos mais diversos domínios.

Fonte: Guo e Luqui (2000).

Complementando, Werner e Braga (2000) classificam esses em: locais, específicos a um domínio e de referência. Ainda compartilhando a mesma visão, Werner e Braga (2000) identificam que a chance de um desenvolvedor reutilizar um dado componente está diretamente relacionada à disponibilidade com que um componente pode ser localizado e recuperado.

3.1.4 Frameworks

Na literatura existem diversas definições para os *Frameworks*. Segundo Firesmith (1994), *Framework* é uma coleção de classes colaborativas que capturam os padrões em pequena escala e mecanismos maiores que implementam requisitos e projetos em comum

[14] *Middleware* – consiste em uma aplicação intermediária que faz a integração entre dois sistemas.

num domínio de aplicação específico. Johnson e Foote (1988) conceituam *Framework* como um conjunto de classes que constituem um projeto abstrato para soluções de uma família de problemas. Já para Mattsson (1996), trata-se de uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com um grande potencial de especialização.

Assim sendo, um *Framework* orientado a objetos é uma arquitetura que permite a reutilização de todo ou parte de um sistema. A representação é feita por meio de um conjunto de classes abstratas e concretas que se relacionam pela forma como as suas instâncias interagem, de modo que é disponibilizada uma solução reutilizável para uma família de problemas semelhantes. Esse conjunto de classes deve ser extensível e flexível de maneira que seja possível a construção de várias aplicações empregando esforço reduzido uma vez que grande parte das funcionalidades será reaproveitada e não criada, especificando apenas as particularidades de cada software. Portanto, assim como a maioria das técnicas de reutilização, a abordagem de *Frameworks* pode reduzir drasticamente o custo inerente ao desenvolvimento de aplicações, pois uma boa parte do núcleo já está implementada, precisando apenas de adaptações (JONSHON, 1997; SOMMERVILLE, 2011; WIRFS-BROCK, JONHSON, 1990).

Classificação e tipos de framework

Existem três tipos de classificação atribuídos aos *frameworks*, seja quanto à forma de utilização ilustrada no Quadro 3.2 e quanto ao propósito de utilização, conforme descrito no Quadro 3.3 (FUJIOKA, 2011; PROTA, 2009; SILVA, 2000; SILVA e OLIVEIRA, 2006).

Quadro 3.2 – Características dos tipos dos frameworks com relação à forma de utilização.

Tipo	Característica
Caixa Branca	Foca na herança de classes; Estende ou modifica funcionalidades pela definição de subclasses e pela sobreposição de métodos;
Caixa Preta	Foca na composição dos componentes; Usa a funcionalidade já presente no framework; As entidades internas do framework não podem ser vistas ou alteradas; As instanciações e composições feitas determinam as particularidades da aplicação;
Híbrido(Caixa Cinza)	A maioria dos frameworks apresenta uma organização híbrida; Também são conhecidos por frameworks de caixa-cinza; Possuem funcionalidades prontas e aquelas que podem ser criadas ou alteradas A grande maioria deles são de caixa-branca com algumas funcionalidades já prontas (caixa-preta);

Fonte: PROTA (2009) ; FUJIOKA (2010).

Em relação à forma de utilização, os *frameworks* podem ser classificados em: caixa branca, caixa preta ou híbrido. Os frameworks de caixa branca são focados na herança de classes, estendem ou modificam uma funcionalidade pela definição de subclasses com sobreposição de métodos (SILVA, 2000). Já nos de caixa preta, os desenvolvedores devem saber apenas quais objetos estão disponíveis e as regras para combiná-los. Esses objetos não podem ser vistos ou alterados. Nesse tipo de *framework* o reuso se dá pelas conexões entre os componentes, não havendo preocupação em saber como eles realizam as tarefas individuais. As instanciações e composições feitas determinam as particularidades da aplicação (SILVA, 2000).

A maioria dos *frameworks* apresenta uma organização híbrida, e são também conhecidos como *frameworks* de caixa cinza, pois existem funcionalidades prontas e aquelas que podem ser criadas ou alteradas (SILVA e OLIVEIRA, 2006; PROTA, 2009, FUJIOKA, 2011).

Quadro 3.3 – Características dos tipos dos frameworks com relação à finalidade

Tipo	**Característica**
Suporte	São raros; provê serviços de nível de infraestrutura (e não de aplicação); Acesso a arquivos; Computação distribuída; *Device Drivers*
Aplicação	Também chamados de framework horizontal; encapsula conhecimento (“expertise”); aplicável a uma vasta gama de aplicações; resolve apenas una fatia do problema da aplicação, exemplo: framework para construção de interface.

Domínio	Também chamado de framework vertical; encapsula conhecimento (“expertise”); aplicável a aplicações pertencendo a um domínio particular de problema; resolve boa parte da aplicação; Exemplo: framework para construir aplicações de controle de manufatura.

Fonte: PROTA (2009) ; FUJIOKA (2011).

Em relação aos problemas que podem ser tratados por *frameworks*, pode-se categorizar em horizontais, verticais ou de infraestrutura. Os *frameworks* horizontais destinam-se a resolver apenas uma parte do problema da aplicação e por esse fato conseguem atender a uma grande parcela do mercado. Já os verticais, encapsulam os conhecimentos aplicáveis a sistemas de um domínio particular. Destinam-se a resolver todo ou boa parte do problema, podendo gerar aplicações inteiras (PROTA, 2009; SILVA, 2000; SILVA; OLIVEIRA, 2006). Os de infraestrutura buscam solucionar problemas no nível de infraestrutura (e não de aplicação).

Ainda segundo Sommerville (2007, p. 282), que concorda com (FAYAD; SCHMIDT, 1997), a classificação dos frameworks pode ser feita de três formas distintas:

- *Frameworks* de infraestrutura de sistema - esses frameworks são compatíveis com o desenvolvimento das infraestruturas de sistemas, como comunicação, interfaces de comunicação, interfaces com o usuário e compiladores (FAYAD; SCHMIDT, 1997);
- *Frameworks* de integração com *middleware* - consistem em um conjunto de classes de objetos-padrão e associados, que aceitam a comunicação de componentes e a troca de informação. Entre os exemplos desse tipo de framework estão CORBA (CORBA, 2014), COM e DCOM (MICROSOFT, 2014), e *Java Beans* (ORACLE, 2014); e
- *Frameworks* de aplicação corporativa - ocupam-se de domínios específicos de aplicações, como telecomunicação ou sistemas financeiros. Eles incluem o conhecimento de domínio de aplicações e são compatíveis com o desenvolvimento de aplicações para o usuário final.

O *framework* adaptado neste trabalho é o de **infraestrutura**, pois provê soluções para gerenciar concorrência, comunicação entre outros aspectos de infraestrutura genéricos, não focando em um tipo de problema específico. Em relação ao tipo se enquadra no tipo **caixa cinza,** e, em relação à sua finalidade é classificado como de domínio, pois trata de problemas

comuns em aplicações Web tais como controle de acesso, sessões, persistência e separação por camadas.

Framework e padrões de projeto

Padrões descrevem problemas no ambiente e o núcleo da sua solução, de forma que é possível utilizá-los inúmeras vezes, sem nunca aplicá-los do mesmo modo (ALEXANDER; ISHIKAWA; SILVERSTEIN, 1977). São considerados, ainda, como soluções que estão em um nível mais abstrato do que *frameworks*, os quais são normalmente implementados utilizando linguagens orientadas a objetos como Java,[15] C++[16], entre outras. Conforme contextualizado, tais padrões descrevem o projeto de uma solução para um problema recorrente, podendo incluir um exemplo de implementação. Consequentemente, *frameworks* são mais específicos que padrões de projetos, pois estão relacionados a um domínio de aplicação, um aspecto de infraestrutura ou de integração de *middleware*.

Enquanto um *framework* pode possuir vários padrões de projeto, o contrário não é verdadeiro.

Framework e componentes

Para Barreto (2009) e Gimenes e Huzita (2005), os componentes de software podem ser classificados como unidades independentes, as quais encapsulam dentro de si seu projeto e implementação e também disponibilizam funcionalidades para o meio externo por meio de interfaces bem definidas. Componentes podem apresentar dois tipos de interface: as interfaces fornecidas, que expõem seus serviços, e as interfaces requeridas, por onde o componente explicita suas dependências. Os *frameworks* fornecem interfaces bem definidas e os pontos de extensão que as aplicações devem estender. Contudo, enquanto um *framework* possui necessariamente pontos de extensão, um componente totalmente autocontido não possui interfaces requeridas.

Em função disto, um componente deve ser ligado a uma interface requerida de outro componente, enquanto os pontos de extensão dos *frameworks* são menos exigentes,

[15] Java – Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995. Disponível em: < http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em: 20 out 2014.

[16] C++ - A linguagem C++ foi desenvolvida inicialmente por Bjarne Stroustrup na At&T, de 1979 a 1983, à partir da Linguagem C, tendo como ideia principal a de agregar o conceito de classes, de orientação à objetos, àquela linguagem. Disponível em: <http://www.inf.ufrgs.br/~johann/cpp2004/>. Acesso em: 20 out. 2014.

possibilitando que sejam ligados componentes, classes ou qualquer artefato que realiza o contrato definido.

Assim, conforme Szyperski (1997), pode-se concluir que os componentes podem ser desenvolvidos com a utilização de *frameworks*, os quais por sua vez podem ser desenvolvidos utilizando componentes, ou seja, essas duas tecnologias são complementares de forma que *frameworks* podem ser usados para auxiliar o desenvolvimento de componentes e vice-versa.

3.2 Arquitetura Orientada a Serviços (SOA)

Uma arquitetura de software pode ser considerada um conceito abstrato que dá margem a uma série de interpretações e definições. Conforme Braga (2008), a definição utilizada pelo *IEEE*[17] alega que uma arquitetura de software trata basicamente de como os componentes fundamentais de um sistema se relacionam intrinsecamente e extrinsecamente (ANSI/IEEE, 2000). Arquiteturas de Softwares vêm sofrendo uma mudança estrutural, na qual estão aprimorando suas estruturas antes estáticas, monolíticas e centralizadas para arranjos dinâmicos, modulares e distribuídos (BARESI et al., 2006; NAKAMURA, 2012; QUEIROZ, 2009).

Neste sentido, o modelo arquitetural SOA (*Service-oriented Architecture*) apoiado por tecnologias orientadas a serviços surge como uma nova abordagem de desenvolvimento de software que pode auxiliar na concretização deste objetivo de projeto arquitetural (KRAFZIG et al., 2004). Uma arquitetura orientada a serviços tem como componente fundamental o conceito de serviços. SOA estabelece um modelo arquitetural que faz busca de serviços como principal meio para aprimorar a eficiência, agilidade e produtividade de uma organização (ERL, 2009; NAKAMURA, 2012).

Em Affonso (2009) tem-se:

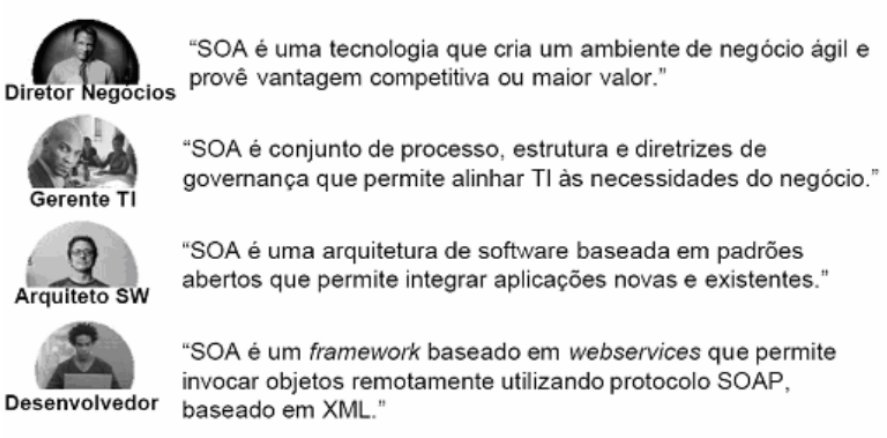
> Essencialmente, SOA é uma arquitetura de software que define a topologia das interfaces, implementação das interfaces e as chamadas das interfaces. SOA é um relacionamento de serviços e consumidores dos serviços, ambos módulos de software grandes o bastante para representar uma completa função do negócio. (AFFONSO, 2009, p. 57).

Para Marzulo (2009) “Arquiteturas Orientadas a Serviço (SOA) representam uma nova abordagem para utilização dos recursos de TI em apoio ao negócio da organização”

[17] IEEE: Instituto de Engenheiros Eletricistas e Eletrônicos é uma organização sem fins lucrativos.

(MARZULO, 2009, p. 123). Já Kumar et al. (2012) descreve SOA como sendo “um estilo arquitetural único para elaborar e planejar a solução corporativa usando serviços” (KUMAR et al., 2012, p. 38). De acordo com Nakamura (2012), “SOA não é uma arquitetura concreta, não é um *framework* ou uma ferramenta” (NAKAMURA, 2012, p. 9), Assim, pode-se definir SOA como um conceito ou modelo que, quando considerado, pode levar ao desenvolvimento de um projeto de software concreto (JOSUTTIS, 2008; NAKAMURA, 2012).

Figura 3.1 – Diferentes interpretações de SOA



Fonte: BRAGA, 2008.

Por não ser um modelo arquitetural concreto, SOA pode levar a diversas interpretações conforme apresentado pela Figura 3.1, logo é importante ressaltar que este trabalho não toma nenhuma destas interpretações, a linha teórica seguida neste trabalho é aquela descrita por Kumar et al. (2012) em que SOA é definida como “um método de arquitetar o aplicativo corporativo como um conjunto de serviços de cooperação” e complementa que o “usuário corporativo pode ser um humano ou uma aplicação cliente” (KUMAR et al., 2012, p. 8).

Embora não seja algo obrigatório em uma Arquitetura SOA, os *Web Services* são geralmente associados à mesma. De fato, com o advento da Internet e surgimento dessa categoria de serviço, SOA teve uma ascensão considerável, daí a referida associação (BELL, 2008; KUMMAR et al.; 2012; MARZULLO, 2009). Por permitir a comunicação entre sistemas escritos em tecnologias diferentes em um nível igual à comunicação de computadores em uma rede (BHAKTI; ABDULLAH, 2010), a tecnologia de *Web Services* tornou-se bem aceita na integração de sistemas heterogêneos, bem como na disponibilização de soluções abertas.

Figura 3.2 - Modelo operacional triangular SOA

Fonte: Adaptado de ENDREI et al. (2004)

A Figura 3.2 ilustra a interação entre os elementos de uma arquitetura SOA, dentro de um modelo organizacional no qual a colaboração entre as entidades segue o paradigma "publique, pesquise, conecte e invoque" ("*publish, find, bind and invoque*"), e o papel de cada um dos seus elementos é (ENDREI et al., 2004):

- Provimento do Serviço – define o comportamento de quem está disponibilizando o serviço;
- Consumo do Serviço – determina o comportamento daquele que representa o cliente; e
- Registro do Serviço – determina o comportamento que a organização deve ter para divulgar seu serviço e o do cliente para localizar. É nesse elemento que são persistidas as informações gerenciáveis acerca dos repositórios, que normalmente possuem:
    - Informações sobre o negócio tais como nome, descrição e contrato;
    - Informações técnicas como linguagens e tecnologias utilizadas, infraestrutura de acesso, entre outras coisas; e
    - Informações sobre os serviços em si tais como os procedimentos, estrutura entre outras coisas.

Esses elementos fazem interação em SOA da seguinte forma: o **Consumidor de**

Serviços realiza localização dinâmica, pesquisando no **Registro de Serviços** em busca de um serviço que atenda os seus critérios. Existindo tal registro, é fornecido um contrato de interface e o endereço no qual o serviço desejado está publicado, de forma que o **Consumidor de Serviço** possa conectar no **Provedor de Serviço** e consumir o serviço localizado. No entanto, para que isso ocorra é necessário que a descrição ou meta-dados do serviço sejam publicados no **Registro de Serviços** pelo **Provedor de Serviços**, caso contrário, o serviço não pode ser localizado (ENDREI et al., 2004).

3.2.1 *Web Services*, padrões e tecnologias

Os *Web Services* são uma implementação da arquitetura SOA (ERRADI e MAHESHWARI, 2005), no entanto não são obrigatórios em uma arquitetura orientada a serviço, conforme Earl (2005) afirma:

> Você não precisa de *Web Services* para implementar aplicativos SOA! Essas palavras você escutará muitas vezes quando se tenta explicar os princípios da arquitetura orientada a serviços. Mas utilizar *Web Services* para implementar SOA é uma magnífica ideia. (EARL, 2005)

Web Service pode ser definido como "a materialização da ideia de um serviço que é disponibilizada na Internet e pode ser acessado em qualquer lugar do planeta." (MARZULLO, 2009, p. 150). Um *Web Service* basicamente pode ser considerado como um sistema de software, identificado por meio de um endereço URI (*Uniform Resource Identifier*), no qual interfaces públicas e contratos são definidos e descritos em XML, de tal maneira que essas definições podem ser descobertas por outros sistemas de software. Estes, então, podem interagir com o *Web Service* de um modo prescrito pela sua definição, usando mensagens baseadas em XML e transportadas por protocolos da Internet.

Por estarem comumente associados na literatura a ideia de que SOA não pode ser implementada sem a utilização de *Web Services* é comum (BRAGA, 2008). No entanto, essa não é uma proposição verdadeira, uma vez que é possível construir aplicações SOA utilizando tecnologias como Java RMI[18] (*Java Remote Method Invocation*), CORBA[19] (*Common Object*

[18] RMI – é uma solução da plataforma Java para tratar da comunicação entre objetos distribuídos.
[19] CORBA – É um padrão proposto pela *Object Management Group* (OMG, 2014) com o propósito de integrar sistemas por meio da troca de dados.

Request Broker Architecture), COM/DCOM[20] (Component Model/Distributed Component Model) ou MOM[21] (*Message Oriented Middleware*) ou REST[22] em substituição aos *Web Services*, entretanto, tais tecnologias apresentam dependências e problemas que geram impedimentos de utilização conforme apresentado no Quadro 3.4.

Quadro 3.4 – Tecnologias e dificuldades encontradas para Arquitetura SOA

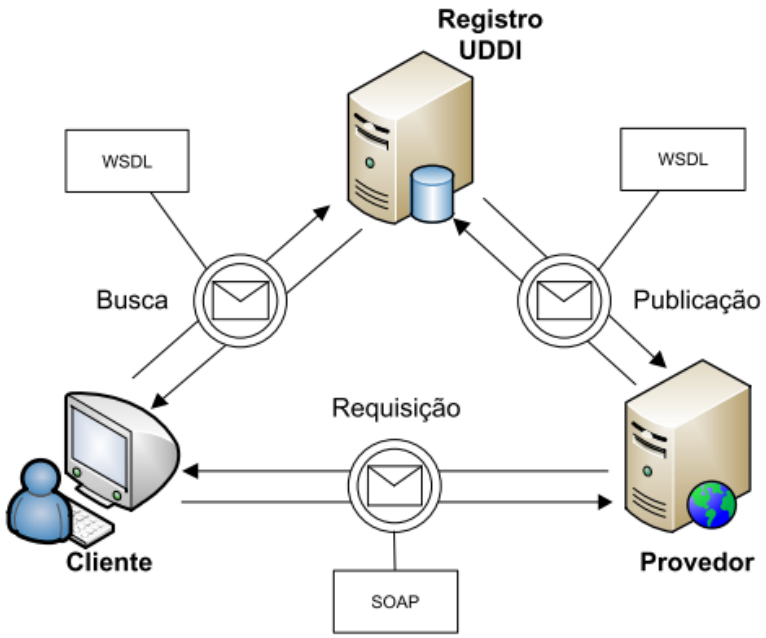
Tecnologia	Dificuldades
DCOM	Tecnologia totalmente proprietária e dependente de plataforma.
CORBA	Muitos protocolos complexos
MOM	Esforço no desenvolvimento e manutenção de uma aplicação intermediária.
RMI	Muito acoplada à tecnologia Java não se comunicando facilmente com outras linguagens de programação.
REST	O principal problema do REST está na flexibilidade que o programador tem para definir o corpo da mensagem que torna os problemas de interoperabilidade mais comuns.

Fonte: O autor.

Por utilizar protocolos abertos de fácil integração entre aplicações desenvolvidas em plataformas diferentes, os *Web Services* terminaram se popularizando em detrimento das demais tecnologias no que se refere a SOA.

[20] COM/DCOM – São tecnologias proprietárias desenvolvidas pela Microsoft (MICROSOFT, 2014) para que objetos possam realizar comunicação entre si em sistemas distribuídos.
[21] MOM – É uma abordagem de integração que utiliza um *Middleware* orientado a mensagens para tratar a comunicação entre objetos. É fracamente acoplado, sendo uma alternativa a métodos sincronizados.
[22] REST – Descreve um estilo de arquitetura para troca de mensagens entre sistemas derivados de vários estilos baseados em redes como a web (FIELDING, 2000).

Figura 3.3 – Modelo básico de SOA com *Web Services*



FONTE: NAKAMURA (2012, p. 9).

Conforme apresentado na Figura 3.3, o modelo de uma arquitetura orientada a serviços para *Web Services* e seus componentes tem sua estrutura fundamentada pela W3C basicamente utilizando três especificações principais:

- SOAP (*Simple Object Access Protocol*) – é o protocolo utilizado para troca de mensagens entre *Web Services*. Provê uma estrutura padrão de empacotamento para transporte de arquivos XML permitindo que a comunicação entre aplicações seja possível independentemente de linguagem ou plataforma (MARZULLO, 2009; PAPAZOGLOU; GEORGAKOPOULOS, 2003);
- WSDL (*Web Services Definition Language*) – Tecnologia XML, a qual descreve de forma padronizada a interface de um Web Service. Ela especifica como o serviço deve ser acessado e quais as operações disponíveis (MARZULLO, 2009; THOMAS et al., 2003); e
- UDDI (*Universal Description, Discovery and Integration*) – é uma base de registros contendo a descrição dos *Web Services* na qual estes podem ser publicados e pesquisados (FARKAS; CHARAF, 2003; MARZULLO, 2009).

Arquitetura de Web Services

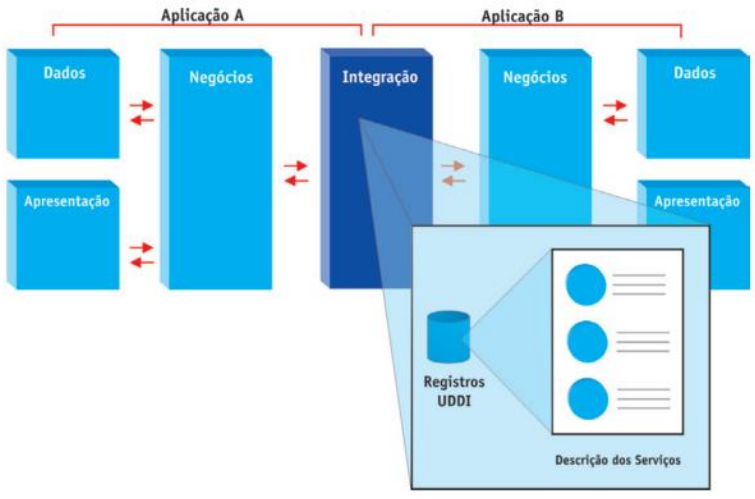
Em uma arquitetura SOA, a padronização é um ponto fundamental, pois, sem isso, manter e implementar a interoperabilidade entre sistemas pode ficar mais complexo, além de

dificultar a reutilização. Esses padrões estão presentes na arquitetura de *Web Services* conforme exposto na seção anterior (SOAP, WSDL e UDDI). Para que seja possível aplicar a utilização desses é necessário que exista pelo menos um servidor em rede para que os serviços possam ser acessados.

No entanto, para que esses serviços possam ser localizados é necessário que as informações referentes à localização do servidor bem como os parâmetros de entrada e saída estejam em um local no qual possam ser pesquisados. O padrão em *Web Services* é que esses dados sejam publicados no registro UDDI, mecanismo pelo qual descrições de *Web Services* podem ser localizadas por requisitantes em potencial.

De acordo com Braga (2008) o processo de descoberta, dependendo da necessidade, pode ocorrer em variadas situações: **(i)** desejo de estabelecer novas relações de negócio para transações online; **(ii)** um arquiteto que esteja projetando uma nova aplicação pode querer pesquisar a disponibilidade de lógica de programação genérica dentro da organização de forma que as descrições de serviço existentes possam ser descobertas e novas oportunidades de reuso implementadas; **(iii)** o projetista pode querer comercializar o *Web Service* para terceiros, fornecendo lógica de aplicação pré-construída com flexibilidade para ser incorporada (localmente ou remotamente) por outra aplicação; e **(iv)** um desenvolvedor que está construindo novos serviços precisará acessar as definições de interface para serviços existentes. O registro interno poupa ao desenvolvedor a preocupação de saber se a interface que está sendo incorporada é ou não a mais atual. A Figura 3.4 apresenta um exemplo de organização em um registro UDDI privado.

Figura 3.4 – Descrições de serviços em um registro UDDI

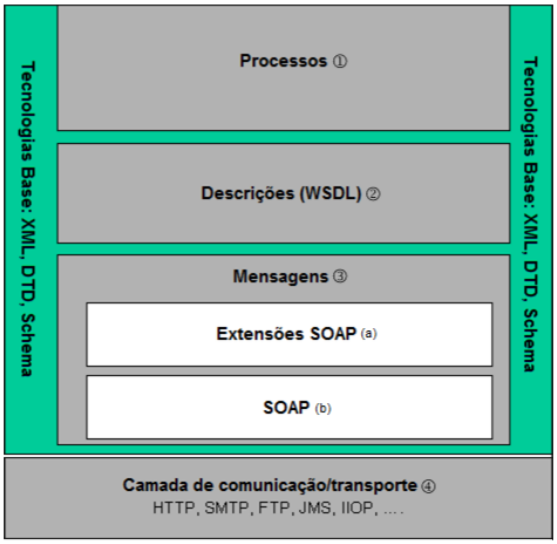


Fonte: BRAGA (2008, p. 42).

Para realizar a publicação do serviço é utilizada a WSDL, a qual é localizada como resultado da pesquisa efetuada por parte do agente consumidor. Uma vez que o consumidor tem acesso às informações da WSDL, no qual, estão os dados essenciais a respeito do serviço e do servidor, é possível executar a invocação de operações, para isso são enviadas mensagens encapsuladas por meio do protocolo SOAP.

A 3.5 apresenta a pilha da arquitetura de um *Web Service*, na qual são expostas as várias camadas que fazem composição desse modelo. A organização desta vai do nível de transporte até a camada de processo.

Figura 3.5 – Pilha da Arquitetura de um *Web Service*.



Fonte: Nakamura (2012).

Em resumo, a pilha ilustrada pela Figura 3.5, basicamente é formada pelas seguintes camadas:

1. Camada de Processos – Trata do gerenciamento e coordenação dos processos entre os serviços que compõe o *Web Service;*
2. Camada de Descrições – Responsável pela representação semântica formal das mensagens entendida pelos *Web Services.* É nela que estão descritas as restrições referentes às mensagens bem como a forma como os *Web Services* podem ser acessados. Essa camada também utiliza os arquivos WSDL que contêm a descrição dos serviços, formato das mensagens e também os tipos de dado e protocolo, entre outras (TAVARES, 2009);
3. Camada de Mensagens – Esta camada permite que a troca de mensagens seja realizada

em ambiente distribuído e descentralizado de forma interoperável. As mensagens neste nível podem ser serializadas em um arquivo XML seguindo as especificações do protocolo SOAP (NAKAMURA, 2012; TOYOHARA, 2009); e

4. Camada de Comunicação – Camada que especifica os protocolos para troca de mensagens entre clientes e provedores de serviço. Dentre os protocolos existentes, destacam-se o HTTP[23], SMTP[24], FTP[25], entre outros (NAKAMURA, 2012; TAVARES, 2009).

3.3 Computação em Nuvem

A computação em nuvem (*Cloud Computing*) é um conceito relacionado à forma como os serviços, aplicativos, capacidade de armazenamento, de impressão, de processamento e outros recursos de TI que não estão conectados diretamente ao computador são fornecidos. Nesse modelo, a disponibilização dos ativos computacionais é fornecida sob demanda pela Internet.

Como não se sabe exatamente a localização geográfica desses dispositivos, costuma-se dizer que estão "na nuvem" (*in cloud*). A definição do Amazon (2015) a respeito desse modelo é "por definição, diz respeito à entrega sob demanda de recursos de TI e aplicativos pela Internet, com modelo de definição de preço conforme a utilização" (AMAZON, 2015).

Outra definição mais disseminada foi provida pelo National Institute of Standards and Technology (NIST) que define a Comuputação em Nuvem como sendo " um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços" (MELL; GRANCE, 2011; JANSEN; GRANDE, 2011).

Essa abordagem possui algumas características essenciais:

- Autoatendimento sob demanda (***On-demand Self-service***) - O cliente contrata recursos computacionais de acordo com a necessidade e demanda sem intervenção humana. Por exemplo, caso exista um aumento de tráfego entre as requisições, o usuário poder adquirir mais recursos de forma automática (MELL; GRANCE, 2011; JANSEN; GRANDE, 2011; SOUSA et al., 2010);

[23] HTTP – um protocolo para transferência de texto na internet.
[24] SMTP – um protocolo para troca de e-mail.
[25] FTP – É um protocolo comum utilizado normalmente para realizar conexão e envio de arquivo para servidores.

- Rápida Elasticidade (***Rapid Elasticity***) - Essa característica refere-se à capacidade de incluir e remover recursos com elasticidade, ou seja, é possível ter uma escalabilidade sobre uma infraestrutura que parece ilimitada de forma instantânea (MELL; GRANCE, 2011; SOUSA et al., 2010);
- Pool de Recursos (***Resource Pooling***) - Os recursos ficam organizados em um pool para servir múltiplos usuários, de forma que há uma virtualização de armazenamento onde a localização física dos recursos fica transparente para o cliente (MELL; GRANCE, 2011; SOUSA et al., 2010);
- Amplo Acesso à Rede (***Ubiquitous Network Access***) - Os serviços disponibilizados na nuvem estão acessíveis de qualquer plataforma, sendo utilizadas em plataformas heterogêneas, por exemplo, acessíveis desde um computador pessoal ou um smartphone (MELL; GRANCE, 2011; SOUSA et al., 2010);
- Serviços Mensuráveis (***Measured Service***) - Essa característica é relativa à forma como os recursos adquiridos são controlados e monitorados, de modo que há uma transparência entre o cliente e o fornecedor, ou seja, os serviços adquiridos e a utilização em dado momento são monitorados e informados automaticamente (MELL; GRANCE, 2011; SOUSA et al., 2010). Além disso, a disponibilidade de um serviço em função de recursos computacionais é minimizada, uma vez que os recursos podem ser adquiridos para suprir alguma sobrecarga.

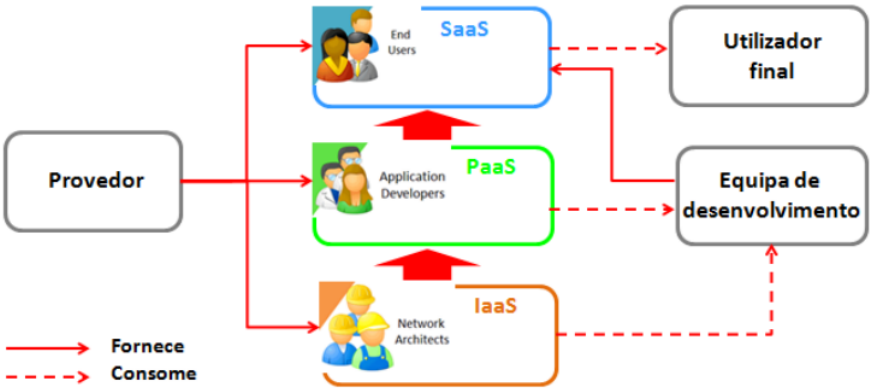
Ainda conforme SEI (2014), os modelos de serviços dessa abordagem podem ser classificados em:

- Software as a Service (SaaS) – Dentre os três modelos de serviço, o SaaS é o mais amplamente conhecido, visto que o foco principal são os usuários finais, de maneira a fornecer os serviços de uma aplicação sendo executada na infraestrutura nas nuvens (PIRES, 2013; MELL; GRANCE, 2011);
- *Platform as a Service* (PaaS) - Nesse modelo o provedor disponibiliza plataformas computacionais completas (incluindo, naturalmente, a infraestrutura computacional), ou seja, provê ao consumidor a capacidade de implantar aplicações criadas pelo próprio usuário (PIRES, 2013; MELL; GRANCE, 2011); e
- *Infrastructure as a Service* (IaaS) - nesse modelo o provedor oferece ao consumidor uma infraestrutura de processamento e armazenamento com a configuração que se adequa às suas necessidades, ou seja, o cliente não precisa adquirir uma máquina física

(PIRES, 2013; MELL; GRANCE, 2011).

A Figura 3.6 ilustra os papéis e relações envolvidos nesses modelos de computação em nuvem desde o seu conceito aos usuários finais. Assim, pode-se verificar que o *IaaS* fornece recursos de *hardware* e *software* ao *PaaS* e SaaS, já o *PaaS* oferece as ferramentas de desenvolvimento ao *SaaS* e por fim é possível observar que o *SaaS* é o responsável pela execução e disponibilização dos recursos computacionais aos usuários finais.

Figura 3.6 – Relações e papéis dos modelos de Computação em nuvem



Fonte: Pires (2013).

A utilização do conceito na implantação e desenvolvimento de sistema permite um acompanhamento e escalabilidade maior se comparado aos modelos tradicionais, no qual os serviços são utilizados em um servidor físico hospedado em um *data center* local, ou seja, nesse modelo o usuário pode realizar a utilização dos serviços de infraestrutura sem se preocupar com desgaste de equipamento ou necessidade de recursos computacionais, pois os mesmos, conforme elencado, podem ser adquiridos automaticamente de acordo com a demanda.

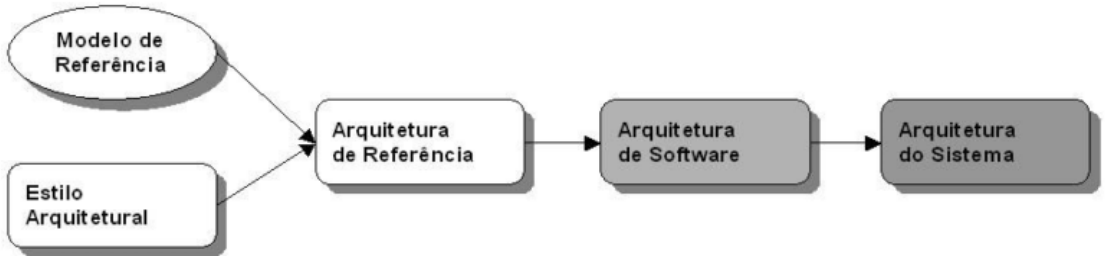
3.4 Arquitetura de Referência

Uma Arquitetura de Referência (**AR**) é um modelo que é utilizado para projetar e implementar arquiteturas de software. Uma **AR** é elaborada a partir de Modelos de Referência (**MR**) e Estilos Arquiteturais (**EA**). O planejamento desses artefatos pode ser facilitado em função da maturidade do domínio para o qual a arquitetura está sendo planejada. Ainda

conforme Sommerville (2011, uma **AR** é derivada de um estudo de domínio de aplicação e não de sistemas existentes podendo ser utilizadas para a implementação ou comparação de aplicações ou sistemas distintos.

De acordo com Bass et al. (2003) é comum existir a generalização de **AR**, **MR** e **EA** de modo que estes são confundidos com Arquiteturas de Software. Observe na Figura 3.7 que apesar de serem utilizados para definição de uma arquitetura de um sistema, nenhum deles pode ser considerada como Arquitetura de Software e sim partes que auxiliam na tomada de decisões importantes para elaboração e implementação desta.

Figura 3.7 – Relação entre Modelo de Referência, Estilo Arquitetural, Arquitetura de referência, Arquitetura de Software e Arquitetura do Sistema



Fonte: Bass et al. (2003).

Os artefatos que são utilizados para definição de uma **AR** são:

a) **Modelo de referência** – decomposição de problemas de forma padronizada e segmentada em partes conhecidas que cooperam para resolver um problema (BASS et al., 2003).
b) **Estilo arquitetural** – são esquemas de organização estruturada que estabelecem um padrão por meio de um conjunto de componentes, suas responsabilidades e a forma como se relacionam.

Enquanto **MR** adicionam soluções mapeando funcionalidades para um domínio específico, os estilos arquiteturais são utilizados para o projeto da **AR** onde é definido como será a implementação que irá apoiar o projeto da arquitetura de software.

Arquiteturas de referência são importantes para definição de sistemas com melhor qualidade, pois permitem que o projetista de software se preocupe com aspectos mais isolados e específicos de implementação, abstraindo a estrutura organizacional. O conjunto de técnicas descritas pode ser utilizado tanto para construção de sistemas simples como para a

implementação de softwares mais complexos compostos por subsistemas independentes que caracterizam as particularidades da aplicação diferenciando as arquiteturas umas das outras, no entanto, suas características comuns são compartilhadas pela **AR.** A arquitetura de cada módulo não é uma definição individual e sim a derivação realizada a partir de um modelo geral elaborado para todo sistema.

Assim, o reuso de arquiteturas de referência na modelagem de uma arquitetura de software permite que o arquiteto do software se preocupe com as características particulares do negócio da aplicação.

3.5 Trabalhos relacionados

Na literatura é comum encontrar trabalhos que pesquisem e tratem do reuso em aplicações (ALMEIDA et al., 2007; DONEGAN, 2007; SILVA, 2011; FUJIOKA, 2011; NAKAMURA, 2012; FUJIOKA, 2015), como forma de aumentar a produtividade, seja na simplificação do código ou na sistematização da geração de sistemas.

Apesar de existirem evidências comprovadas (ALMEIDA et al., 2007) e conforme exposto na Seção 3.1 de que as técnicas de reutilização trazem inúmeras vantagens para quem as utiliza, no domínio dos sistemas de RV, é escassa a investigação a respeito das técnicas de reuso. Para investigar a relação de utilização de reuso nesse domínio foi realizada uma revisão sistemática (RS) na qual seguiu-se o protocolo estabelecido (Apêndice A), baseado nas considerações de Kitchenhan et al. (2007) e Petticrew e Roberts (2006). A execução desse protocolo resultou em cento e quarenta e três artigos (143).

Desses, apenas em quinze (15) foram identificadas abordagens que compreendiam a utilização de técnicas de reuso conforme exibido no Quadro 3.5. Dentre esses, os trabalhos de Freiberger et al. (2014) e de Souza (2014) são os mais relevantes no contexto dessa Tese de doutorado. Na sequência os trabalhos são discutidos.

Quadro 3.5 – Artigos por abordagem de desenvolvimento

ARTIGOS	ABORDAGEM DE DESENVOLVIMENTO				
	FRAMEWORK	DBC	SOA	NUVEM	PADRÕES DE PROJETO
(BAUER ET AL., 2001)	X	X	X		

(OLIVEIRA ET AL.,2003)	X	X			
(REICHER ET AL., 2003); (ALLARD ET AL., 2004); (VERBRAECK E HOUTEN; 2005); (LLORA ET AL., 2008); (LI ET AL., 2010)		X			
(PAN ET AL., 2009)	X	X	X		
(POLYS ET AL., 2009)					X
(SHAO E MCGRAW, 2009); (LA E KIM, 2010)	X		X	X	
(OLIVEIRA E NUNES, 2010)	X				
(DIDIER ET AL., 2012)	X				
(FREIBERGER ET AL., 2014)			X	X	
SOUZA (2014)	X			X	

Fonte: Adaptado de Freiberger et al. (2012).

Bauer et al. (2001) apresentam o DWARF (*Distributed Wearable Augmented Reality Framework*), um *framework* baseado em componentes para sistema de RA composto por um *middleware*[26] responsável pela combinação de serviços de forma dinâmica, bem como a comunicação dos componentes.

Oliveira et al. (2003) apresentam um *framework* para ambientes virtuais que utiliza componentes distribuídos e dinâmicos e tem como seu principal artefato o JADE (*Java Adaptive Dynamic Environment*), esse, por sua vez, permite que em tempo de execução seja possível realizar o gerenciamento dos componentes e recursos do AV. Esse framework tem sua base fundada sobre um pequeno núcleo multiplataforma, sendo objetivo desse projeto ser flexível e reduzir o tempo para evolução e construção de AV. Para alcançar esses objetivos promove a diminuição da curva de aprendizado em uma arquitetura não monolítica.

A pesquisa de Reicher et al. (2003) compara plataformas de desenvolvimento de aplicações no domínio dos sistemas de RA. Tal comparação é realizada baseando-se em uma arquitetura de referência proposta pelo autor na qual são estabelecidos atributos fundamentais em sistemas de RA. Nesse mesmo trabalho também foi descrito um catálogo de atributos de qualidade esperados em sistemas para RA.

[26] *Middleware* – é um programa que fica entre pelo menos dois ou mais sistemas e atua como o intermediário na troca de mensagens entre partes do sistema.

O trabalho de Allard et al. (2004) apresenta um *middleware* intitulado de FlowVR que integra componentes distribuídos em aplicações de RV, os quais, por sua vez, são controlados pelo motor do FlowVR, que gerencia a produção e o consumo de dados. A troca de informações e dados são mediadas pelo *middleware*, uma vez que os componentes não se conhecem. De modo similar, Verbraeck e Houten (2005) apresentam uma plataforma construída em camadas e componentes genéricos para desenvolvimento de simulação de cadeias de suprimento. Nesse modelo, os componentes podem variar em diferentes simulações uma vez que o conhecimento não é incorporado aos componentes o que aumenta o potencial de reuso.

Apesar de não ser voltado especificamente para ambientes virtuais, o trabalho de Llora et al. (2008) apresenta o Meandre como proposta de uma arquitetura de criação, registro, publicação, localização e execução de componentes distribuídos que são baseados em serviços no qual a execução é orientada ao fluxo de dados. Essa arquitetura provê ferramentas com as quais é possível criar componentes e fluxos. Além disso, também disponibiliza um ambiente distribuído de execução orientado a serviços multinúcleo.

Já o trabalho de Pan et al. (2009) apresenta o SOHR (*Service Oriented* HLA RTI), um *framework* que opera em um ambiente distribuído de processamento utilizando o padrão HLA IEEE para distribuição de serviços de jogos multiusuários executado em grids computacionais. A interoperabilidade entre componentes distribuídos em rede pode ser apoiada pelo padrão HLA.

O trabalho de Polys et al. (2009) explora a utilização de padrões de projeto na visualização de cenas 3D, se embasando na ES no trabalho de GAMA et al. (1994). Com base nisso ele definem alguns padrões de projeto para visualização de cenas 3D com o X3D.

O *framework* SOESS (*Service-Oriented Embedded-Simulation Software*) é proposto por Shao e McGraw (2009) e tem como objetivo produzir aplicações de simulação militar interoperáveis entre sistemas e plataformas, principalmente com sistemas legados. Para isso, utiliza a composição de componentes. Oliveira e Nunes (2010) apresentam um *framework* orientado a objetos, o ViMet (*Virtual Medical Training*) capaz de produzir aplicações para simulação de exames de biópsia. Nesse projeto foi implementado o ViMeTWIzar, uma ferramenta capaz de simplificar a geração de aplicações e consequentemente reduzir os esforços de entendimento do *framework* e codificações de aplicações.

De acordo com Bari (2011) e Freiberger et al. (2012), a utilização de plataformas móveis, sensíveis ao contexto e ambientes virtuais estão bem próximas. Nessa linha, La e Kim (2010) apresentam um framework para construção de sistemas orientados a serviços e

sensíveis ao contexto. Além disso, tais sistemas são executados em ambientes na nuvem. Ainda, nessa arquitetura são propostos vários algoritmos relacionados com a adaptação de serviços em contextos diferentes para aplicações móveis.

Li et al. (2010) combinam em sua proposta o desenvolvimento baseado em componentes e um *middleware*. Essa combinação é utilizada na produção de sistemas de RV para subestações de transformação de energia elétrica. Esse trabalho destaca os pontos favoráveis em tal abordagem, principalmente no que se refere a aspectos como a redução de custos e aumento da qualidade, possíveis em função do reuso.

Já o trabalho de Didier (2012) apresenta o ARCS (*augmented reality component system*), um *framework* que permite integrar outros componentes de RA a um sistema. Para isso é preciso definir um nível de abstração para a integração que é realizada na ferramenta com a leitura de um arquivo XML sendo executado no nível de desktop.

A proposta de Freiberger et al. (2014) consiste numa plataforma orientada a serviço, na qual é possível tanto consumir como executar mundos virtuais. A plataforma é disponibilizada na nuvem e permite a produção de AV que podem ser reutilizados em outros sistemas que façam utilização da mesma plataforma. Nessa plataforma, a execução das operações da arquitetura é realizada utilizando *Web Services*. No entanto não é descrito a forma como isso pode ser realizado em função da multiplicidade de tipos de ambiente, além disso, o trabalho não apresenta resultados completos sobre a implementação da plataforma. Outro item que não fica claro é como outros sistemas vão poder consumir os componentes no caso de não utilizarem a arquitetura proposta pelo autor.

Em seu trabalho, Souza (2014) apresenta o *framework* AvComponent, com o qual é possível desenvolver componentes e compartilhá-los em uma infraestrutura em Nuvem. Ele utiliza uma abordagem na qual os objetos são persistidos em banco de dados instalados em ambientes de *cloud*, permitindo que os componentes sejam acessados de qualquer local conectado à Internet. Apesar de propor o compartilhamento entre ferramentas, não deixa claro como outras tecnologias podem ser utilizadas. O *framework* é fortemente dependente da tecnologia X3D.

Nesta seção foram destacadas abordagens que fazem utilização de técnicas de reuso, apesar de não deixarem de forma clara e explícita que estão utilizando uma técnica de reuso. As abordagens de reuso foram identificadas com leitura dos trabalhos e não expressas no texto como um dos objetivos, exceto, o trabalho de Freiberger et al. (2014) e Souza (2014).

Constata-se que abordagens tradicionais como *frameworks* estão sendo exploradas com mais frequência em relação a outras técnicas, entretanto, tendências como o

desenvolvimento para nuvem já começam a aparecer. Outro ponto relevante é que não foram encontradas abordagens que explorassem a disponibilização de AV com RV sem dependência de tecnologia de geração 3D. Além disso, ainda não existem números relevantes de pesquisa em RV e RA sobre o tema *reuso* ou que utilizem abordagens sistemáticas e efetivas de reuso.

3.6 Considerações

O desenvolvimento de aplicações exige cada vez mais conhecimento técnico dos programadores, se comparado com as tecnologias usadas nas décadas passadas. Essa tendência se dá em função da multiplicidade de ambientes, tecnologias e integrações que as aplicações precisam dar suporte, além da considerável mudança de requisitos por parte dos clientes, seja por ajuste ou alteração no foco dos negócios.

Portanto, é importante que o desenvolvimento de software seja cada vez mais ágil e eficiente. A reutilização de software auxilia no alcance deste objetivo. Entretanto, embora existam diversas técnicas de reuso de software (e elas venham sendo aplicadas e estudadas em diversas áreas e nos mais variados tipos de aplicação e domínios) as pesquisas que foram encontradas ou não relacionam diretamente técnicas de reuso ou possuem uma abordagem muito dependente de tecnologia que deixa obscuro a reutilização com outras tecnologias.

Outro conceito importante elencado foi o de SOA com *Web Services*, estrutura que permite a integração e comunicação entre aplicações utilizando padrões abertos, ou seja, independente da linguagem de programação ou plataforma, possibilitando a comunicação heterogênea entre sistemas, o que corrobora para potencializar a reutilização de funcionalidades e serviços em aplicações.

A unificação destes conceitos para geração de ambientes virtuais tridimensionais mostra-se, portanto, promissora. Tal conjunção vai permitir que componentes e sistemas legados também tenham capacidade de exercer o papel de consumidores de um *Web Service* na arquitetura definida neste trabalho. Isto será possível desde que estas aplicações sejam capazes de interagir com outros aplicativos utilizando *Web Services.* Por ser uma tecnologia aberta e utilizar padrões estabelecidos, é possível a comunicação de aplicações de diferentes implementações (Java[27], .NET[28], PHP[29], entre outras), inclusive proprietárias com os *Web*

[27] JAVA – linguagem de programação orientada a objetos multiplataforma.
[28] DOT.NET – linguagem de Programação OO proprietária para desenvolvimento de aplicações em arquitetura de sistemas da Microsoft.
[29] PHP – linguagem de script popular para o desenvolvimento de páginas dinâmicas para internet.

Services.

Conforme exposto no Quadro 3.5 da Seção 3.5, é possível identificar que não existe um número considerável de trabalhos a respeito da reutilização em RV e RA se comparado com outras áreas (ALMEIDA et al., 2007), bem como são reduzidas as pesquisas a respeito do tema ou que realizem abordagem efetiva a respeito. Ainda, apesar do reduzido número de investigações localizadas dentro do escopo e intervalo de tempo definidos, é verificada uma tendência nessa área em vista dos benefícios que podem ser alcançados com o reuso.

Além disso, os eventos em RV e RA estão abrindo espaços para discussão e investigação das práticas da ES, com intuito de verificar e adaptar tais técnicas em seus domínios. Um exemplo de evento é o SEARIS[30], um Workshop que vem sendo realizado desde 2008 em conjunto com o principal evento de RV da IEEE o "IEEE VR[31]". Neste evento são discutidos diversos tópicos relacionados à ES em RV tais como otimização, desenvolvimento, técnicas e algoritmos para visualização e renderização de AV e mais recentemente em 2014 foi aberto um debate sobre a utilização de arquiteturas orientadas a serviço para tecnologias virtuais[32].

Se a análise for realizada tomando-se como parâmetro o ano das publicações, percebe-se que existiu uma maior ocorrência nos anos recentes de 2009 a 2014, as pesquisas anteriores a 2009 em função das atualizações tecnológicas atuais podem ter ficado obsoletas, uma vez que não se encontra trabalhos recentes ou continuados das respectivas pesquisas. Além disso, as possibilidades de aplicações com o advento da computação em nuvem abriram novas oportunidades de investigação.

Dentre as ferramentas e propostas apresentadas duas possuem trabalhos relacionados (FREIBERGER et al., 2014; SOUZA, 2014). A plataforma de Freiberger et al. (2014) disponibiliza uma arquitetura na qual é possível, por meio de *Web Services*, executar e produzir mundos virtuais na nuvem. Ainda na mesma pesquisa os autores informam que é possível reutilizar esses mundos, no entanto, essa reutilização não fica clara, nem o grau de interoperabilidade, pois os mundos da plataforma são produzidos com tecnologia proprietária e dependente de plataforma: Java3D (ORACLE, 2014).

Já o *Framework* de Souza (2014) tem dependência do X3D (X3D, 2014) e foi implementado com base em outro sistema, o VEPersonal (AQUINO, 2007). Também não

---

[30] SEARIS - software engineering and architectures for realtime interactive systems.
[31] IEEVR – IEEE Virtual Reality. disponível em <http://ieeevr.org/>.
[32] **Invited talk: rest oriented architecture styles for virtual technologies:** tired or wired? Disponível em <http://www.searis.net/index.php5/workshop_2014>.

ficando claro como seria o reuso entre aplicações diferentes.

Assim, de acordo com o contextualizado existe uma oportunidade de investigação em relação à aplicação do reuso na disponibilização e gerenciamento de AV cujo repositório não seja direcionado a uma tecnologia específica. Além disso, as tendências atuais de programação em nuvem abrem oportunidades para integração dessas técnicas com SOA aumentando as possibilidades de escalabilidade proporcionadas pela computação em nuvem.

4 UMA ARQUITETURA DE REFERÊNCIA PARA GERENCIAMENTO E REUSO DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

Este capítulo apresenta uma Arquitetura de Referência baseada em SOA para gerenciamento e reuso de ambientes virtuais tridimensionais. Tal modelo propõe o desenvolvimento para reuso por meio do compartilhamento de AV tridimensionais não imersivos que utilizam RV (ver Seção 0.2). Além disso, tais ambientes devem ser recuperados por qualquer ferramenta que implemente *Web Services* por meio de um barramento ESB (ver Seção 0.1).

Com base na RS (Apêndice A) e também em pesquisa própria, chegou-se à conclusão de que existem oportunidades de pesquisa na construção de sistemas que auxiliem no desenvolvimento de AV com RV e RA. Na literatura foram encontrados apenas dois trabalhos, conforme exposto na Seção 3.5, porém com resultados não conclusivos, reforçando a hipótese de que o presente trabalho segue uma linha ainda pouco explorada e com oportunidades de investigação e verificação em aberto. As próximas seções descrevem o trabalho desenvolvido, tecnologias e estudos de caso que foram utilizados como prova de conceito para validar esta investigação.

4.1. Abordagem proposta

O termo Arquitetura de Software refere-se à estrutura de componentes do programa/sistema, suas inter-relações, princípios e diretrizes que conduzem sua concepção e progresso ao longo do tempo (GARLAN; PERRY, 1994). Conforme Bass et al. (2003), a definição de Arquitetura de Software está associada à forma como a(s) estrutura(s) do sistema estão organizadas, que componentes compõem o software, as interfaces destes componentes e suas respectivas relações. Em uma visão posterior, Barbosa (2009) descreve a Arquitetura de Software como a organização do sistema, por meio de seus componentes, considerando os princípios de design e evolução.

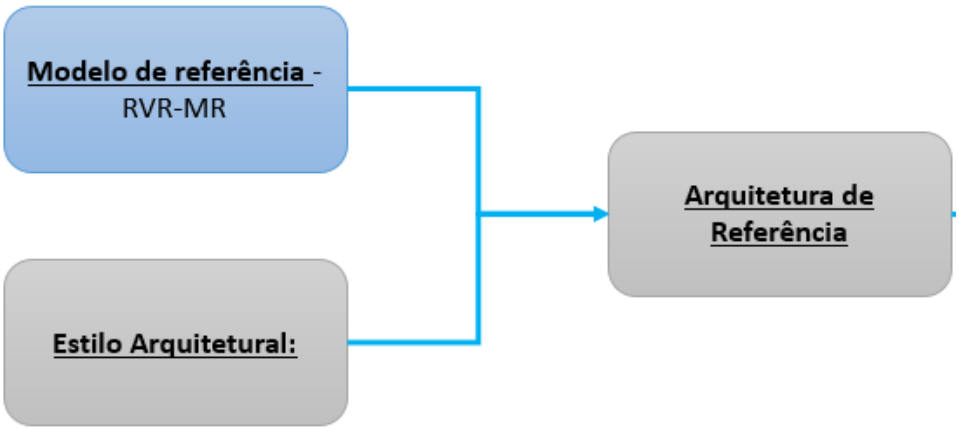
Assim, a abordagem utilizada nessa tese consiste na definição de uma arquitetura de referência que possa ser utilizada como modelo para geração de infraestruturas de gerenciamento e disponibilização de AV com SOA, implementada para *Web Services*. Com o intuito de validar esta estrutura, foi implementada uma infraestrutura básica seguindo tal modelo e uma aplicação cliente para consumir AV dessa infraestrutura através de um ESB. Neste sentido, esta Seção apresenta a Arquitetura de Referência e modelos sugeridos para tal

arquitetura e que foram utilizados como base para codificação dos experimentos.

4.1.1. Modelo de referência RVR

Conforme foi contextualizado na Seção 3.4, para definição de uma **AR** são necessários dois (2) elementos (**E**stilo **A**rquitetural e **M**odelo de **R**eferência). A Figura 4.1 exibe o relacionamento do **MR** na composição da **AR**.

Figura 4.1 – Relação do modelo de referência na composição de uma Arquitetura de Referência.



Fonte: Adaptado de Bass et al. (2003).

Na literatura, o **MR** é caracterizado como uma abstração da realidade, que pode ser expressa pelo formalismo de um método de modelagem, conforme os requisitos de um usuário (VERNADAT, 1996). De acordo com Bass et al. (2003), um **MR** é a divisão de funcionalidades, juntamente com o fluxo de dados entre os componentes. Um **MR** é caracterizado como um padrão de decomposição do problema. Os **MR** possuem características de domínios maduros, decorrente da experiência sobre este domínio. Conforme Mackenzie et al. (2006), no **MR** é proposto um vocabulário e um entendimento comum sobre o que são os elementos específicos do domínio trabalhado. Este contém uma normativa na forma de um modelo abstrato.

Assim, seja o seguinte **MR** para reuso proposto neste trabalho e exibido na Figura 4.2.

Para um Mundo Virtual (**MV**) tem-se as Tecnologias (**T**) utilizadas para construção de Objetos Virtuais (**OV)**, onde esses estão persistidos em um Repositório de Objetivos Virtuais (**ROV**) que são distribuídos a partir de um contexto de reutilização (**CR**), onde:

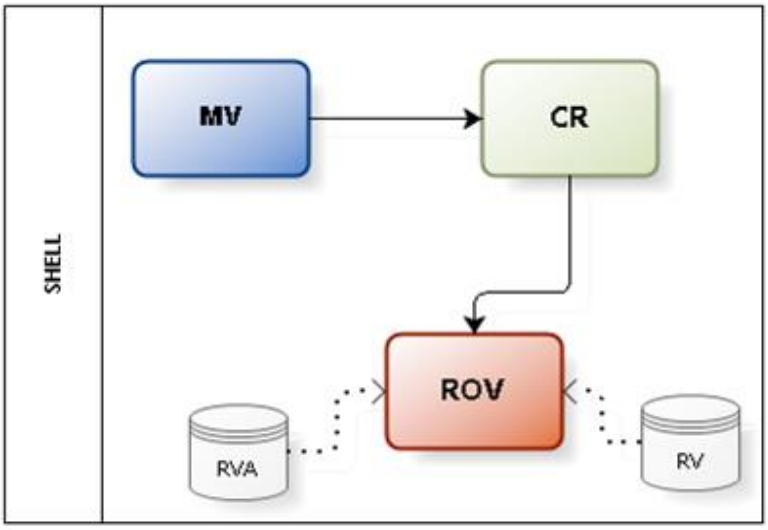
i. Para (**T**), tem-se $\boldsymbol{T = \{P_1, ..., P_n\}}$ conjunto das plataformas em que são executas as tecnologias (*Web*, *Desktop*, *Mobile*);

ii. Para um (**OV**), tem-se $\boldsymbol{OV = \{T_{i1}, \ldots, T_{in}\}}$ conjunto das tecnologias utilizadas para construção de **OV (**X3D, X3DOM, WebGL, entre outras);

iii. Para um (**ROV**), tem-se $\boldsymbol{ROV = \{OV_1, ..., OV_n\}}$ conjunto dos Objetos Virtuais persistidos no repositório; e

iv. Para um (**MV**), tem-se $\boldsymbol{MV = \{\exists\ OV_1, ..., OV_n\}}\ U\ \boldsymbol{CR\ \{P_{i1}, ..., P_{in}\}}$ conjunto dos Objetos Virtuais combinados para geração de um mundo em uma plataforma (**P**) que é acessado por meio de um contexto de reutilização (**CR**).

Figura 4.2 – Modelo de Referência (MR) RVR



Fonte: O autor.

Para o **MR** apresentado na Figura 4.2, seja **ROV** um repositório de **OV** tal que **CR** é o contexto de reutilização desses objetos e **MV** é um mundo virtual de interesse. Existe um

processo que agrega os demais elementos chamados (**Shell)** cuja finalidade é juntar ou colar as partes abstraindo detalhes tecnológicos para tornar possível a reutilização e objetos virtuais produzidos com tecnologias diferentes.

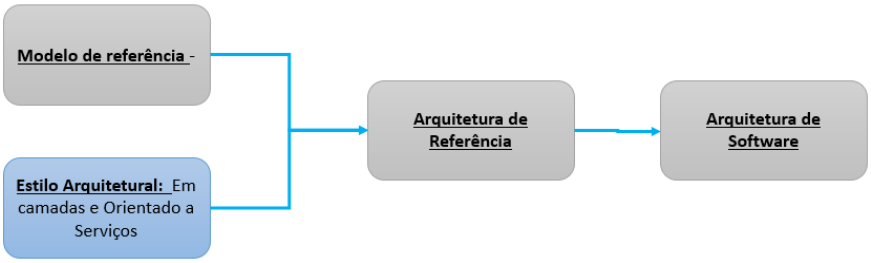
Tal **MR** foi projetado a partir da **RS** executada, na qual foi evidenciado o seguinte problema: **OV** estão sempre associados às tecnologias e os mundos gerados a partir desses têm restrições relacionadas às escolhas de linguagens de programação, plataformas operacionais, ambientes de autoria ou outro recurso de software utilizado em sua produção. O problema nos **MV** são as restrições $\sum_{1}^{n} (Oi, \tau i) \, / Oi \epsilon Re \top i \epsilon \top$ para se fazer reutilização de objetos em um novo contexto no qual tecnologias distintas possam ser usadas.

Para abstrair esse problema, o **MR** apresentado define um protocolo genérico para ser utilizado como base para definição de uma arquitetura de referência para **RV** dentro de um **CR** de **OV.**

4.1.2. Estilo Arquitetural

De acordo com o que foi descrito na Seção 3.4, o Estilo Arquitetural é um dos componentes necessários para definição de uma Arquitetura de Referência. Dessa forma, esta seção descreve os dois estilos que foram utilizados para tal. A Figura 4.3 apresenta a relação do **EA** na composição da **AR**.

Figura 4.3 – Relação do estilo arquitetural na composição da arquitetura.



Fonte: O autor.

De acordo com Murakami (2006), alguns estilos arquiteturais são frequentemente utilizados como soluções para todas as formas de sistemas. Entretanto, um bom projetista deveria selecionar um estilo que atenda às necessidades de um problema particular a ser resolvido, ou seja, os estilos arquiteturais escolhidos para projeto de sistemas devem atender as necessidades desse sistema ou domínio de aplicações. Existem **EA** descritos na literatura (SUMMERVILLE, 2011; ALMEIDA et al., 2007), sendo que os mais aderentes ao modelo proposto nesse trabalho são os estilos baseados em SOA (SOA, 2015) e Camadas (SUMMERVILLE, 2011).

A utilização de **SOA** como um dos padrões escolhidos permite a construção de soluções para Internet de forma que a interoperabilidade entre elas é mantida. Esse padrão permite que aplicações desenvolvidas em plataformas (**P**) diferentes com tecnologias (**T**) distintas possam trocar informações utilizando uma linguagem de comunicação padrão. Ainda conforme Gonzaga (2014), o desenvolvimento de aplicações baseadas no paradigma SOC (*Service Oriented Computing*) permite que serviços sejam integrados e disponibilizados por provedores distribuídos na Internet. Durante o processo de execução é possível que diversos sistemas disponibilizados em diferentes provedores sejam integrados. As etapas podem ser providas por provedores internos ou externos (SANTANA et al., 2009).

Já a utilização do padrão **Camadas** apresenta uma solução que minimiza o acoplamento entre as partes do sistema. Nesse estilo, os sistemas são separados em camadas que se comunicam entre si trocando informações. Um sistema organizado em camadas tem sua organização estruturada hierarquicamente, onde cada camada fornece serviços para a camada superior e consome os serviços da camada inferior (GARLAN; SHAW, 1993). A organização do sistema em camadas permite o acoplamento ou substituição de camadas de uma forma que o impacto das mudanças é menor.

Esses dois estilos combinados são aderentes ao **Objetivo Geral (OG) I** (Definir uma

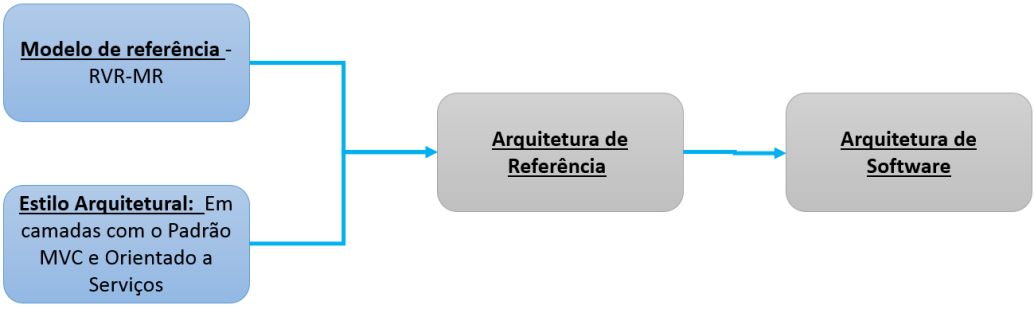
Arquitetura de Referência que possa ser aplicada para RV com Reuso de Software e SOA independentemente da plataforma utilizada) e consequentemente colaboram para a obtenção das respostas das perguntas elencadas no **OG** II (**a -** Como explorar o reuso de componentes virtuais em ambientes tridimensionais, utilizando o conceito de reuso mantendo a interoperabilidade entre sistemas?; **b -** É viável utilizar técnicas de reutilização dentro de uma infraestrutura de reuso de componentes virtuais que funcione de forma distribuída utilizando uma Arquitetura Orientada a Serviços?;e **c -** É factível que a interoperabilidade entre sistemas de RV seja alcançada com disponibilização destes componentes utilizando Web Services?)

4.1.3. Arquitetura de referência

Com definições abstratas e genéricas formando um modelo de referência para reuso de objetos virtuais tridimensionais, pode-se agora reunir o modelo, juntamente com o estilo arquitetural escolhido, SOA e Camadas, para definição da Arquitetura de Referência para reuso em RV.

Para Definição da AR deste trabalho foram utilizados o **MR** definido na Seção 4.1.1 e o **EA** descrito na Seção 4.1.2. A Figura 4.4 exibe o relacionamento entre elas para geração da **AR** ilustrada na Figura 4.5.

Figura 4.4 – Relação entre o Modelo de referência e o Estilo Arquitetural para definição da Arquitetura de referência.



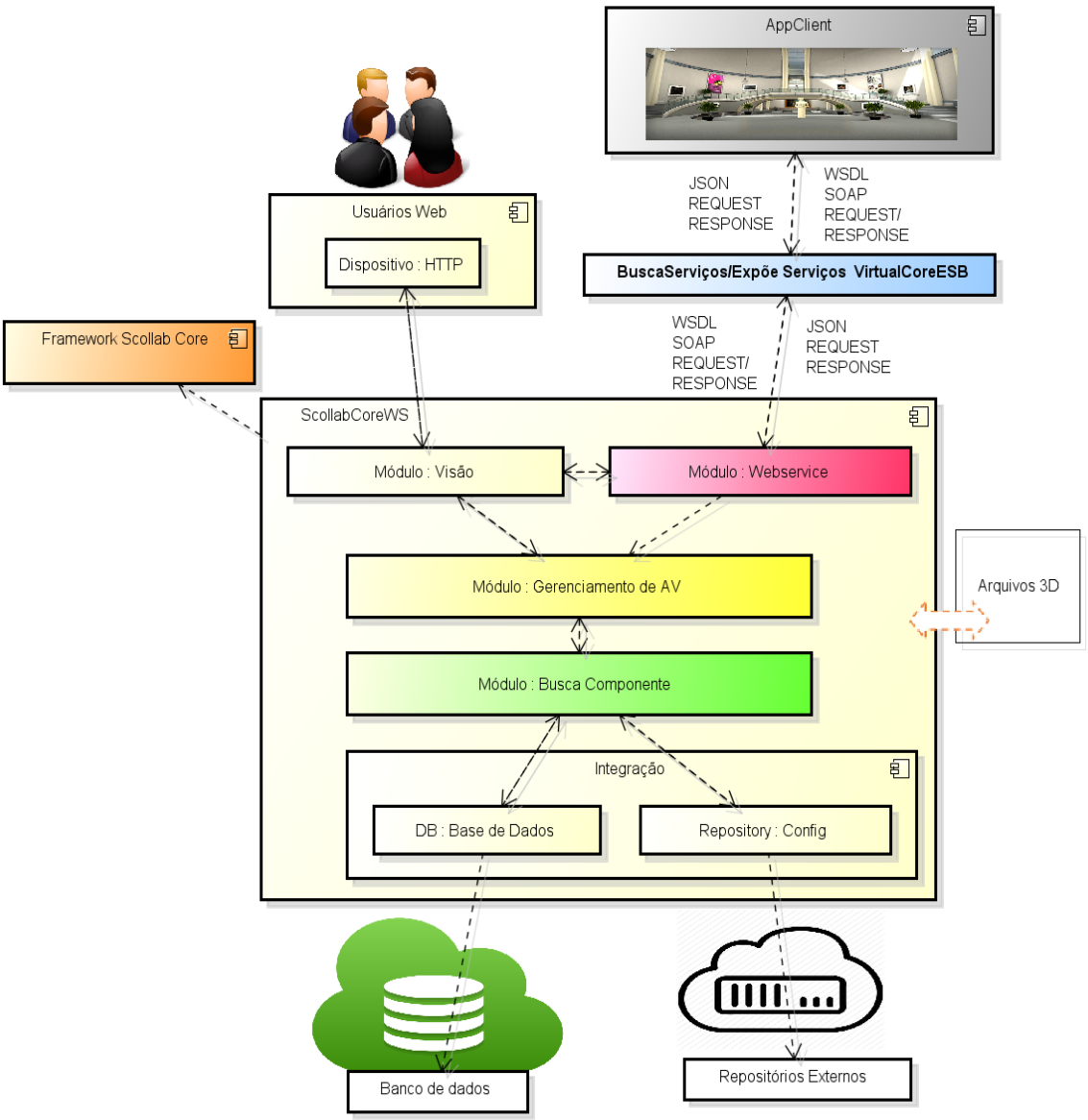
Fonte: O Autor.

Em tal **AR** os **OV** podem ser gerenciados e também acessados para composição de novos **MV** a partir de um **CR** que não seja dependente de tecnologia (**T**), nesse caso, a utilização de SOA com *Web Services*. Conforme contextualizado, *Web Services* são independentes de tecnologias e podem ser implementados na maioria das linguagens de

programação atuais o que garante a interoperabilidade do **CR** em relação ao acesso dos **ROV** da arquitetura.

Na **AR** exibida pela Figura 4.5 pode-se observar uma estrutura modular em que as funcionalidades são expostas para aplicações externas por meio de uma camada de serviços descrita na literatura como ESB[33] (*Enterprise Service Bus*). Por outro lado, oferece uma camada com interface web amigável para que usuários possam gerenciar AV e componentes tridimensionais.

Figura 4.5 – Arquitetura de referência

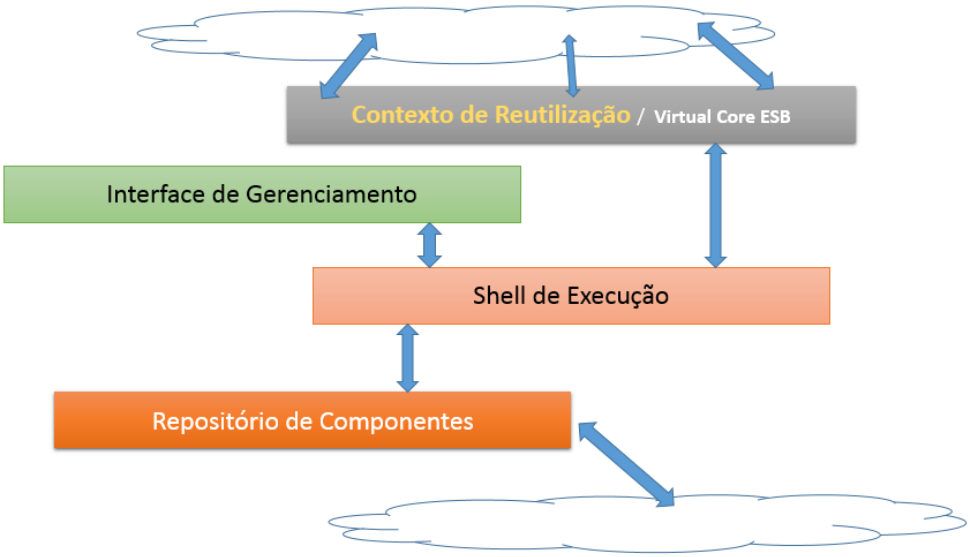


[33] ESB – *Enterprise Service Bus* "fornece um ambiente de hospedagem para *Web Services*, permitindo a conexão e a exposição de serviços sobre protocolos de transporte baseados em padrões da internet, como HTTP e FTP" (PASLEY, 2005; MURAKAMI, 2006).

Fonte: O Autor.

Os componentes da arquitetura apresentada na Figura 4.5 podem ser categorizados em quatro camadas genéricas conforme exibido na Figura 4.6.

Figura 4.6 – Organização estrutural da arquitetura de referência.



Fonte: O autor.

- **Interface de gerenciamento** – Conjunto de componentes que auxiliam graficamente os usuários responsáveis pela persistência e administração dos **OV** de um repositório;
- ***Shell* de Execução** – Conjunto de componentes que tratam das operações de baixo nível e de infraestrutura da arquitetura, questões de persistência e tratamento de erros bem como o registro de logs de atividades realizadas e o controle de acesso baseado em papéis. Para o desenvolvimento do *shell* o pesquisador pode optar por qualquer *framework* que abstraia esses tipos de operação. Essa camada é a responsável pela orquestração dos subsistemas internos da **AR**;
- **Repositório de componentes** – Conjunto de artefatos que garante a recuperação e busca de componentes com base nos metadados associados ao **OV**. Define os contratos que devem ser implementados para interação com os repositórios; e
- **Contexto de reutilização** – Esse contexto é representado pela camada de serviços (**VirtualCoreESB**) por meio da qual são expostas as operações disponíveis para busca e recuperação dos componentes da arquitetura.

Cada um dos componentes exibido na Figura 4.5 agrega responsabilidades a **AR**, sendo responsáveis pela execução de fluxos e também pela forma como a **AR** executa suas operações. Cada uma dessas partes tem seu propósito descrito segundo as suas obrigações e

relações dentro da arquitetura:

- **Usuários Web e Módulo Visão** – Nessas camadas estão os dispositivos (Navegadores Web e Dispositivos Móveis) que são utilizados para acessar os módulos de gestão da arquitetura proposta e funcionalidades atribuídas aos usuários dos repositórios. Por meio desses dispositivos os usuários consomem e interagem com os **OV** persistidos no **ROV**;
- **Arquivos 3D –** Representa o local em que os arquivos físicos dos componentes serão salvos. Observe que essa é uma das opções disponíveis, uma vez que os autores podem optar por salvar o arquivo em formato binário na própria base de dados;
- **Framework ScollabCore** – Esse item representa o **ScollabCore**, ou seja, toda a parte de controle, segurança, exceções, utilidades entre outras características de infraestrutura são implementadas neste *Framework* ;
- **ScollabCoreWS** – Os componentes desse módulo representam os itens que foram estendidos ou devem ser implementados na arquitetura adaptada a esta tese;
- **VirtualCoreESB** - Esta camada é a responsável pela exposição dos serviços da arquitetura para aplicações externas;
- **Módulo *Web Service*** - Este módulo trata da implementação e gerenciamento dos serviços internos. Suas interfaces são expostas na camada **VirtualCoreESB**. Essa camada pode enviar e receber mensagens com o protocolo SOAP utilizado na comunicação entre *Web Services.* É por meio deste módulo que as funcionalidades dos repositórios podem ser acessadas externamente;
- **Módulo Busca Componente** - Responsável pelas implementações que tratam as operações de busca e recuperação de metadados e dos componentes **OV** persistidos nos **ROV** internos e externos;
- **Módulo Gerenciamento de AV** - Realiza as operações relacionadas à validação dos dados em componentes, como por exemplo, não permitir que um artefato seja enviado para persistência sem as devidas descrições. É também a partir desta camada que saem as requisições para a invocação de operações de persistência junto à camada de integração, além de tratar de informações a respeito dos ambientes persistidos;
- **Camada de Integração** – Essa é a camada responsável pela integração das aplicações com o **ROV**, importante para a interoperabilidade de repositórios tanto locais como

externos. É nessa camada que estão as classes *Data Acess Object* (DAO[34]) que contêm os principais métodos utilizados em sistemas CRUD (*Create, Retrieve, Update e Delete*)[35] de modo que essa camada gerencia as operações de integração com as bases de dados. É aqui também que são informados os meta-dados a respeito dos repositórios que devem ser pesquisados para recuperação de componentes externos por meio do **Repository:Config**; e

- **Repositórios Externos e Banco de Dados** – Estes componentes externos representam como e onde os componentes podem estar persistidos. O Item "**Repositórios Externos**" utiliza as regras de configuração do **Repository Config**. A partir desse componente é realizada a invocação a repositórios externos.

4.1.4. Arquitetura

O ScollabCore é um arcabouço de software pré-montado para o desenvolvimento de aplicações JEE fornecendo infraestrutura para criação de novas aplicações a partir do modelo montado no próprio *framework*. Sua arquitetura foi baseada no padrão MVC[36] (FOWLER, 2002) e também utiliza outros *frameworks* de infraestrutura (BARRETO, 2006; FAYAD et al. 1999a, b; FAYAD; JOHNSON, 2000; FAYAD; SCHMIDT, 1997).

Esse *framework* proporciona soluções para lidar com as questões de baixo nível como persistências de dados, controle de transações, segurança, entre outras. No modelo apresentado pela Figura 4.6 esse *framework* representa o *Shell* de execução.

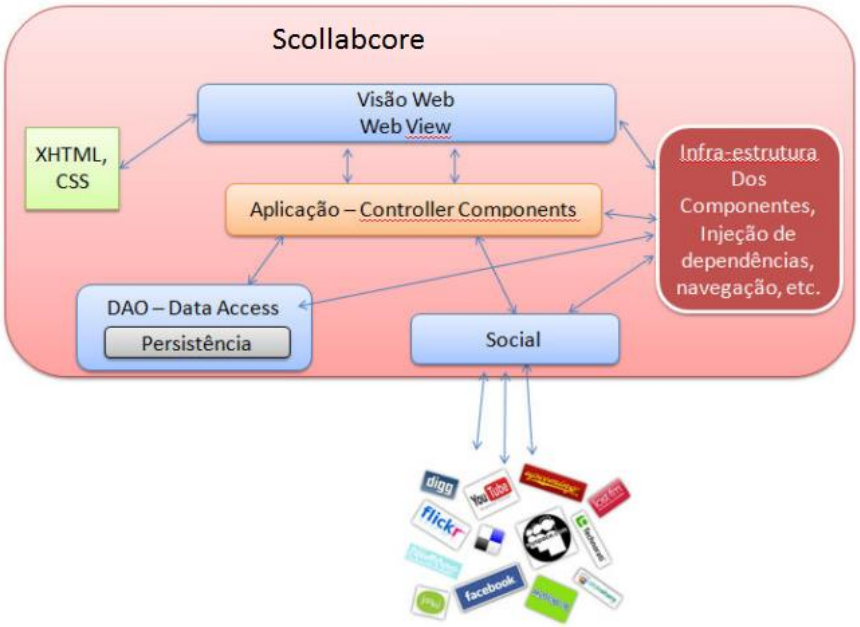
Arquitetura do ScollabCore

Essa seção apresenta a arquitetura do ScollabCore e seus módulos, os quais são utilizados no desenvolvimento dos estudos de caso. A Figura 4.7 apresenta a sua estrutura e seu funcionamento interno, já a Figura 4.8 exibe os componentes que fazem parte do núcleo da ferramenta. Na sequência, os mesmos são detalhados, sendo excluído dessa descrição o Módulo Social, pois não foi utilizado em nenhuma funcionalidade desenvolvida neste trabalho, tornando-se irrelevante para esta pesquisa.

---

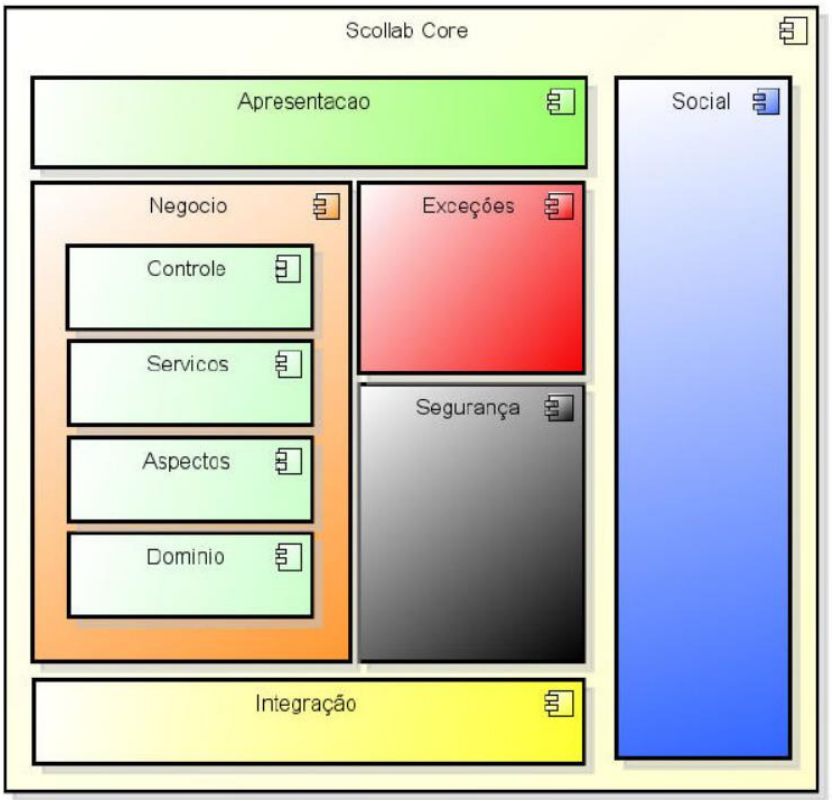
[34] DAO – padrão de projeto arquitetural que trata da camada de dados de uma aplicação, normalmente realizando operações de persistência no banco de dados (FUJIOKA, 2011).
[35] CRUD – operações comuns em aplicações. Consiste nas operações de inclusão, remoção, consulta e listagem de informações de um sistema de banco de dados.
[36] MVC – padrão de projeto que proporciona a separação entre a lógica de negócio e a lógica de apresentação. Esta separação permite que o desenvolvimento, teste e manutenção sejam aplicados de forma isolada em ambas as camadas (FOWLER, 2002).

Figura 4.7 – Arquitetura do ScollabCore



Fonte: Fujioka (2011).

Figura 4.8 – Módulos que integram o ScollabCore



Fonte: Fujioka (2011).

Além da descrição dos módulos, algumas tecnologias que foram utilizadas para implementação são referenciadas no trabalho e descritas no Apêndice B. Os objetivos dos

principais módulos dessa arquitetura são:

- **Apresentação –** este módulo faz utilização de outro *framework*, que é parte da arquitetura do ScollabCore, o *Java Server Faces* (JSF). Adicionalmente a ele foi utilizada na implementação do cliente web a biblioteca Primefaces[37] para a camada de apresentação (ver Apêndice B);
- **Aspectos –** O módulo de Aspectos do ScollabCore possui configurações que permitem a invocação de métodos de forma declarativa. A Figura B.1.1 (Apêndice B) apresenta trecho do arquivo de configuração utilizado para realizar este procedimento. A linha sublinhada no arquivo, por exemplo, está informando ao *framework* para que todas as classes dentro do pacote **br.ufpb.di.labes.focos** que tenham seus nomes iniciados com a palavra "*Service*" devem ter suas operações registradas pelo método log. No desenvolvimento desta tese foram adicionados os pacotes de classe dos sistemas desenvolvidos, tanto o cliente como o repositório;
- **Controle –** Contém a responsabilidade de executar as operações de interação com as interfaces web nas aplicações A Figura B.1.2 (Apêndice B) ilustra de forma detalhada as classes desse módulo;
- **Exceções –** Módulo que trata das exceções do sistema. Erros lançados devem ser tratados com respostas de forma que o sistema continue em funcionamento. A Figura B.1.3 exibe as classes e UML desse módulo;
- **Segurança e Autenticação –** Trata da segurança. É utilizado nesta tese para limitar os usuários de acordo com os perfis que os mesmos tenham. Limitando as operações de cada operador em função disto. A Figura B.1.4 apresenta a modelagem das classes base; e
- **Integração**: Este módulo trata da persistência das entidades do sistema. É nele que estão configurados os DAO Genéricos responsáveis pelas operações mais comuns em aplicações, os CRUD (*Create, Retrieve, Update e Delete*). Esse módulo foi estendido no projeto desta tese, incluindo um módulo para configuração de repositórios externos. Mais detalhes das classes base estendidas podem ser observados na Figura B.1.5.

A próxima seção elenca o que foi realizado para estender esse *framewok* de forma que ele pudesse ser utilizado no desenvolvimento para prova de conceito dos experimentos deste

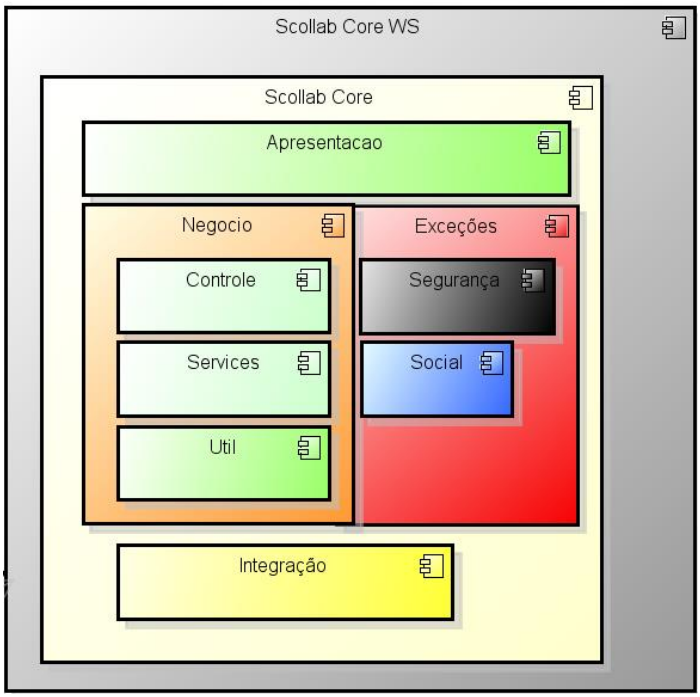
[37] **Primefaces** – conjunto de componentes para JSF. Disponível em <www.primefces.org> acesso em 12 fev. 2015.

trabalho.

4.1.5. ScollabCoreWS

Para incluir as operações necessárias para realização deste projeto foi necessário incluir um módulo para tratamento de *Web Services*. A Figura 4.9 exibe como ficou a composição da arquitetura após tal ajuste.

Figura 4.9 – Arquitetura do ScollabCoreWS



Fonte: O autor

Para tal foi utilizado o *springframework*[38] (SPRING, 2015), com o qual foi simplificada a complexidade referente à configuração do *Web Service* e geração do WSDL da aplicação. Na Figura 4.10 são ilustradas as linhas que foram incluídas para adicionar esse suporte. Nesse caso foi necessário parametrizar o caminho de invocação dos serviços, ilustrado na imagem pela *tag* **<*binding*>** a qual recebe a URL de requisição. Já a *tag* **<*service*>** informa o nome da classe que vai ser invocada e terá suas funções expostas no WSDL para consumo externo.

[38] SPRINGFRAMEWORK – O Spring é um *framework* não intrusivo baseado em padrões de projeto, inversão de controle (ioc) e Injeção de Dependência (DI) ver (apêndice B). Disponível em: <http://projects.spring.io/spring-framework/>. Acesso em: fev. 2015.

Figura 4.10 – Configuração dos *Web Services* para consumo dos mundos

```
<import resource="spring-bean.xml"/>
    <wss:binding url="http://digitaloceam.rodrigofujioka.com/scollabcore:8080/webservice/consomeAV">
      <wss:service>
          <ws:service bean="#AVWS"/>
      </wss:service>
  </wss:binding>
```

Fonte: O autor

Os trechos de código que representam a implementação da chamada aos serviços são exibidos na Figura 4.11 , foram adicionadas três operações básicas necessárias para interação do repositório com cliente descrito na Seção 4.5.

Figura 4.11 – Trecho de código que é invocado pelos *Web Services*

```
@WebService
@SOAPBinding(style = Style.RPC)
public class AVWS {
    private AvDAO avDAO
    @WebMethod
    public AmbienteVirtual ConsomeComponentes(
            @WebParam(name = "idComponente") String idComponente,
            @WebParam(name = "idGrupo") String idGrupo) {
            ...
        return av;
    }

    @WebMethod
    public AmbienteVirtual[] buscaCompoenentes(
            @WebParam(name = "parametros") String[] parametros) {
            ...
        return listaAVParametros;
    }
    @WebMethod
    public AmbienteVirtual[] listaComponentes() {
            ...
        return listaAV;
    }
}
```

Fonte: O Autor

Ainda referente ao código exposto, ressalta-se que, o programador, caso tenha a necessidade, poderá incluir novas funcionalidades, bastando para isso preceder o método pela

anotação *@WebMethod*[39], essa marcação faz com que o Spring disponibilize a operação no serviço. Já o serviço é definido pela anotação *@WebService*[40], responsável por indicar ao compilador que o arquivo corresponde à definição do SEI de um serviço Web. Observe também a anotação *@SOAPBinding*[41] indicando que o serviço utilizará a abordagem SOAP e não REST[42], que é a outra abordagem que tem suporte.

Essa seção apresentou a descrição e alguns aspectos essenciais da extensão aplicada no ScollabCore em relação ao tratamento e geração de *webservices*. Também foi informado que novas funcionalidades podem ser adicionadas para tratar questões de requisitos, no entanto, algumas simples customizações devem ser realizadas conforme exposto.

4.1.6. Modelo de dados

Para desenvolvimento dos repositórios é disponibilizado o modelo de dados genérico exibido na Figura 4.12. Trata-se de uma referência para implementação, ou seja, a sua utilização fica a critério do pesquisador que pode estender ou reduzir tal modelo conforme o projeto em desenvolvimento.

---

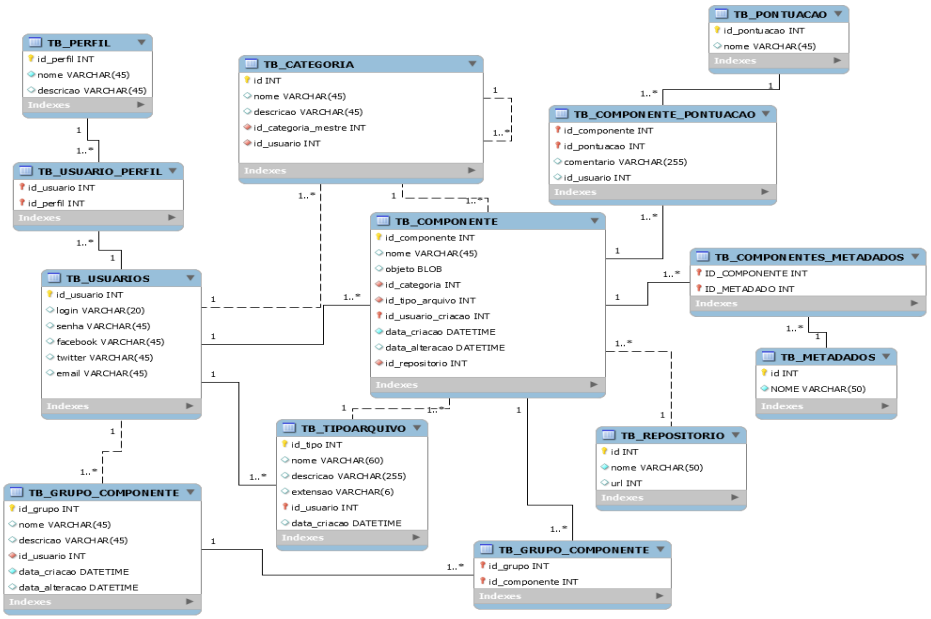
[39] @Webmethod – define uma operação de um webservice. Disponível em: <http://examples.javacodegeeks.com/enterprise-java/jws/jax-ws-spring-integration-example/>. Acesso em: fev. de 2015.

[40] @Webservice – anotação que indica que a classe JAVA é um serviço da web. Disponível em: <http://examples.javacodegeeks.com/enterprise-java/jws/jax-ws-spring-integration-example/>. Acesso em fev. de 2015.

[41] @SoapBinding – informa o tipo de abordagem que será utilizado. Disponível em: <http://examples.javacodegeeks.com/enterprise-java/jws/jax-ws-spring-integration-example>. Acesso em: fev. de 2015

[42] REST – outra abordagem para troca de mensagens entre serviços meta-dados.

Figura 4.12 – Modelo de dados da Arquitetura de referência



Fonte: O autor.

Na sequência são descritos o propósito e a utilidade de cada tabela do modelo sugerido. As tabelas **TB_PERFIL**, **TB_USUARIOS** e **TB_USUARIO_PERFIL** são utilizadas para inclusão e controle das permissões e acessos de cada usuário ao sistema. Um usuário pode assumir diversos perfis dentro da arquitetura. Os perfis sugeridos e descritos nesse trabalho são:

- **ROLE_ADMIN** – Usuário que realiza operações de manutenção e tem total acesso às configurações da arquitetura;
- **ROLE_AUTOR** – Esse perfil indica que o operador é um produtor de ambientes, ou seja, ele disponibiliza objetos e ambientes tridimensionais em grupos que podem ser públicos ou privados; e
- **ROLE_CONSUMIDOR** – Essa regra define que o usuário apenas realiza consumo de objetos da infraestrutura e também pode solicitar acesso a grupos de objetos. Os dados desse usuário são utilizados também pela aplicação cliente.

Dentre as tabelas existentes a **TB_REPOSITORIO** é preenchida no momento da inclusão do componente, informando o nome do repositório e o nome do artefato do qual faz parte um determinado componente.

A tabela **TB_COMPONENTE** é utilizada para persistir os componentes em 3D que

podem variar de simples objetos 3D a um AV conforme definido na Seção 4.1.1, artefatos são persistidos em um campo do tipo BLOB[43] ou salvos localmente na estrutura de arquivos da arquitetura.

As tabelas **TB_TIPOARQUIVO** e **TB_CATEGORIA** são utilizadas pela **TB_COMPONENTE** na classificação dos objetos e também na busca e recuperação de componentes. O tipo de arquivo (**X3DOM, X3D, VRML** ou **WebGL**) define sua extensão, já a **TB_CATEGORIA** informa qual o domínio de aplicações que melhor representa o componente persistido (**Simulação, Exploração, Educação,** entre outras).

Os objetos 3D são agrupados na **TB_GRUPO** e **TB_COMPONENTE_GRUPO.** Nesse conjunto de tabelas é possível que o usuário com a regra ROLE_AUTOR crie, por exemplo, um grupo descrito como ***Museu*** e associe a esse grupo um conjunto de componentes relativos a uma categoria **Exploração** por exemplo. Dessa forma a localização de experimentos exploratórios ligados ao grupo ***Museu*** torna-se mais simples.

Cada componente deve ser vinculado a uma ou mais palavras-chave. Nesse caso a associação é realizada no momento do cadastro, onde o usuário digita as informações de forma que os dados sejam persistidos nas tabelas **TB_METADADOS** e **TB_COMPONENTE_METADADOS.** Esse conjunto de tabelas é utilizado pelo módulo de busca da arquitetura. Tais palavras auxiliam no processo de localização de AV pelo cliente.

As demais tabelas da arquitetura são utilizadas para classificação de qualidade, nas quais os consumidores que utilizam um objeto do repositório podem atribuir uma nota ao componente em termos de: “Qualidade gráfica”, “Coesão”, “Navegabilidade”.

A utilização desse modelo segue como sugestão auxiliando no desenvolvimento de novas aplicações baseadas na arquitetura proposta.

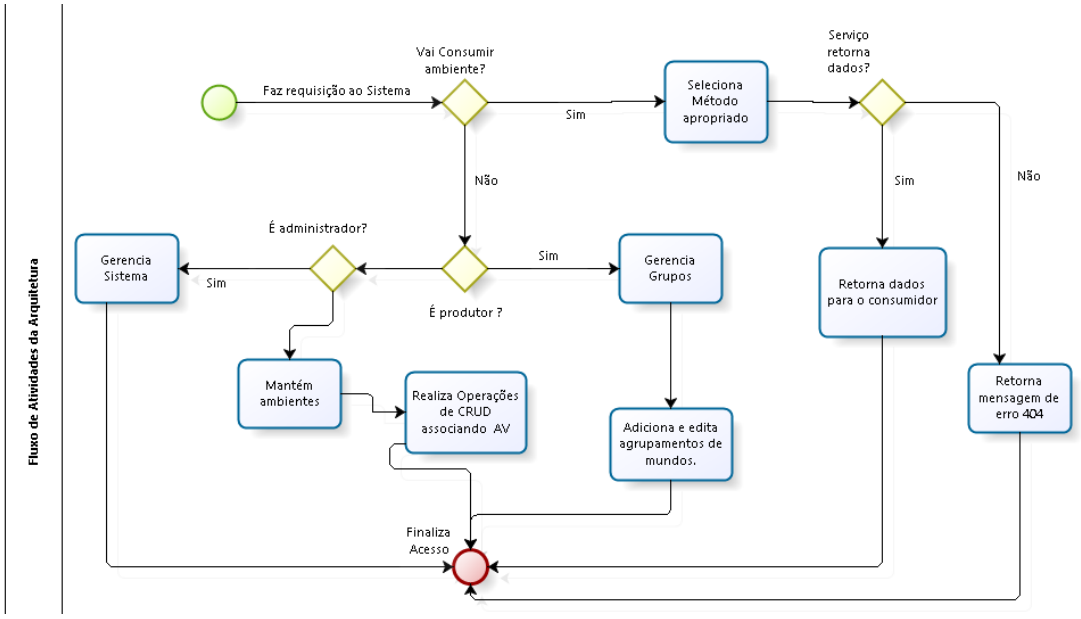
4.1.7. Diagrama de atividades

O diagrama de atividades da Figura 4.13 apresenta o fluxo de navegação genérico proposto para a arquitetura desse trabalho. Esse fluxo leva em consideração os perfis sugeridos inicialmente nesse trabalho (ROLE_ADMINISTRADOR, ROLE_AUTOR e ROLE_CONSUMIDOR).

---

[43] BLOB – é um tipo de dado do SGBD que permite a inclusão de objetos grandes como imagens, sons entre outros. Mais informações em: <http://support.microsoft.com/kb/103257/pt>. Acesso em: fev. 2015.

Figura 4.13 – Fluxo de atividades para um sistema genérico independente de domínio para gerenciamento de ambientes virtuais



Fonte: O autor.

A sequência de operações ilustrada na Figura 4.13 é:

1. Fluxo é iniciado com o envio de requisição para o sistema. Aí são apresentados dois fluxos para recuperação ou gerenciamento dos AV:
   a) Acessa interface gráfica web para Gerenciar AV ou componente 3D da arquitetura e desvia fluxo para o item 3; ou
   b) Acessa *Web Services* para consumo de AV 3D, caso no qual o fluxo é direcionado para o item 2.
2. O cliente poderia acionar um método dentre os disponíveis no *Web Service* (*lista_componentes*, *consome_componentes ou busca_componentes*) que retorna um conjunto de meta-dados contendo a resposta solicitada ou uma mensagem que pode vir nos formatos XML ou JSON[44] indicando erro 404 e desvia o fluxo para o item 7;
3. Caso o cliente tenha escolhido a opção de não consumir componentes:

a) O sistema verifica as permissões do usuário:

i. Se **Role_Autor**: O sistema altera o fluxo para o item 4;

ii. Se **Role_consumidor**: O sistema altera o fluxo para o item 5**;** e

iii. Se **Role_administrador**: O sistema altera o fluxo para o item 6.

[44] JSON - Javascript Object Notation – Formato de dados utilizado para troca de dados entre aplicações. Disponível em: <json.org>. Acesso em: 17 mai. 2015.

4. Este fluxo apresenta interface web com as opções de autoria e manutenção de AV que podem ser executadas pelo operador com a permissão **Role_Autor**: ([**Manter Componente]; [Manter Grupos]; [Consome Ambientes]; [Pesquisa Repositórios];).** Após utilizar o sistema o usuário pode selecionar a opção sair, o que desvia o fluxo para o item 7;

5. Usuário com **Role_Consumidor** tem acesso a interface web com as opções: **[Pesquisa Repositórios]; [Consome Ambientes]; [Pesquisa Componentes]; [Listar Grupos de repositório])**, Após utilizar o sistema o usuário pode selecionar a opção sair, o que desvia o fluxo para o item 7;

6. O Usuário com **Role_Administrador** tem acesso à maioria das operações do sistema mais as opções: ([**Manter Tipos de Arquivo], [Manter Usuários], [Manter repositórios]**). Após utilizar o sistema, o usuário pode selecionar a opção sair, o que desvia o fluxo para o item 7;

7. Encerra o fluxo de operação genérico que poderia ser implementado por aplicações clientes.

Este é um modelo de fluxo genérico de referência na codificação dos sistemas que vão realizar integração com a arquitetura apresentada ou implementar repositórios. No entanto, não é fixa, ou seja, podem ser configurados novos fluxos alternativos, dependendo das necessidades.

Trata-se de uma recomendação não obrigatória. Isso dá a liberdade para que os desenvolvedores implementem seus clientes de forma customizada de acordo com as necessidades do projeto.

4.2. Descrição da implementação e Arquitetura da Integração

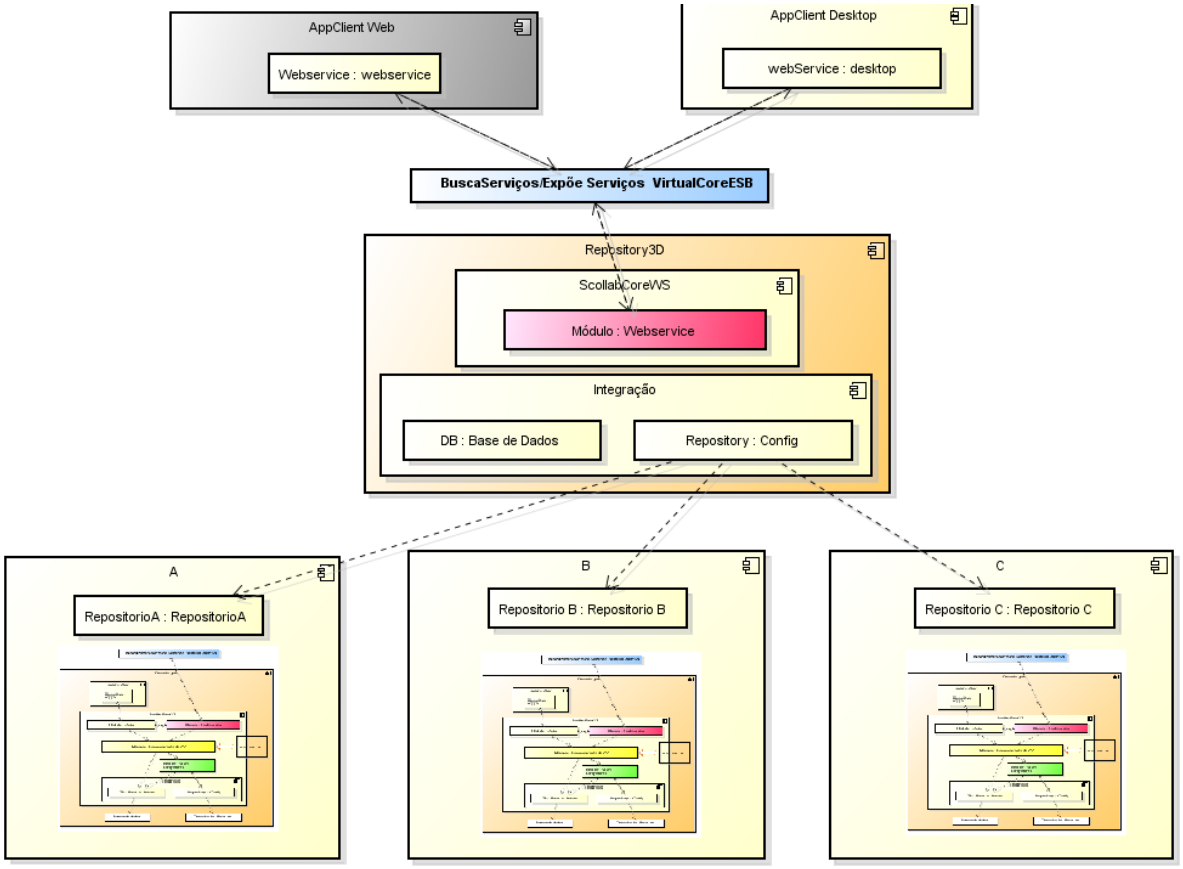
Os ambientes foram incluídos nos formatos descritos na Seção 0, também deve ser ressaltado que nenhum dos componentes 3D persistidos neste trabalho de tese foi criado pelo autor e sim baixados dos seguintes websites Smithsonian[45], Ibiblio[46] e X3DOM[47]. Outros tipos de arquivo poderiam ter sido explorados, no entanto, este trabalho de tese limitou o escopo aos três tipos explicados na contextualização.

No desenvolvimento das aplicações, conforme já foi abordado anteriormente é

[45]Smithsonian – disponível em: <http://3d.si.edu/browser>. Acesso em: fev. 2015.
[46] Ibiblio – disponível em: < http://www.ibiblio.org/e-notes/webgl/webgl.htm> acesso em: fev. 2015.
[47] X3DOM examples – disponível em: < http://examples.x3dom.org/>. Acesso em: fev. 2015.

utilizado a customização do *Framework* ScollabCore (FUJIOKA, 2011) descrito na Seção 4.1.5, na qual foram adicionados componentes relacionados a busca e recuperação de AV utilizando *webservices*. A utilização desse framework permitiu que a implementação fosse reduzida, uma vez que não foi necessário projetar a aplicação completa, ou seja, foram reaproveitados os componentes existentes bem como sua organização interna para tratamento de transações, persistência e concorrência. Para prova de conceito, foram implementadas duas aplicações que são descritas na Seção 4 e 4.5.

Figura 4.14 - Arquitetura das aplicações Integradas.



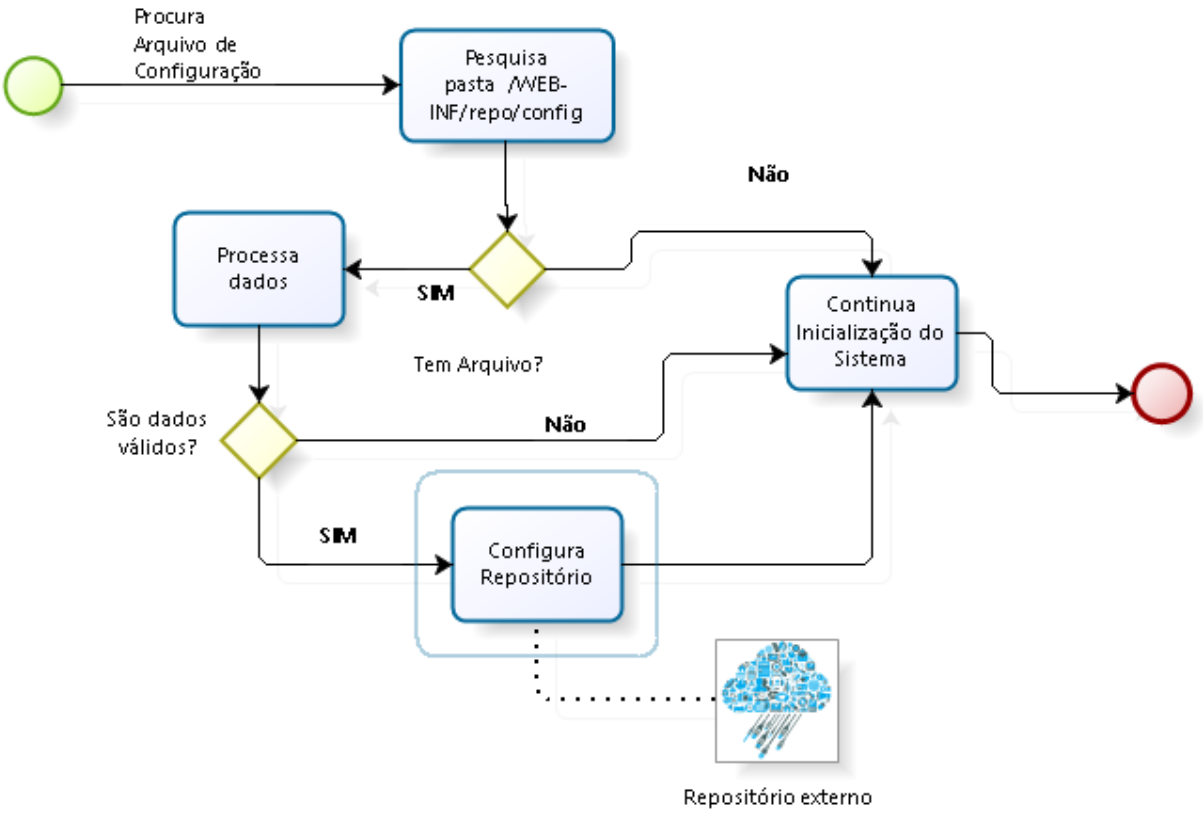
Fonte: O autor

A Figura 4.14 apresenta o arcabouço da arquitetura após a integração das duas ferramentas que foram programadas nesse trabalho e estão contextualizadas na Seção 4.4 e 4.5. O componente representado pelo *AppClient Desktop* é ilustrativo. Esse modelo também exibe a integração possível com outros três (3) repositórios encontrados fora da arquitetura, **o repositório A**, **repositório B** e o **repositório C**.

Para habilitar a utilização de repositórios externos é preciso incluir um arquivo de

configuração com o seguinte nome **configuracaorep.xml** na pasta WEB-INF/repo/config. Esse local é percorrido na inicialização do sistema pelo módulo **Repository:Config,** que recupera o arquivo e processa os seus dados para executar buscas externas quando solicitado conforme ilustra a Figura 4.15.

Figura 4.15 – Processo de busca e configuração de repositórios externos.



Fonte: O autor.

A Figura 4.16 apresenta a estrutura e a forma como o arquivo de configuração deve ser preenchido, de forma que a aplicação seja capaz de realizar a leitura e estabelecimento da conexão com outros repositórios.

Figura 4.16 – XML de configuração dos repositórios.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<repositorios>
 <repositorio>
      <nome>Pão simples</nome>
      <configuracao>
        <url ativa="true">repositorio.instituicao.com.br:8080/repositorio</url>
        <login>Luke</login>
        <token>!@#AJSHDUJQ@!@#!@#</token>
        <indexar>true</indexar>
      </configuracao>
      <instrucoes>
        <passo>Verifique as conexões e a liberação da porta 8080</passo>
        <passo>Certifique-se de estar conectado na Internet.</passo>
        <passo>Esse é um repositório de objetos tridimensionais da Instituição .</passo>
      </instrucoes>
   </repositorio>
</repositorios>
```

Fonte: O autor

Cada uma das ***tags*** exibidas na Figura 4.16 identificam o repositório e os dados necessários para configurá-lo, o objetivo de cada uma delas é:

- <**Repositorios**> - Define a raiz do arquivo onde todos os repositórios devem ser descritos;
- <**Repositorio**> - Indica as configurações do repositório. Essa tag pode aparecer mais de uma vez dentro da **tag** <**Repositorios**>;
- <**Nome**> - Um nome para o repositório para fins de organização e localização;
- <**Configuração**> - contém os parâmetros de acesso e autenticação se forem necessários.
    - **Url** – onde é preenchido o endereço do webservice e se o repositório está ativo ou não;
    - **Login** – caso o repositório necessite de **autenticação** é informado o usuário nesse ponto; e
    - **Token** – Token de acesso ao repositório, se necessário.
- <**Instruções**> - Descreve o repositório e dá instruções se for o caso.

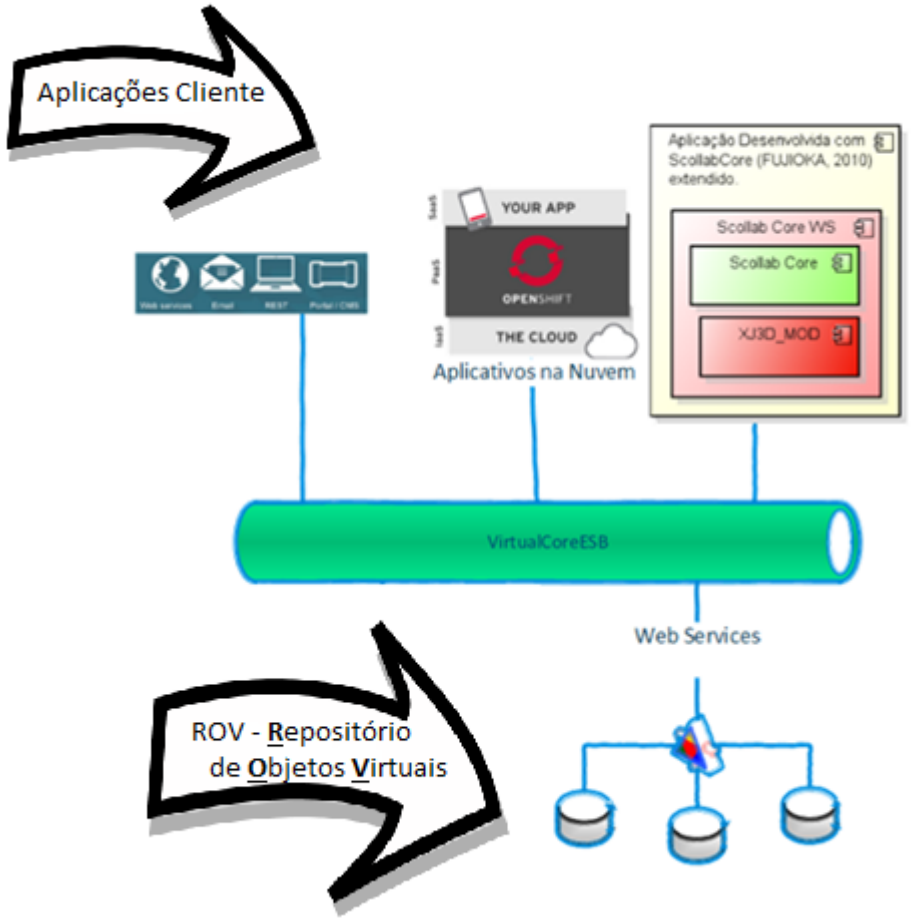
Por exemplo, se um pesquisador/desenvolvedor desejar implementar seu aplicativo sem utilizar a infraestrutura proposta, poderá selecionar qualquer linguagem de programação que dê suporte à utilização de *Web Services* utilizando o protocolo SOAP. Essa flexibilidade permite a interoperabilidade entre aplicações. Isto é possível em função da inexistência de dependência dessa tecnologia.

Na Figura 4.17 é exibido o modelo de execução após a implantação do repositório e da aplicação em nuvem, onde as aplicações desenvolvidas em qualquer plataforma podem

interagir com a infraestrutura proposta neste trabalho. Tais aplicações podem, portanto, consumir os artefatos persistidos no repositório.

Durante a execução dos testes os aplicativos desenvolvidos foram implantados em duas plataformas de *Cloud Computing*, o OPEN-SHIFT da REDHAT[48] e a Digital Ocean [49]. Nesta abordagem se verifica a utilização de modelos de serviços categorizados como: *Software as a Service*[50] (**SaaS**), *Platform as a Service*[51] (**PaaS**) e o *Infrastructure as a Service*[52] (**IaaS**) (Ver Seção 3.3).

Figura 4.17 – Execução e utilização para a arquitetura.



[48] OPEN-SHIFT – Serviço PaaS da Red Hat (RED HAT, 2015).
[49] DIGITAL OCEAN – Serviço PaaS com diversos recursos com baixo custo (DIGITALOCEAN, 2015).
[50] SaaS – S*oftware as a Service* refere-se a aplicações standard residentes na *cloud*, disponíveis pela internet normalmente por browser. Todos podem ser utilizadores deste serviço, interagem diretamente com o software na *cloud* tal como se o mesmo estivesse instalado localmente. Como exemplo de aplicações SaaS conhecidas estão: gmail ou google maps (MARCHÃO, REIS, 2013).
[51] PaaS – *Platform as a Service* disponibilizam-se ambientes de desenvolvimento sobre os quais as aplicações são construídas, normalmente baseadas em modelo SOA (MARCHÃO, REIS, 2013).
[52] IaaS – *Infrastructure as a* Service oferece recursos computacionais como processamento ou armazenamento, os quais podem ser obtidos como se fossem um serviço na web (MARTINS, 2010).

Fonte: O Autor.

Não se pode deixar de destacar que essa perspectiva deverá abrir novas oportunidades de investigação no contexto da pesquisa realizada neste trabalho, configurando boa oportunidade de desenvolvimento de trabalhos futuros em relação à escalabilidade e à qualidade na distribuição de artefatos.

Outro item que deve ser esclarecido é que os objetos virtuais consultados para descrição de mundos em três dimensões poderão ser persistidos utilizando outras tecnologias além de X3D (WEB3Da, 2014), X3DOM (X3DOM, 2014) ou a WebGL (KHRONOS,2014). A arquitetura proposta é voltada para geração e disponibilização de AV e não renderização dos mesmos. Desta forma é possível realizar ajustes para adaptar este modelo a qualquer tecnologia.

Também é importante ressaltar que o usuário poderá gerar mundos virtuais a partir de uma infraestrutura de reuso sem a necessidade de qualquer experiência na construção de AV, ou seja, o ambiente vai ser recuperado de um repositório com base nos critérios de busca informados na aplicação cliente.

4.3. Método de Validação

A metodologia é o conjunto de atividades empregadas na investigação e o método representa o fluxo no qual as operações necessários são executadas para alcançar os objetivos e resultados desejados (CERVO; BERVIAN, 2002). Assim, conforme a classificação metodológica, este trabalho é caracterizado da seguinte forma:

Quanto à abordagem de pesquisa - Predominantemente qualitativa. A definição de uma **A**rquitetura de **R**eferência para reuso de **OV** e distribuição de **AV** utilizando um **CR**. Tal definição é realizada a partir de modelos de forma que, a pesquisa foi executada qualitativamente em relação à produção de protótipos de uma abordagem particular da qual não se pode extrair números. A implementação desses protótipos segue o padrão definido na **AR,** sendo suas funcionalidades aderentes ao proposto para o sistema.

Quanto ao método de pesquisa - Esse trabalho é classificado como um "*Estudo de Caso*" do tipo único[53], pois a pesquisa foi composta de várias técnicas, e ao final foi realizada a análise dos resultados obtidos no intuito de encontrar respostas para as hipóteses levantadas.

[53] Tipos de estudo de caso conforme Yin (2005): **Casos Únicos** são adequados quando representam o caso decisivo ao se testar uma teoria bem formulada ou é um caso raro ou extremo ou ainda um caso revelador; ou **Casos Múltiplos** usam uma lógica de replicação para prever resultados semelhantes ou reduzir resultados contrastantes.

Verificar a viabilidade de um repositório para AV com RV utilizando *Web Services* em em SOA.

Quanto aos objetivos de pesquisa - "*Exploratória*", pois é composta basicamente por uma revisão da literatura seguida pela proposta de uma arquitetura, desenvolvimento de protótipos experimentas e sua avaliação. Além disso, Yin (2005) concorda, já que trata de problemas pouco conhecidos, objetivando definir hipóteses ou proposições para futuras pesquisas "**viabilidade de compartilhamento de AV independente de tecnologia**".

Quanto à natureza da pesquisa: É classificada como sendo "*Aplicada*", pois o seu objetivo principal tem uma definição prática determinada: Modelagem e definição de uma Arquitetura de Referência para Gerenciamento e Reuso de Ambientes Virtuais Tridimensionais.

Quanto aos procedimentos de pesquisa - Sua caracterização é bibliográfica, uma vez que a execução da mesma foi preparada a partir de material já publicado, composto principalmente procedentes de artigos recuperados de periódicos, conferências, livros e de especificações de padrão de tecnologias.

Essas diferentes abordagens em relação ao método proporcionam ao pesquisador maior flexibilidade, contribuindo para uma compreensão geral da natureza desse tipo de pesquisa (MERRIAM, 2009). Cada um desses tipos é aplicado em modelos de abordagem com características distintas.

Assim as próximas seções apresentam os experimentos realizados e também a análise dos resultados encontrados com sua execução.

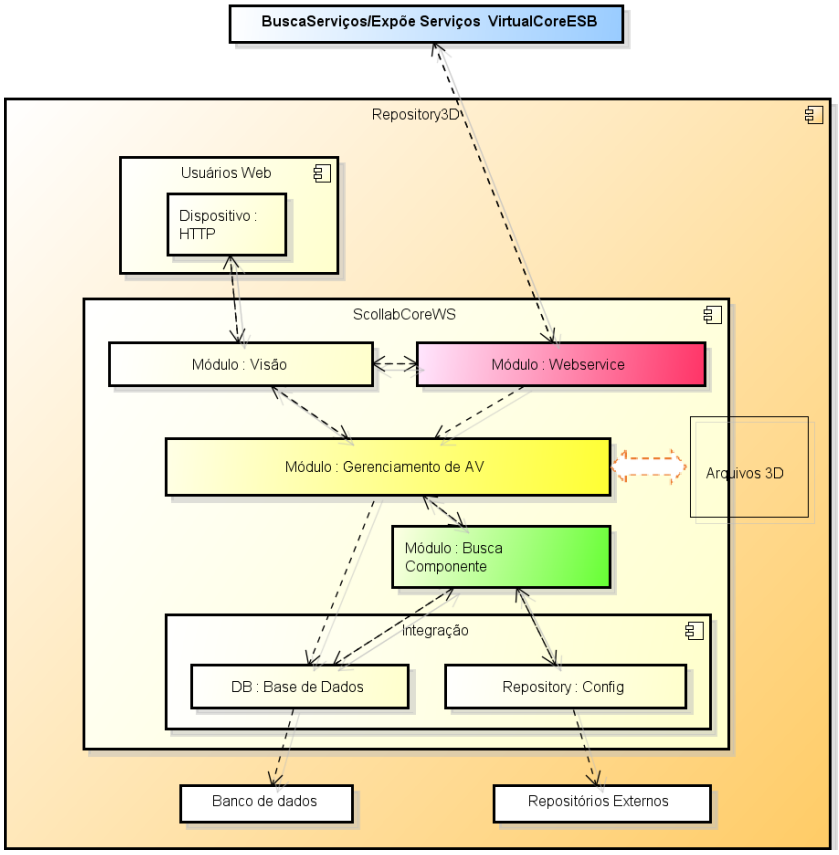
4.4. Estudo de caso (Repositório)

O primeiro experimento consistiu no desenvolvimento de um repositório capaz de gerenciar e disponibilizar AV utilizando *webservices.* A implementação busca responder a seguinte pergunta (b) elencada nos objetivos gerais (Ver Seção 1.1):

"É viável utilizar técnicas de reutilização dentro de uma infraestrutura de reuso de componentes virtuais que funcione de forma distribuída utilizando uma Arquitetura Orientada a Serviços?"

Seguindo esta linha, a Figura 4.18 ilustra a arquitetura do repositório que foi gerado a partir da arquitetura de referência descrita na Seção 4.1.1.

Figura 4.18 – Base da infraestrutura de reuso baseada em SOA



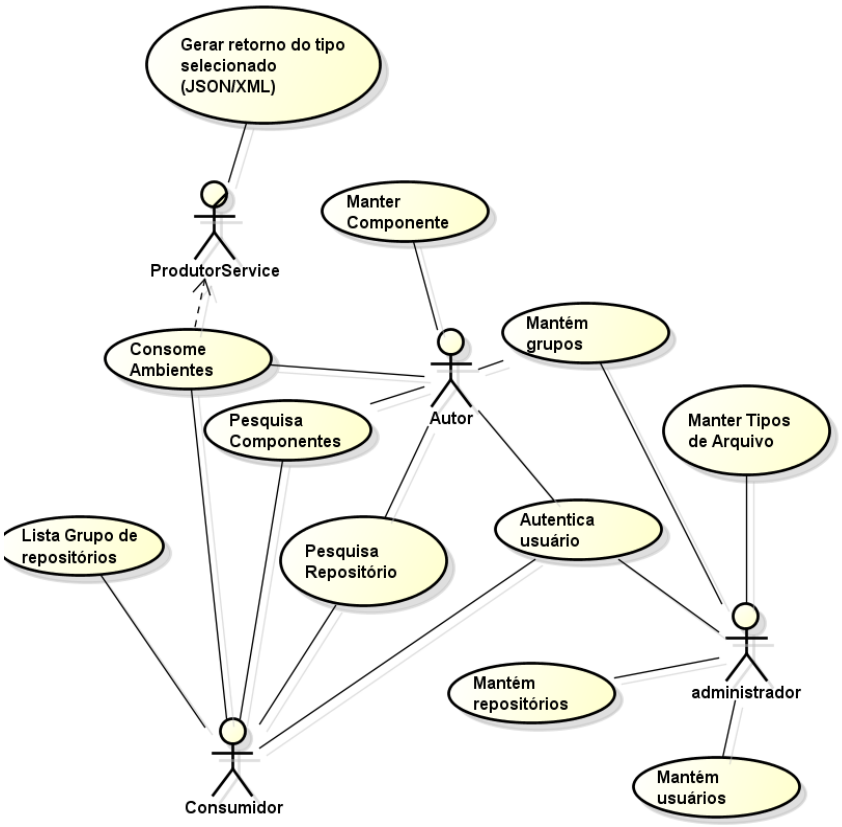
Fonte: O autor.

Nesse modelo a interação entre os módulos é realizada por meio do barramento VirtualCoreESB que se comunica com a Internet, permitindo que aplicações externas realizem a pesquisa e recuperação de AV. Já o diagrama de casos de uso especificado e utilizado para implementação da aplicação repositório é ilustrado pela Figura 4.19

Nele, são definidas as atividades que podem ser realizadas na visão de cada tipo de ator (Ver Seção 4.1.4). Tal modelo auxiliou na implementação desse experimento.

Para isso foram implementadas as funcionalidades essenciais na realização dos testes desse estudo de caso: **Manter Componentes, Pesquisa Componentes, Cadastra categoria e agrupamento de componentes por tipo de arquivo**. O Objetivo foi implementar um repositório, o mais simples possível e na sequência consultar os AV persistidos por uma aplicação cliente que é descrita na Seção 4.5.

Figura 4.19 – Diagrama de casos utilizado para implementação organizado por Perfil de usuário.



Fonte: O autor

Quando o ator realiza a operação de *Login*, o sistema identifica os papéis que estão associados ao operador e realiza o direcionamento para o fluxo de telas apropriado. O sistema exibe ou oculta as operações em função do perfil conforme explicado na Seção 4.1.7.

O controle de acesso é gerenciado pelo módulo de segurança do ScollabCore (FUJIOKA, 2011) com o SpringFramework (SPRING, 2015). Na página responsável pela exibição das opções do menu é utilizada a tag “<sec:authorize access="hasRole('nome Da regra')">” que realiza essa verificação e exibe o trecho correspondente ao perfil do usuário conforme ilustrado no Quadro 4.1.

Quadro 4.1 – *Tag* de segurança que verifica quais opções serão exibidas na tela.

```
<!-- Usuário Comum  -->
<sec:authorize access="hasRole('ROLE_CONSUMIDOR')">
    <h:commandButton action="#{repositorioController.carregar}">Pesquisa Repositórios</h:commandButton>
    <h:commandButton action="#{componenteController.carregar}">Pesquisa Componentes</h:commandButton>
    <h:commandButton action="#{webServiceController.carregar}">Consome ambientes</h:commandButton>
    <h:commandButton action="#{grupoController.carregar}">Listar Grupos de repositório</h:commandButton>
</sec:authorize>

<!-- Administrador -->
<sec:authorize access="hasRole('ROLE_ADMINISTRADOR')">
<h:commandButton action="#{repositorioController.carregar}">Pesquisa Repositórios</h:commandButton>
<h:commandButton action="#{categoriaController.carregar}">Manter Catgorias</h:commandButton>
<h:commandButton action="#{usuarioController.carregar}">Manter Usuários</h:commandButton>
<h:commandButton action="#{relatorioController.carregar}">Emite Relatórios</h:commandButton>
<h:commandButton action="#{tipoArquivoController.carregar}">Manter tipos de Arquivo</h:commandButton>
<h:commandButton action="#{grupoController.carregar}">Manter Grupos</h:commandButton>
<h:commandButton action="#{webServiceController.carregar}">Consome ambientes</h:commandButton>
</sec:authorize>

<!-- Autor -->
<sec:authorize access="hasRole('ROLE_AUTOR')">
    </h:commandButton action="#{manterComponenteController.carregar}">Manter Componentes</h:commandButton>
    </h:commandButton action="#{manterComponenteController.carregarAssociacao}">Associa Objetos 3D</h:commandButton>
    </h:commandButton action="#{notificadorController.carregar}">Notifica Atualizações</h:commandButton>
    </h:commandButton action="#{categoriaController.carregar}">Manter Catgorias</h:commandButton>
    </h:commandButton action="#{grupoController.carregar}">Manter Grupos</h:commandButton>
    </h:commandButton action="#{repositorioController.carregar}">Pesquisa Repositórios</h:commandButton>
    </h:commandButton action="#{manterComponentesController.carregar}">Pesquisa Componentes</h:commandButton>
    <h:commandButton action="#{webServiceController.carregar}">Consome ambientes</h:commandButton>
</sec:authorize>
```

Fonte: O autor

Além dessa configuração, ainda existe o controle realizado em nível de contexto, ou seja, se um usuário tentar acessar um recurso que não tem autorização o sistema bloqueia o acesso. O Quadro 4.2 exibe o arquivo de configuração responsável por esse controle. Esse arquivo também é configurado no core do ScollabCore.

Quadro 4.2 – Arquivo de configuração da segurança do Spring configurado no ScollabCore para esse estudo de caso.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:sec="http://www.springframework.org/schema/security"
    xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd ">

    <!-- Habilita a utilização de annotations nos Beans -->
    <context:annotation-config />

    <!-- Realiza o scanneamento das pastas indicadas para construção dos beans -->
        <context:component-scan base-package="com.rodrigofujioka.scollabcore.webcore.controle.mb" />
        <context:component-scan     base-package="com.rodrigofujioka.scollabcore.webcore.integracao.dao" />  1
        <context:component-scan     base-package="com.rodrigofujioka.scollabcore.webcore.negocio" />

    <!-- ****************** AUTHENTICATION PROVIDER ***************** -->
    <sec:http>
        <sec:intercept-url pattern="/consumidor/*" access="hasRole('ROLE_CONSUMIDOR')" />
        <sec:intercept-url pattern="/autor/*" access="hasRole('ROLE_AUTOR')" />
        <sec:intercept-url pattern="/administrador/*" access="hasRole('ROLE_ADMINISTRADOR')" />  2


        <sec:form-login login-page="/loginps"   default-target-url="/principal"
            authentication-failure-url="/loginError"/>
                                                                                    3
        <sec:logout logout-url="/logout" logout-success-url="/loginps?logout" />

        <sec:csrf />
    </sec:http>

    <bean id="autenticacaoService"
        class="com.rodrigofujioka.scollabcore.webcore.negocio.service.global.seguranca.AutenticacaoServiceImpl">  4
    </bean>

    <sec:authentication-manager>
        <sec:authentication-provider ref="autenticacaoService" />
    </sec:authentication-manager>

</beans>
```

Fonte: O autor

Para ajustar a configuração às necessidades desse projeto foram necessárias pequenas alterações para informar quais os caminhos e quais perfis poderiam realizar o acesso a determinado recurso. Os itens enumerados no Quadro 4.2 são:

1) Informa quais os pacotes da aplicação que serão percorridos em busca de classes para serem construídas. Encarrega-se de criar objetos na memória sem a necessidade de instanciação manual no código.;
2) Informa quais as regras de URL que serão tratadas. Nesse caso só permite a essas pastas usuários autenticados com as devidas permissões;
3) Configura quais páginas serão exibidas para o usuário caso ele não esteja autenticado ou para erros na autenticação ou autorização;
4) Indica qual classe a ser utilizada para autenticação no sistema. Nesse ponto o programador pode personalizar a forma de autenticação conforme as necessidades do projeto.

Para execução desse repositório alguns casos de uso foram implementados, seguindo-se a referência da Figura 4.19 e são contextualizados a seguir:

- **Pesquisa de repositórios** - O usuário pode listar quais repositórios estão configurados na aplicação. Esta tela é ilustrada pela Figura 4.20. Tal caso de uso permite que o usuário liste todos os repositórios ou aplique filtros parametrizados com o nome do repositório, descrição, versão ou a data de publicação. A inclusão, alteração ou exclusão de repositórios só pode ser realizada com usuários que possuem perfil Role_Administrador.

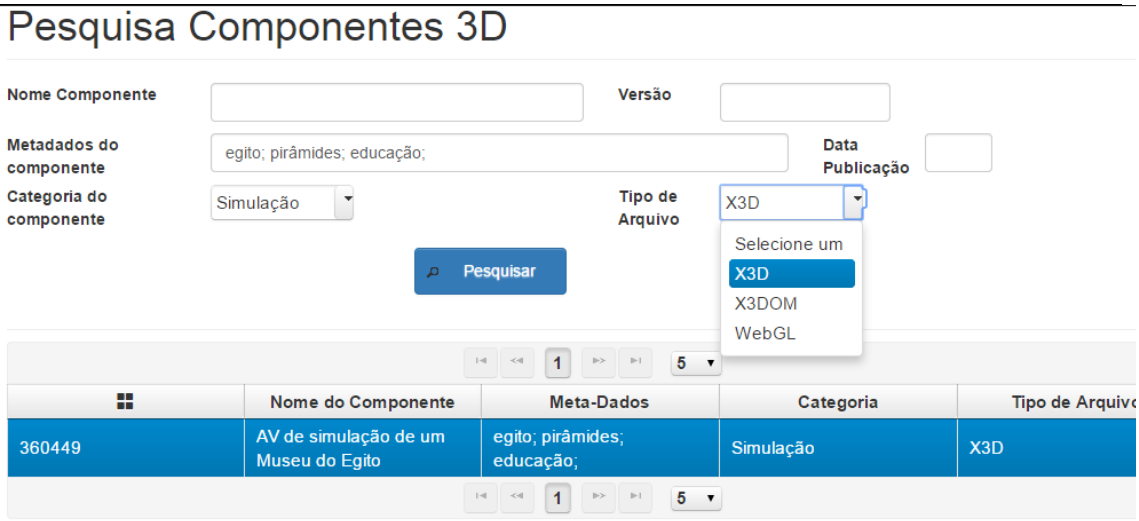
Figura 4.20 – Visão da opção pesquisa repositórios

Fonte: O autor

- **Pesquisa Componentes** - Neste caso de uso é possível que o usuário efetue pesquisas por componentes persistidos na infraestrutura do repositório configurado. A Figura 4.21 exibe o resultado da busca realizada por um dos componentes (Objetos 3D) que foram incluídos para teste. Nesse caso, foi passado um conjunto de meta-dados separados por “;” (ponto e vírgula). Esse conjunto de palavras é incluído em uma lista que é passada como parâmetro para uma consulta JPQL[54] que busca e retorna resultados relacionados.

[54] *Java Persistence Query Language:* linguagem de consulta utilizada para realizar consultas com *frameworks* de Mapeamento Objeto Relacional como o Hibernate. Trata-se de um tipo de dado bastante semelhante a SQL só que adaptada ao contexto dos objetos em Java. Mais informações a respeito são fornecidas na seção 4.5 que descreve o ScollabCore e as tecnologias de implementação.

Figura 4.21 – Tela pesquisa componentes 3D

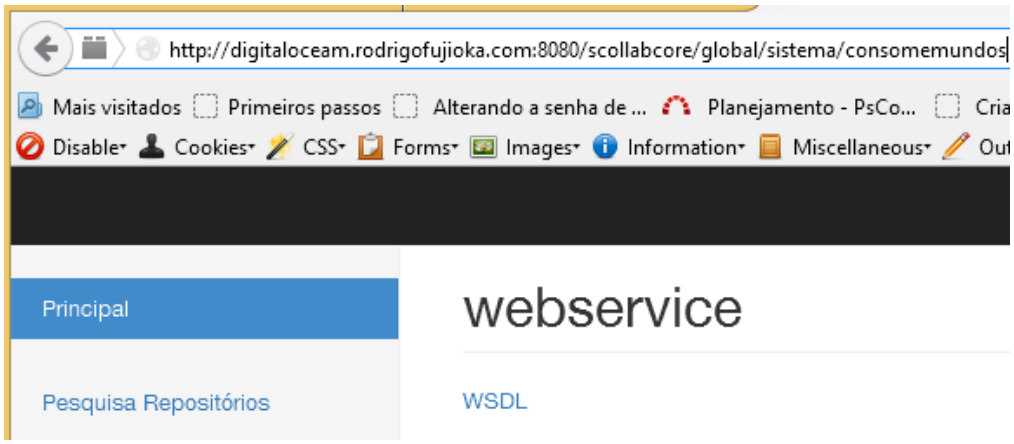


"SELECT C FROM COMPONENTE C WHERE C.ID IN (SELECT M.ID_COMPONENTE FROM METADADOS M WHERE M.NOME IN (:LISTAMETADADOS))"

Fonte: O autor

- **Consome ambientes**: Essa funcionalidade exibe apenas a página com o endereço dos WSDLs que devem ser implementados pelas aplicações clientes, caso queiram consumir os componentes da arquitetura. A Figura 4.22 apresenta a tela consumir mundos quando a opção é invocada. Observe que apenas um link é apresentado contendo o endereço de acesso ao *Web Service*.

Figura 4.22 – Tela após seleção da opção *consumir mundos*.



Fonte: O autor

A Figura 4.23 exibe a página que é exibida após o clique no link da Figura 4.22. Nela é exibido o endereço do WSDL de acesso ao serviço no qual estão disponíveis as operações. O *clique* no *link* da coluna informações, direciona o usuário para o arquivo do WSDL (Ver Figura B.2.1 e Figura B.2.2 – Apêndice B) contendo o arquivo **XML** detalhado com os serviços que podem ser invocados e como deve ser realizado esse acesso, bem como os parâmetros e tipos de retornos esperados após a invocação. Esse link deve ser utilizado na aplicação cliente para invocar as operações de consumo, busca e listagem de componentes.

Figura 4.23 – Descrição do WSDL após clicar no link Figura 4.22

Web Services

Ponto Final	Informações
Nome do Serviço: {http://ws.scollabcore.rodrigofujioka.com/}RepositorioServiceService Port Name: {http://ws.scollabcore.rodrigofujioka.com/}RepositorioServicePort	Endereço: "http://digitalocean.rodrigofujioka.com:8080/scollabcore/ws/AVWS WSDL: "http://digitalocean.rodrigofujioka.com:8080/scollabcore/ws/AVWS?wsdl Classe de implementação: com.rodrigofujioka.scollabcore.negocio.ws.AVWS

Fonte: O Autor

- **Manter Categorias**: Esse item gerencia as categorias de AV disponíveis no repositório. A Figura 4.24 exibe listagem de categorias salvas e o conjunto de operações que podem ser realizadas, tais como [Cadastrar Categoria], [Cadastrar Subcategoria], [Cadastrar Componente] e [Listar Objetos].

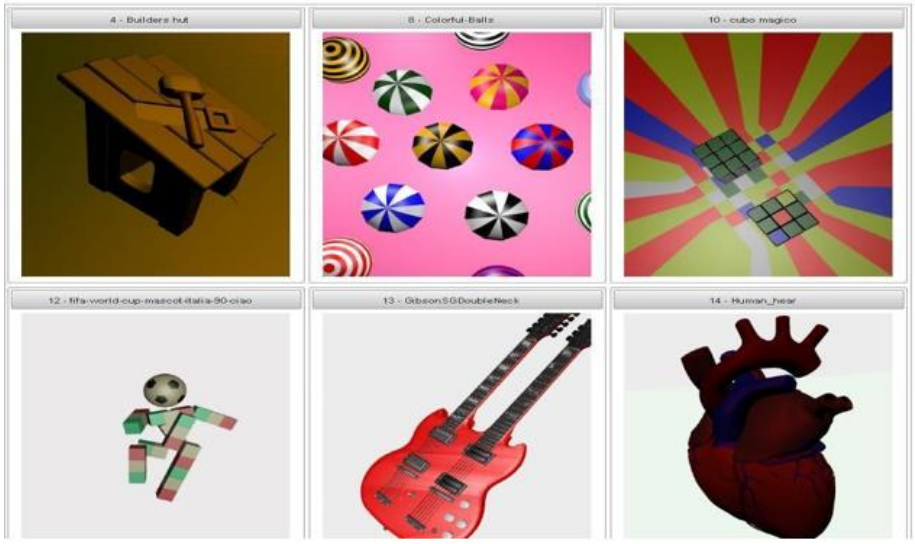
Figura 4.24 – Manter Categorias

Fonte: O autor

A atividade manter componentes permite incluir objetos, enquanto a opção manter categorias é utilizada para associar o objeto a uma categoria existente. A Figura 4.25 exibe a listagem de objetos em 3d cadastrados no repositório direto no *browser* sem a necessidade de se utilizar *plugin*[55]. Essa listagem foi invocada a partir da funcionalidade *manter categorias*.

[55] A exibição de objetos 3D sem a utilização de *plug-ins* com X3D só é possível com a conversão dos **OV** X3D para X3DOM de forma que a WebGL se encarrega de exibir os **OV**.

Figura 4.25 – Lista de Objetos 3D cadastrados no repositório.

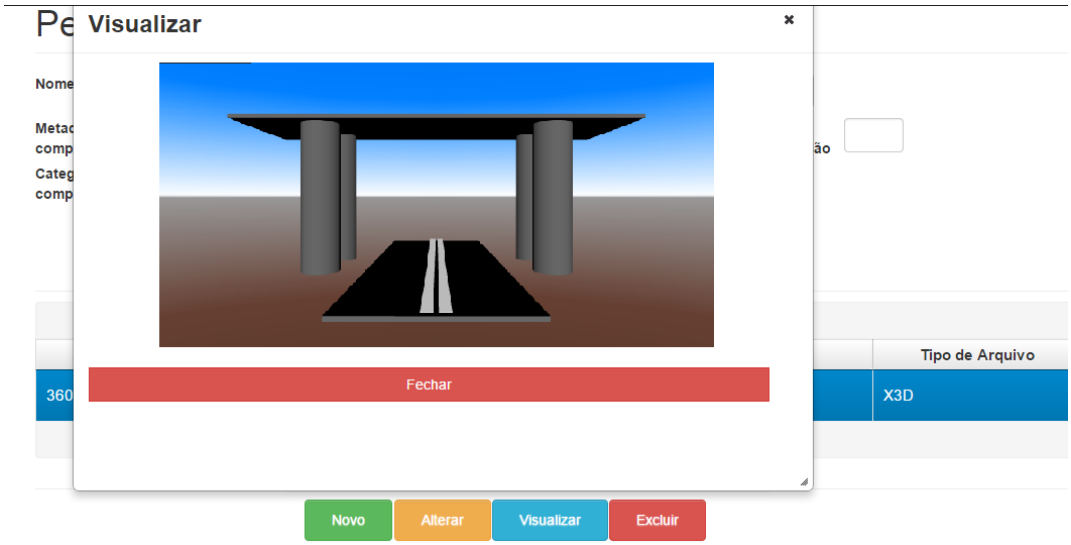


Fonte: O Autor

- **Manter Componente**: A Figura 4.26 exibe a tela dessa funcionalidade na qual é possível **incluir** objetos 3D e **Visualizar** componentes sem auxílio de *plugin.* Além das opções **Alterar**, na qual o operador pode alterar títulos e meta-dados do componente ou Excluir. Observe que para tipos X3D é realizada a criação de um arquivo X3DOM convertido com a ferramenta Instantreality[56]. Tal geração é realizada para que seja possível exibir a miniatura em 3D dos componentes persistidos, coisa que com o X3D não seria possível realizar sem a utilização de um *plugin*, pois os navegadores web não possuem o motor de execução para esse tipo de arquivo.

[56] InstantReality. Disponível em: <http://doc.instantreality.org/tools/x3d_encoding_converter/>. Acesso em: 15 de fev. 2015

Figura 4.26 – Manter Componentes



Fonte: O autor

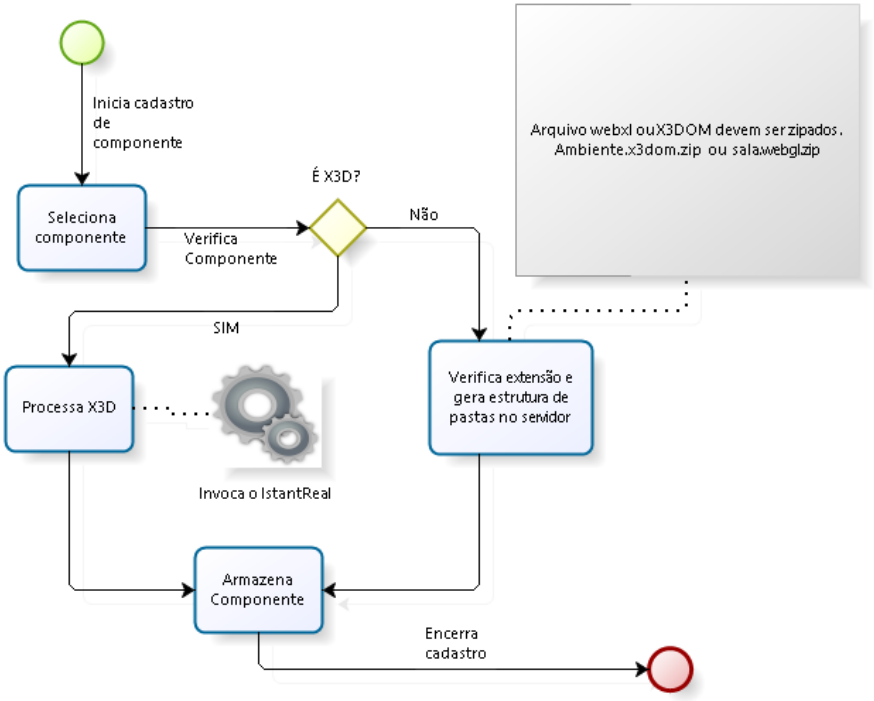
O processo de cadastro de um componente no repositório é exibido pela Figura 4.27. Note-se que para os tipos de arquivos com dependência de arquivos *JavaScript* é necessário compactar a estrutura de arquivos no formato ZIP.

Já para arquivos X3D existe a conversão e geração de um arquivo X3DOM. Esse procedimento inicia o fluxo com uma requisição de inclusão ao repositório, seguindo-se da verificação do tipo. Quando é identificado o tipo X3D é realizado o processo de geração do X3DOM utilizando a ferramenta InstantReality[57]. Esse processo gera uma pasta com o ID (identificador) do componente contendo a estrutura necessária para exibir o objeto.

Para componentes WebGL, o processo é semelhante, embora não exista a conversão. No entanto, a estrutura de pasta é criada extraindo-se o arquivo enviado. Assim, ao ser chamado pela tela Manter Componentes é possível visualizar o AV.

[57] Ferramenta para conversão de arquivos x3d em x3dom. Disponível em: <http://www.instantreality.org/>. Acesso: em dez. 2014.

Figura 4.27 – Processo de inclusão de um arquivo no repositório.



Fonte: O autor

Essa seção descreveu o experimento da implementação do repositório com base na Arquitetura de Referência. Nela foram ilustradas e descritas algumas funcionalidades da interface web, além de ter sido descrito o arquivo WSDL que é exposto, permitindo o acesso externo de outras aplicações por meio da invocação das operações do serviço de busca e recuperação de componentes.

Também foi utilizado um diagrama de casos de uso básico de referência para implementação do repositório. A próxima seção apresenta o experimento elaborado para consumir os objetos desse repositório como prova de conceito para validação da arquitetura, no que se refere ao acesso e à interoperabilidade de ferramentas.

4.5. Estudo de Caso (Cliente)

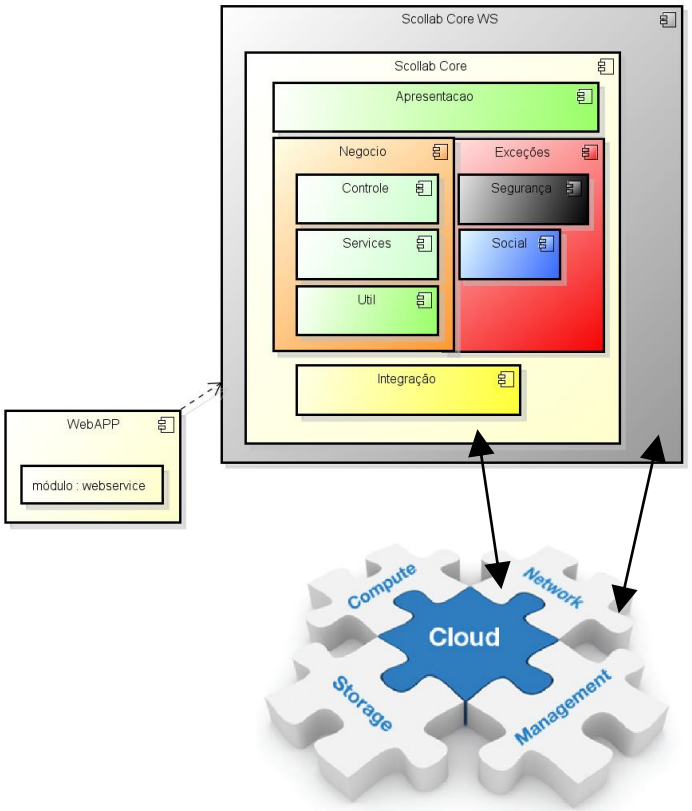
Após a construção de um repositório a partir da Arquitetura de Referência definida e da configuração dos *Web Services* de compartilhamento, foi desenvolvida uma aplicação cliente. O objetivo desta aplicação é testar a conectividade bem como a recuperação de **OV** do **ROV** invocando para isso as operações disponibilizadas pelo barramento ESB. Tal barramento é identificado pelo componente **VirtualCoreESB** especificado na **AR**. O

propósito dessa aplicação é a partir da invocação dos serviços, coletar evidências a respeito do modelo que foi utilizado para construção do repositório descrito na Seção 4.4.

A Figura 4.28 apresenta a arquitetura da aplicação cliente, representada na AR como **AppClient** (Seção 4.1.3). Nesse modelo a comunicação é habilitada com o desenvolvimento de *Web Services* consumidores que interagem com o repositório por meio do barramento ESB.

Para a instanciação da aplicação cliente foi utilizado o *framework* **ScollabCoreWS**. O acesso aos *Web Services* foi configurado no módulo de serviços do *framework*. Assim, foram construídas três telas, cada uma recuperando um tipo de componente do repositório.

Figura 4.28 – Arquitetura da aplicação Web desenvolvida a partir do ScollabCoreWS.

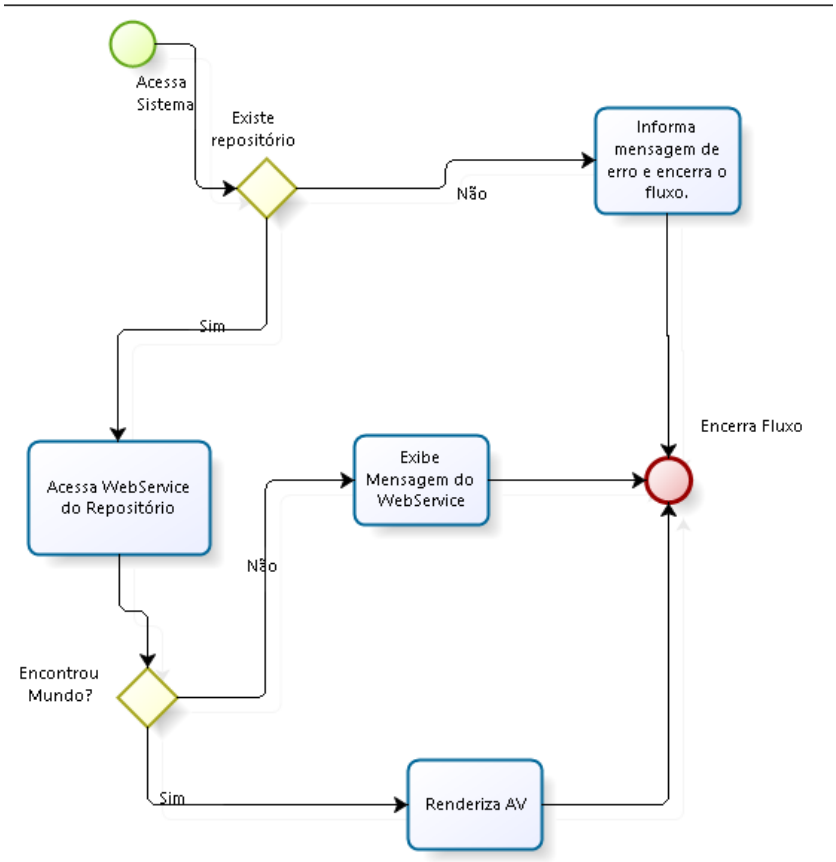


Fonte: O autor.

Para uma melhor compreensão do processo executado na aplicação cliente, é apresentado o fluxo na Figura 4.29. Os componentes dessa arquitetura são detalhados na Seção 4.1.5, com exceção do pacote WebAPP que foi desenvolvido para essa prova de

conceito seguindo o processo da Figura 4.29.

Figura 4.29 – Processo de requisição e invocação dos Componentes do Repositório.



Fonte: O Autor

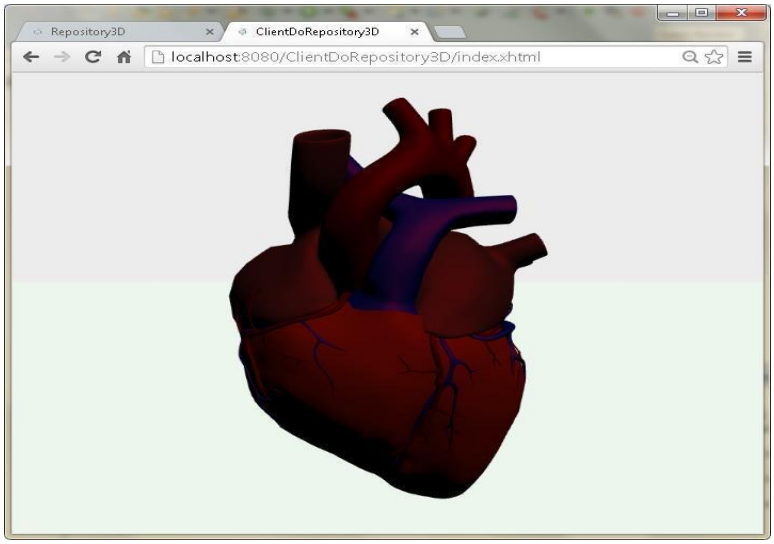
1. O sistema cliente realiza requisição ao **ROV** através de *Web Services* a partir do qual invoca uma das operações disponibilizadas. Caso não exista repositório disponível o sistema informa que não existe conectividade e vai para o fluxo 4. Caso contrário, o fluxo é desviado para o item 2;
2. Se existir repositório o sistema invoca as operações de busca disponibilizadas pelo ESB (**VirtualCoreESB**) do servidor com *Web Services.* Se existir **OV** com os parâmetros passados é realizada a recuperação do **OV** do **ROV** e vai para o fluxo 3. Caso contrário, uma mensagem informando que o **OV** não foi encontrado é exibida;
3. A aplicação recupera o **OV** e renderiza o mesmo na tela para o usuário;
4. Fluxo da aplicação é encerrado.

Com base nesse fluxo foi implementada uma aplicação cliente conforme já exposto nesta seção e a partir dela foram realizados testes de consulta aos **OV** do **ROV** de forma a verificar a viabilidade da **AR** elaborada neste trabalho.

A Figura 4.30 apresenta o resultado da execução do sistema após a invocação do serviço disponibilizado no **VirtualCoreESB** do repositório. Durante o processo de chamada, o identificador do **OV** é passado como parâmetro. Caso exista um componente com esse identificador persistido, o **SHELL** (Conjunto de componentes da AR) do repositório se encarrega de localizar e retornar o arquivo para aplicação cliente que fica responsável pela exibição do **OV** a seu critério. Para essa prova de conceito o objetivo era consultar objetos persistidos na infraestrutura da Seção 4.4 e renderizar para o usuário.

Os testes nos processos de acesso e invocação dos serviços tiveram êxito. O barramento ESB do repositório se comportou da forma esperada. Como nenhum motor de renderização é necessário para exibição de componentes 3D desenvolvidos com a X3DOM o **OV** foi exibido na aplicação Web cliente.

Figura 4.30 – Aplicação consumindo AV X3DOM do repositório diretamente.



Fonte: O autor

Já a Figura 4.31 exibe a execução de um modelo de urso desenvolvido com WebGL. O fluxo é idêntico ao executado para recuperação do coração na Figura 4.30. O que muda nesse caso é a tecnologia utilizada para persistir o objeto. Tanto para o processo de conexão como para recuperação o resultado foi positivo.

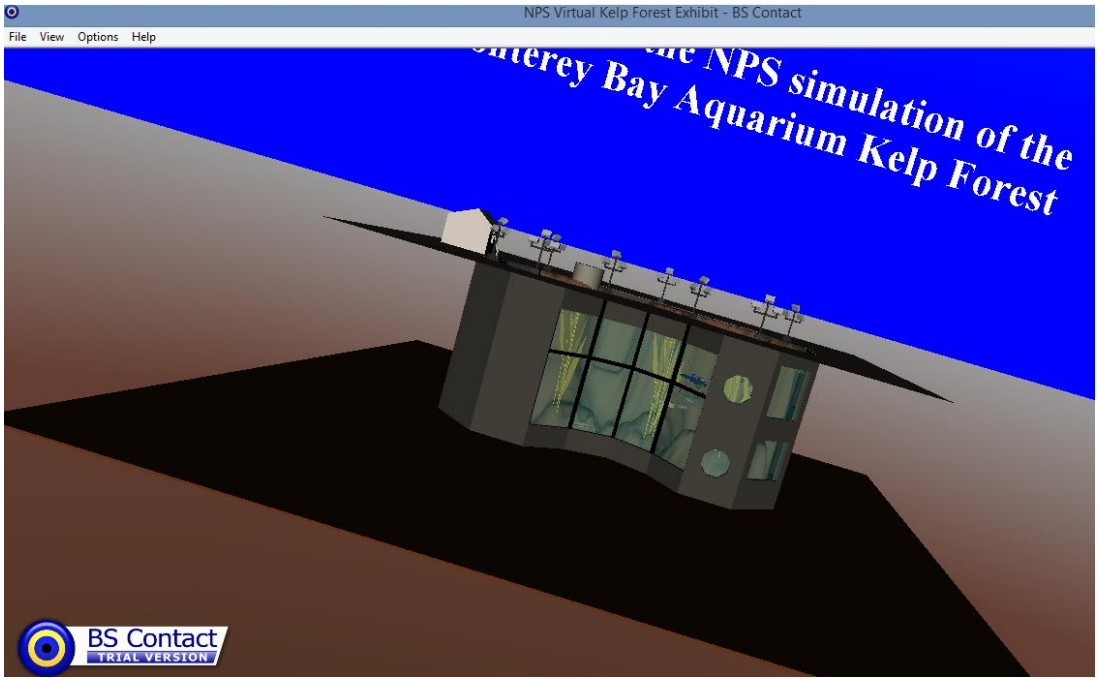
Figura 4.31 – Aplicação consumindo AV WebGl.

Fonte: O autor

Por fim, o processo executado para recuperação é executado novamente com parâmetros diferentes para recuperação de um componente utilizando X3D que é renderizado conforme ilustrado pela Figura 4.32. Essa renderização foi possível após a instalação do *plugin BS Contact*[58]. O fluxo para recuperação foi o mesmo das outras duas tecnologias, a diferença esta no motor de renderização que no caso do X3D depende de *plugin* ou ferramenta para ser exibido.

[58] BS CONTACT – Plugin para renderização de cenas x3d. Disponível em <http://www.bitmanagement.com/en/products/interactive-3d-clients/bs-contact>. Acesso em fev 2015.

Figura 4.32 – Ambiente X3D incluído no repositório “the kelp forest exhibit" (X3DGRAPHICS, 2015).



Fonte: O autor

Em relação ao tipo X3D poderia ter sido utilizado o *toolkit* Xj3d (Hudson et al., 2002), que trabalha com ambientes descritos na linguagem X3D. Entretanto, não faz parte do escopo deste trabalho tratar detalhes relacionados à exibição dos componentes do repositório.

Esse experimento consumiu os componentes do repositório acessando as operações do serviço disponibilizado na arquitetura por meio do barramento ESB **VirtualCoreESB**. Apesar de ter se utilizado do mesmo núcleo no desenvolvimento, é possível afirmar que qualquer software pode realizar o reuso dos componentes da infraestrutura, bastando para isso, implementar *Web Services* para consumir os artefatos armazenados na própria infraestrutura de rede ou em nuvem. Troca-se a implementação, mas é mantida a comunicação via *Web Services*, o que garante a interoperabilidade entre ferramentas.

As evidências coletadas com a execução desse experimento permitiram responder a pergunta **(c)** elencada nos objetivos gerais (Ver Seção 1.1):

c. É factível que a interoperabilidade entre sistemas de RV seja alcançada com disponibilização destes componentes utilizando *Web Services*?

4.6. Considerações finais

Neste capítulo foram apresentados a arquitetura de referência definida nesta tese, o modelo de dados, as arquiteturas dos experimentos, bem como o diagrama de casos de uso utilizado nas implementações. Também foram descritos os experimentos realizados bem como os resultados que foram obtidos.

Ainda, foi abordado o *framework* ScollabCore, que foi estendido e utilizado para construção dos aplicativos. Sua utilização foi essencial para reduzir o tempo e a qualidade do código uma vez que o retrabalho na construção de novos aplicativos foi minimizado. Além disso, foi descrita a extensão realizada no *framework* para dar suporte à arquitetura de referência definida neste trabalho.

O processo de extensão do *framework* bem como o desenvolvimento das aplicações conforme relatado nas seções anteriores, mostrou ser um processo relativamente simples, pois, não foi necessário desenvolver ou planejar toda arquitetura, uma vez que, todo núcleo foi reaproveitado do *framewok* **ScollabCoreWS**, visto que não houve alterações significativas no núcleo do referido *framework*.

5 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho abordou conceitos essenciais para o entendimento da tese descritos nos capítulos anteriores. Apesar do tempo prolongado entre o início do doutoramento até a elaboração do presente trabalho, observa-se que o tema inicialmente proposto se manteve como oportunidade de investigação ao longo dos anos.

Outro item que deve ser destacado é a participação de diversos alunos de graduação e pós-graduação. As teorias e conceitos investigados durante o processo de escrita deste trabalho foram responsáveis pela geração de trabalhos de conclusão de curso, dissertações de mestrado, projetos de pesquisa e extensão e um livro.

Na Seção 5.1 serão elencadas as contribuições desse trabalho, já a Seção 5.2 relata as limitações encontradas. Por fim, na Seção 5.3 são elencadas oportunidades de investigação, trabalhos futuros e também a produção acadêmica resultante desta pesquisa.

5.1 Contribuições e resultados

Os trabalhos localizados na literatura que tratam de reuso no contexto dos AV e utilizam RV e RA propõem um reuso muito dependente de tecnologia, ou seja, é um reuso no qual existe uma forte dependência de ferramentas para geração de AV ou com enfoque em tecnologias específicas. A estrutura da arquitetura de referência dessa tese considerou o gerenciamento de AV de forma que a sua disponibilização e gerenciamento fosse transparente, de forma a não obrigar os consumidores ou clientes a utilizarem unicamente uma dada tecnologia Assim, a arquitetura foi definida e implementada pensando-se em tratar o reuso em AV gerando as seguintes contribuições:

i. Um modelo de referência para reuso em RV com repositório independente de tecnologia (Seção 4.1.1);
ii. A definição de uma arquitetura de referência para a distribuição de AV flexível à utilização de tecnologias descritivas de mundos em 3 dimensões geradas a partir da contribuição (i) combinada com os estilos arquiteturais SOA e Camadas (Seções 4.1.2 e 4.1.3);
iii. Um modelo de dados para persistência e recuperação de componentes da arquitetura que contempla o acesso baseado em papéis (Seção 4.1.6);
iv. Desenvolvimento de repositório extensível baseado na arquitetura de referência definida que permite busca e recuperação de **OV** em um **ROV** independente de

tecnologia (Seção 4.4);

v. Aplicação cliente que invoca as operações do barramento ESB e consome os **OV** desse repositório (Seção 4.5);

vi. Um protocolo de busca para revisões sistemáticas em Reuso de Software no contexto dos AV que utilizam RV e RA (Apêndice A);

vii. Localização de temas não explorados que podem ser tratados cientificamente em trabalhos futuros conforme descrito na Seção 5.3;

viii. *Framework* ScollabCore (FUJIOKA, 2011) estendido, sendo gerada uma aplicação Java Web que se conecta na infraestrutura SOA;

Com a execução dos experimentos, foi possível mostrar que foram alcançados os objetivos gerais propostos neste trabalho e responder as seguintes questões:

a. Como explorar o reuso de componentes virtuais em ambientes tridimensionais, utilizando o conceito de reuso mantendo a interoperabilidade entre sistemas?

   R - Essa questão é respondida com as contribuições: (i, ii) geradas na Seção 4.4 e 4.5 onde aplicações são implementadas seguindo o modelo da **AR**.

b. É viável utilizar técnicas de reutilização dentro de uma infraestrutura de reuso de componentes virtuais que funcione de forma distribuída utilizando uma Arquitetura Orientada a Serviços?

   R – Essa questão é respondida com o estudo de caso executado nas Seções 4.4 e 4.5 com as quais foi possível executar processos de pesquisa e inclusão utilizando SOA com *Web Services*.

c. É factível que a interoperabilidade entre sistemas de RV seja alcançada com disponibilização destes componentes utilizando *Web Services*?

   R – Essa questão é respondida com a execução dos experimentos da Seção 4.4 e 4.5.

O trabalho cuja ideia e contribuição estão mais próximo do apresentado é o de Freiberger (2014) no qual o autor apresenta um modelo conceitual para instanciação de RV, o trabalho do referido autor gerou uma plataforma intitulada de VRServices (FREIBERGER, 2014), na qual é possível gerar mundos através da invocação de serviços com *Web Services* bem como exibi-los. A utilização de serviços provê interoperabilidade à ferramenta, ou seja, é possível acessar essas operações independente de tecnologia. Por outro lado, o cliente fica vinculado à plataforma e ao modelo de objetos do VRservices (FREIBERGER, 2014). Uma vez que, para a execução dos mundos o cliente deve utilizar a plataforma do mesmo. Os **OV** dessa plataforma são abstrações de cenas e objetos abstraídos para Java (ORACLE, 2015).

Diferente da proposta de Freibeger (2014), a arquitetura de referência proposta nesse

trabalho é um modelo que pode ser utilizado como base para o desenvolvimento de ferramentas de RV tais como a própria VRservices (FREIBERGER, 2014), uma das limitações levantados por Freiberger (2014, p152) no seu trabalho foi "[...] incorporar um ESB na plataforma VRServices para permitir a replicação de servidores de serviços, permitindo a redundância de serviços e o balanceamento de carga entre os servidores [...]". Limitação essa que é resolvida na **AR** deste trabalho, com o barramento ESB da **AR** intitulado como **VirtualCoreESB (**Ver Figuras 4.6,4.7 e 4.17**)**. Limitações como essa, levantadas por Freibeger (2014) poderiam ter sido evitadas no planejamento inicial, caso tivesse existido uma arquitetura de referência antes do desenvolvimento do projeto. No repositório implementado na Seção 4.4, existe um barramento no qual são expostos os serviços disponíveis**.**

O trabalho de Freibeger (2014) e o de Souza (2014) tratam diretamente o tema reuso, provendo alternativas para tal em RV, no entanto, existem limitações que amarram as suas soluções as suas plataformas e ferramenta. Situação que não ocorre com a **AR** desse trabalho, pois a mesma é um modelo conceitual e de implementação que pode ser utilizado como base inicial no desenvolvimento para Reuso em RV.

Tais técnicas, apesar de parecerem semelhantes à de sistemas de armazenamento de dados como o DropBox[59] e Google Drive[60] tem como diferencial a forma como os objetos virtuais são disponibilizados. Nos sistemas de armazenamento citados os arquivos são apenas salvos e compartilhados, enquanto que, nos modelos propostos neste trabalho é possível persistir metadados, incluir classificações, identificadores e categorização no nível de banco de dados. Esses dados podem ser utilizados no processo de seleção, filtragem e recuperação de AV. Assim, os arquivos salvos no DropBox[61] e Google Drive[62] não possuem formas de classificação que permitam acesso, filtragem e classificação personalizadas ou a realização de buscas programaticamente como acontece no modelo proposto neste trabalho.

Por ser um tema pouco explorado "Reuso em RV", existem poucos trabalhos que tratam de tal assunto, no entanto, modelos como os de Freiberger (2014) e Souza (2014) abordados neste trabalho representam uma contribuição na forma de se planejar o

---

[59] Dropbox – Ferramenta que permite armazenar arquivos e acessá-los através da internet. Disponível em < www.dropbox.com/>. Acesso em: 20 mai. 2015

[60] Google Drive – Outra ferramenta que permite armazenar arquivos e acessá-los através da internet. Disponível em <www.google.com/Drive> Acesso em: 20 mai. 2015

[61] Dropbox – Ferramenta que permite armazenar arquivos e acessá-los através da internet. Disponível em < www.dropbox.com/>. Acesso em: 20 mai. 2015

[62] Google Drive – Outra ferramenta que permite armazenar arquivos e acessá-los através da internet. Disponível em <www.google.com/Drive> Acesso em: 20 mai. 2015

desenvolvimento de infraestruturas para reuso em RV utilizando SOA. A **AR** apresentada neste trabalho é um modelo genérico cuja aplicação foi realizada para **RV**, tendo como base o **MR** e o protocolo definidos na Seção 4.1.1.

Todos esses trabalhos, incluindo essa tese são contribuições para Engenharia de Software no que diz respeito a modelos de alto nível para reuso no domínio dos sistemas em RV. Todas as questões tiveram respostas favoráveis e foram efetivas no sentido de que realizaram a comunicação e a persistência dos **OV** com base no modelo proposto. Entretanto, algumas limitações foram encontradas e são expostas na próxima seção.

5.2 Limitações

O projeto desta tese foi desenvolvido com o objetivo de tratar o gerenciamento e disponibilização de AV com Reuso de Software e SOA para qualquer tipo de recurso de RV não imersivo, porém os testes foram realizados apenas com os três elencados neste trabalho: X3D, X3DOM, WebGL. Assim, há a necessidade de se conduzirem novas investigações utilizando outros formatos, tais como o Unity3D, AngularJS e o Treejs, que trazem uma abordagem de desenvolvimento que simplifica a construção de **OV** 3D.

Também devem ser elencados como limitações dos estudos de caso:

- Tamanho dos ambientes e componentes testados.
- Aplicação do modelo para sistemas de RA.

5.3 Trabalhos futuros

O trabalho desta tese adiciona à literatura uma proposta diferente dos modelos existentes (FREIBERGER, 2014; SOUZA, 2014), uma vez que enfatiza que uma forma sistemática de reuso em AV é mais abrangente se direcionada para consumo e distribuição de mundos, sem focar na tecnologia utilizada para representar os AV em 3D.

Durante a investigação realizada, verificou-se que algumas formas de reuso bastante pesquisadas poderiam ser exploradas em trabalhos futuros e poderiam fazer utilização da arquitetura desse trabalho, por exemplo:

- Linhas de Produto de Software aplicadas a RV e RA em nuvem
    - Definição de modelo de *Features* para RV e RA
    - Definição de modelos de domínio e processos da engenharia de requisitos para esse domínio de aplicações.

5.3.1 Uma linha de produtos de software aplicadas a RV e RA utilizando SOA

Linhas de Produtos de Software (LPS) consistem em uma coleção de sistemas que compartilham e gerenciam um conjunto de características comuns, satisfazendo as necessidades de um cliente ou domínio específico (SEI, 2014).

A LPS é composta por um conjunto de artefatos que podem ser aplicados a um domínio específico que compartilham entre si um conjunto de funcionalidades e características controladas e gerenciáveis, exceto por algumas variações nos artefatos deste conjunto, que os tornam distinguíveis dos demais (CLEMENTS; NORTHROP, 2005; POHL et al., 2005; PEREIRA, 2011). Ainda conforme Weiss e Lai (1999), uma LPS é um processo projetado para se obter vantagens das características comuns e das variabilidades previsíveis de uma família de produtos.

5.4 Produção Bibliográfica

A produção científica desenvolvida durante a Tese pode ser aferida com a publicação de um Livro, resumos publicados em anais de encontro científico, um TCC e uma dissertação de Mestrado:

- **Título**: Reuse approaches and perspectives in VR and AR in education
    - **Aceito** como capítulo de livro em: Handbook of Research on 3-D Virtual Environments and Hypermedia for Ubiquitous Learning[63].
- **Título**: Arquitetura de Referência para Gerenciamento de Ambientes Virtuais 3D Uma abordagem baseada em Reuso, SOA e Computação em Nuvem para compartilhamento e distribuição de componentes 3D.
    - Livro publicado pela OmniScriptum GmbH & Co. KG com ISBN: 978-3-639-74924-3 no ano de 2015;
- **Título**: Desenvolvimento de componentes virtuais adaptativos
    - Trabalhos publicados e discutidos nos anais de dois (2) encontros de pesquisa e Extensão (UNIPE, 2012; UNIPE,2013);
- **Título**: Utilização de Web Service Para Compartilhamento de Conteúdo Virtual 3D Interativo na Web.

[63] IGI Global - Handbook of Research on 3-D Virtual Environments and Hypermedia for Ubiquitous Learning. Disponível em <http://www.igi-global.com/publish/call-for-papers/call-details/1558>. Acesso em 05 jun. 2015.

    - Trabalhos de conclusão de curso apresentado e aprovado no ano 2014 na UNIPÊ;
- **Título**: GSProject : Um jogo sério voltado para o gerenciamento de projetos
- Dissertação de mestrado aprovada e defendida no programa de Pós-graduação em informática da UFPB no ano de 2013;

Os trabalhos a seguir foram submetidos para publicação:

- **Título**: MR RVR – Um modelo de referência para Reuso em RV.
    - Submetido ao InterScientia 2015;
- **Título**: SOAVR – Uma Arquitetura para compartilhamento AV com RV não imersiva em Nuvem.
    - Será submetido para o SEARIS 2016;

Pretende-se elaborar *papers* durante o ano para submissão no SEARIS, evento do IEEE VR destinado a ES em RV e RA.

REFERÊNCIAS

ALEXANDER, C; ISHIKAWA, S; SILVERSTEIN, M. A. **Pattern Language**. Oxford University Press, 1 edition, 1977.

ABULRUB, A.; ATTRIDGE, A.; WILLIAMS, M. **Virtual Reality in Engineering Education:** The Future of Creative Learning. Global Engineering Education Conference: Learning Enviroments and Ecosystems in Engineering Education, IEEE, Amman, Jordan, 2011.

AFFONSO, F. J. **Metodologia para Desenvolvimento de Software Reconfigurável Apoiada por Ferramentas de Implementação:** Uma Aplicação em Ambiente de Execução Distribuído e Reconfigurável. Tese de Doutorado. São Carlos, Universidade de São Pualo (USP). 2008

ALLARD,J; GOURANTON,V; LECOINTRE, L; LIMET,S; MELIN,E; RAFFIN, B; ROBERT, S. FlowVR. **A middleware for large scale virtual reality applications. Euro-Par** – Parallel Processing, M. Danelutto, M. Vanneschi, and D. Laforenza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, vol. 3149, pp. 497–505. [Online]. 2004. Disponível em: http://www.springerlink.com/content/4bfattp806j5x9r0/. Acesso em: nov. 2014.

ALMEIDA, E. S., ALVARO, A., GARCIA, V. C., MASCENA, J. C. C. P., BURÉGIO, V. A. A., NASCIMENTO, L. M., LUCRÉDIO, D. AND MEIRA, S. R. L. CRUiSE. **Component Reuse in Software Engineering**, Centro de Estudos e Sistemas Avançados do Recife (C.E.SA.R), C.E.S.A.R e-book, 2007.

AMAZON. **O que é a computação em nuvem**? Disponível em: <http://aws.amazon.com/pt/what-is-cloud-computing/>. Acesso em: fev. 2015.

ANASTASSAKIS, G.; RITCHING, T.; PANAYIOTOPOULOS, T. **Multi-agent Systems as Intelligent Virtual Environments**. LNAI 2174, pp. 381-365, 2001.

ANSI/IEEE. **Recommended Practice for Architectural Description of Software** – Intensive Systems. IEEE Computer Society. ANSI/IEEE Std 1471-2000. DOI:

10.1109/IEEESTD.2000.91944. 2000.

ANTTONEN, M. et al. **Transforming the web into a real application platform: new technologies, emerging trends and missing pieces**. Proceedings of the 2011 ACM Symposium on Applied Computing. New York, NY, USA: ACM, 2011. (SAC '11), p. 800–807. ISBN 978-1-4503-0113-8. Disponível em: <http://doi.acm.org.ez22.periodicos.capes.gov.br/10.1145/1982185.1982357>. Acesso em Acesso em: 05 dez. 2014.

AQUINO, M. S. VEPersonal – **Uma Infra-estrutura para Geração e Manutenção de Ambientes Virtuais Adaptativos**. Recife, 180f. Tese de Doutorado. Universidade Federal de Pernambuco-UFPE, 2007.

AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C. **A Multi-Agent Architecture for Generating and Monitoring Adaptive Virtual Environments**. 5th International Conference on Hybrid Intelligent Systems – HIS'05, Rio de Janeiro-RJ, Brazil, Nov 06-09, p.515-517. Publisher: IEEE Computer Society, Washington, DC, USA, 2005.

AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C; SOUZA, D. A. C. M.; FUJIOKA, R. C. Supporting Adaptive Virtual Environments With Intelligent Agents. 7th International Conference On Intelligent Systems Design And Applications, Isda'07, Rio De Janeiro-Rj, Brazil, October, 22-24, 2007.

AVRADINIS, N.; VOSINAKIS, S.; PANAYIOTOPOULOS, T. **Using Virtual Reality Techniques for the Simulation of Physics Experiments** . Proceedings of the 4th Systemics, Cybernetics and Informatics International Conference, Florida, USA, 2000.

AZUMA, R. A Survey of Augmented Reality, em: **Teleoperators and Virtual Environments**. v .6, n.4, August, p. 355-385, 1997.

BAGLIONI, R.; SOUZA, R. C. G. **Engenharia web:** critérios de qualidade para o desenvolvimento e avaliação de sistemas web. Simpósio Brasileiro de Engenharia de Software, 2007, João Pessoa. Workshop de Teses e Dissertações em Engenharia de Software, 2007.

BARBOSA, A. E., CASELLA, B. F., BREGA, J. R. F. **Museus:** sistemas de informação para uma realidade virtual. XIII Encontro Nacional de Pesquisa em Ciência da Informação – XIII ENANCIB. 2012.

BARBOSA, G. M.G. **Um Livro-texto para o Ensino de Projeto de Arquitetura de Software.** (Dissertação de Mestrado) Campina Grande: Pós-Graduação em Ciência da Computação da UFCG, 2009.

BARBOSA, R. B; PORTO, R. M. A. B; MARTINS, C. E. M. A. **Uma proposta de desenvolvimento de aplicações de realidade virtual baseada em webservices**. Workshop De Realidade Virtual e Aumentada -WRVA 1–6. 2012.

BARESI, L.; NITTO, E. D.; GHEZZI, C. **Towards open-world software:** Issue and challenges. Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop (SEW ’06), Washington, DC, USA: IEEE Computer Society. 2006.

BARILL, E. C.; CUNHA, G.G. **A Tecnologia de Realidade Virtual:** Recurso Real Para Potencializar A Educação. Journal Virtual Reality JVR. Volume 2, P1-16. 2009.

BARILLI, E.C.V.; SANTOS H.; CASTIEL, L.D.; TELLES, F.; SABBATINI, R.E. **Internet para Profissionais de Saúde**. Material Didático do Curso a Distância. Escola Nacional de Saúde Pública, Fundação Oswaldo Cruz, 2003.

BARRETO, A., MURTA, L., ROCHA, A. R. **Componentizando Processos de Software Legados visando a Reutilização de Processos**. In VIII Simpósio Brasileirode Qualidade de Software, SBQS’09, Ouro Preto, Brasil. 2009.

BARROS, P. G. **Realidade virtual e multimídia**. Disponível em: <http://www.di.ufpe.br/~if124/vrml/vrml.htm>. Acesso em: 05 dez. 2014.

BASS, L. **Software architecture in practice**. Pearson Education India, 2007.

BASS, P.; CLEMENTS, L.; KAZMAN, R. **Software Architecture in Practice**.

Boston: Addison-Wesley, Second Edition, 2002.

BAUER,M; BRUEGGE , B; KLINKER , G; MACWILLIAMS , A; REICHER, T; RISS , S; SANDOR , C; WAGNER , M. **Design of a component-based augmented reality framework**. IEEE and ACM International Symposium on Augmented Reality, 2001. Proceedings. IEEE, 2001, pp. 45–54.

BEHR, J., ESCHLER, P., JUNG, Y., AND ZÖLLNER. **M. X3DOM** – a DOM-based HTML5/ X3D integration model. Proceedings. Web3D'09, ACMPress, Nova York, USA, 127–135. 2009.

BELL, M. **Service-Oriented Modeling**. John Wiley e Sons, 366 p. ISBN 9780470141113, 2008.

BHAKTI, M.; ABDULLAH, A. **Design of an autonomic services oriented architecture.** Information Technology (ITSim), 2010 International Symposium. 2010.

BOLDYREFF, C.; NUTTER, D.; RANK, S. **Open-Source Artifact Management.** Workshop Open Source Software Engineering, USA, 2002.

BRAGA, R. T. V. **Um Processo para Construção e Instanciação de Frameworks baseados em uma Linguagem de Padrões para um Domínio Específico**. (Tese de Doutorado). Instituto de Ciências Matemáticas e de Computação da USP-SC, São Carlos, São Paulo, Brasil, 2003.

BRAGA, R. T. V.; GERMANO, F. S. R.; MASIERO, P. C. **A Pattern Language for Business Resource Management**. Proceedings of the 6th Annual Conference on Pattern Languages of Programs (PLoP'99). Monticello, Illinois, USA, 1999.

BRUGALI, D.; SYCARA, K. **Frameworks and Pattern Languages:** An Intriguing Relationship. ACM Computing Surveys, ACM Press, v. 32, n. 1, p. 2-7, 1999.

BRUTZMAN, D., DALY, L. X3D: **3D Graphics for Web Authors**. Morgan Kaufmann Publishers. 2007.

BURDEA, G., COIFFET, P. **Virtual RealityTechnology**. John Wiley & Sons. Cruz-Neira, 1994.

BURGARELI, L. A. Gerenciamento de Variabilidade de Linha de Produtos de Software com Utilização de Objetos Adaptáveis e Reflexão. São Paulo, 247f. Tese de Doutorado. Universidade de São Paulo-USP. 2009.

BUSCHMANN, F.; HENNEY, K.; SCHMIDT, D. C. **Pattern-oriented software architecture:** on patterns and pattern languages. Indianapolis: Willey 5 v, 2007.

CANTOR, D.; JONES, B. **WebGL Beginner's Guide.** Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing, 2012.

CAVAZZA, M.; CHARLES, F.; MEAD, S.; STRACHAN, A. **Virtual Actors' Behaviour for 3D Interactive Storytelling.** Proceedings of the Eurographics Conference, 2001.

CELENTANO, A.; NODARI, M.; PITTARELLO, F. **Adaptive Interaction in Web3D Virtual Worlds**. 9th International Conference on 3DWeb Technology, Monterey, California, USA, Abril 5-8, 2004.

CERN. **VRML**. Disponível em: <http://www94.web.cern.ch/WWW94/>. Acesso em: 01 ago. 2014.

CERVO, A. L. e BERVIAN, P. A. **Metodologia Científica**. São Paulo: Prentice Hall., 5a. Edição. 2002

CHAMOVITZ, I.; SABBADINI, F.S.; OLIVEIRA, M.J.F. A utilização da simulação baseada na Web para o estudo de processos operacionais. **Revista Produção Online**. Vol.8 n.4 Dez. 2008. Disponível em: <http://producaoonline.org.br/index.php/rpo/article/viewFile/148/235>. Acesso em: 01 ago. 2014.

CHEESMAN, J.; DANIELS, J. UML Components – **A Simple Process for Specifying Component-based**. Addison-Wesley, 2001.

CHITTARO, L.; RANON, R. **New Directions for the Design of Virtual Reality Interfaces to e-Commerce Sites**. Proceedings Of Avi 2002: 5th International Conference On Advanced Visual Interfaces, Acm Press, New York, Pp. 308-315. 2002.

CHITTARO, R.; RANON, R.; Ieronutti, L. **Guiding Visitors of Web3D Worlds through Automatically Generated Tours.** Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology, ACM Press, New York, pp. 27-38, Mar 2003.

CLEMENTS, P; NORTHROP, L. **Software Product Lines** – Practices and Patterns. vol. 1, 3rd ed: Addison-Wesley, 2005.

COALLIER, F. **A Vision for International Standardization in Software and Systems Engineering.** 6th International Conference on the Quality of Information and Communications Technology. Universidade Nova de Lisboa. Lisboa, Portugal, 2007.

COSTA, P. A. Uma ferramenta para análise automática de modelos de características de linhas de produtos de software sensível ao contexto. Fortaleza, 311f. Dissertação de mestrado. Universidade Federal do Ceará. 2012.

CRNKOVIC, I.; HNICH, B.; JONSSON, T.; KIZILTAN, Z. **Specification, implementation, and deployment of components.** Communications of the ACM, ACM Press, 2002.

CZARNECKI, K., ANTKIEWICZ, M. KIM, C.H.P., LAU, S., PIETROSZEK, K**. Model-Driven Software Product Lines**. ACM SIGPLAN - Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA) – Poster Session. pp.126-127. 2005.

CZARNECKI, K.; EISENERCKER, U. W. **Generative Programming**. Addison-Wesley, 2002.

D'SOUZA, D. F., WILLS, A. C. Objects, Components and Frameworks with UML, The Cataliysis Approach. Addison-Wesley. USA.1999.

DALMON, D. L. ; BRANDÃO, A. A. F; ISOTANI, S; GOMES, G. M. ; BRANDÃO, L. O. **Work in progress** – a generic model for interactivity-intense intelligent tutor authoring tools. Proceedings of 42nd Fron- tiers in Education Conference. 2012.

DEGROSSI, L. C. Uma abordagem para obtenção e disponibilização em tempo real de informações geográficas voluntárias no contexto de gestão de risco de inundação. Dissertação de Mestrado. Universidade de São Paulo (USP).São Carlos, USP, 73f, 2014.

DIDIER, J.; CHOUITEN, M. ; MALLEM, M. ; OTMANE, S. ARCS: **A framework with extended software integration capabilities to build Augmented Reality applications.** Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 5th Workshop. Orange County, CA. Doi: 10.1109/SEARIS.2012.6231170. 2012.

DIGITALOCEAN. **DigitalOcean**. Disponível em: < https://www.digitalocean.com>. Acesso em: mai. 2015.

DIGIAMPIETRI, L. A; ARAÚJO,J. C; ERIC, C. R. N. S; OSTROSKI, H. Combinando workflows e semântica para facilitar o reuso de software. **Revista de Informática Teórica e Aplicada (RITA).** ISSN 2175-2745. 2013.

DONEGAN, P. M. Geração de famílias de produtos de software com arquitetura baseada em componentes. Dissertação de Mestrado, Universidade de São Paulo. 2007.

ECLIPSE. **Eclipse IDE**. Disponível em <http://www.eclipse.org>. Acesso em: 18 ago. 2014.

ENDREI, M.; ANG, J.; ARSANJANI, A.; CHUA, S.; COMTE, P.; KROGDAHL, P.; LUO, M.; NEWLING, T. **Patterns:** Service-oriented architecture and *Web Services*. IBM Redbooks, disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf>. Acesso em: 15 dez. 2014.

ERL, T. **SOA Princípios de Design de Serviços.** 1ª Ed. São Paulo. Pearson Prentice Hall, 322f, 2009.

ERRADI, A.; MAHESHWARI, O. **A broker-based approach for improving *Web Services* reliability.** IEEE International Conference on *Web Services* (ICWS'05), 2005.

FALCÃO, E.L.; MACHADO, L.S.; COSTA, T.K.L. **Programando em X3D para Integração de Aplicações e Suporte Multiplataforma.** Book Chapter. In: Machado, L.S.; Siscoutto, R.A. (Org.) Tendências e Técnicas em Realidade Virtual e Aumentada, Chapter 2, p. 35-63. SBC. 2010. Disponível em: <http://www.de.ufpb.br/~labteve/publi/2010_svrmc.pdf>. Acesso em: 01 ago. 2014.

FARKAS, P.; CHARAF, H. *Web Services* planning concepts. **Journal Of .net Technologies**, p. 9–12, 2003.

FAYAD M. E.; JOHNSON, R. E.; SCHMIDT, D. C. **Software Patterns**. Communications of the ACM, ACM Press, v. 39, n. 10, p. 37-39, 1996.

FAYAD, M. E.; JOHNSON, R. E.; SCHMIDT, D. C. **Building Application Frameworks:** Object-Oriented Foundations of Framework Design. Wiley & Sons, 1999.

FAYAD, M. E.; SCHMIDT, D. C. **Object-Oriented Application Frameworks.** Communications of the ACM, ACM Press, New York, NY, USA, v. 40, n. 10, p. 32-38, 1997.

FERREIRA, M. V. A. S. Framework com as contribuições da convergência digital possibilitada pela utilização das tecnologias interativas da TV digital, associadas ao uso dos dispositivos móveis digitais, para a evolução do modelo brasileiro de governo eletrônico. Florianópolis, 430f. Tese de Doutorado. Universidade Federal de Santa Catarina - UFSC, 2013.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, 2000.

FILHO, R. F. A. Hydra: **Arquitetura e Framework para Desenvolvimento de**

Ambientes Virtuais. Dissertação de mestrado, 91f. Universidade Federal de Pernambuco (UFPE). 2011.

FIRESMITH, D.G.; Frameworks: the golden path to object Nirvana. Journal of Object-Oriented Programming, vol. 7, nº 8, 1994.

FOWLER, M. **Inversion of Control Containers and the Dependency Injection pattern**. 2004: Artigo disponível em: <http://martinfowler.com/articles/injection.html>. Acesso em: 07 Jan. 2015.

FOWLER, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

FOWLER, M.: **How standard is Standard UML 1999**. Disponível em: <http://martinfowler.com/distributedComputing/standard.html >. Acesso em: 07 Jan. 2014.

FRAKES, W.; KANG, K. **Software Reuse Research**: Status and Future. IEEE Transactions on Software Engineering, Vol.31, Nº 7, July, 2005.

FRAKES, W.; TERRY, C. **ACM Computing Surveys (CSUR), ACM Press**, v. 28, n. 2, p. 415–435, out. 1996. Software reuse: metrics and models.

FRANCA, L. P. A.; STAA, A. V. **Geradores de Artefatos:** Implementação e Instanciação de Frameworks. Anais do XV SBES-2001 – Simpósio Brasileiro de Engenharia de Software. Rio de Janeiro, Brasil, 2001.

FRANCO, F.F; FRANCO, N.F; CRUZ, S.R.R; LOPES, R.D. Experiências de uso de Mídias Interativas como Suporte para Autoria e Construção Colaborativa do Conhecimento. Novas Tecnologias na Educação. CINTED-UFRGS, 2007.

FREIBERGER, E. C; NAKAMURA, R.; NUNES, F. L. S. **Abordagens de reuso em aplicações de realidade virtual e aumentada**. IX Workshop de Realidade Virtual e Aumentada. 2012.

FREIBERGER,E. C; NAKAMURA,RICARDO; TORI, R. **Service-Oriented**

Platform for Reuse of Interactive Content of Virtual Reality Applications. IJEEEE 2014 Vol.4(3): 224-231 ISSN: 2010-3654 DOI: 10.7763/IJEEEE.2014.V4.335. 2014.

FREIBERGER,E. C. **Modelo conceitual para instanciação de aplicações de realidade virtual em uma plataforma baseada em serviços**. São Paulo, 176f. Tese de Doutorado – Escola Politécnica Da Universidade De São Paulo (USP), 2014.

FUGITA, H. S.; HIRAMA, K. **SOA – Modelagem, Análise e Design**. 1° Edição. Elsevier. Editora Campus. 176f, 2012.

FUJIOKA, R.C. **Arquitetura de referência para gerenciamento de ambientes virtuais 3D:** uma abordagem baseada em Reuso, SOA e Computação em Nuvem para compartilhamento e distribuição de componentes 1d. OMNISCRIPTUM GMBH & CO. KG. ISBN: 978-3-639-74924-3. 2015.

FUJIOKA, R.C. **SCOLLAB CORE:** a proposta de um framework para construção de aplicações sociais. João Pessoa, 111f. Dissertação de Mestrado – Universidade Federal da Paraíba (UFPB), 2011.

FUJIOKA, R.C. Um modelo baseado em sistemas multi-agentes para monitoramento de ambientes virtuais tridimensionais: um modelo baseado em sistemas multi-agentes para monitoramento de ambientes virtuais tridimensionais. Monografia. Centro Universitário de João Pessoa (UNIPÊ), 2008.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. **Padrões de Projeto:** Soluções Reutilizáveis de Software Orientado a Objetos. Bookman, 2000.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns** – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

GARLAN, D., & PERRY, D. **Software architecture:** practice, potential, and pitfalls. In Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on, pp. 363-364, IEEE, 1994.

GARLAN, D.; SHAW, M. **An introduction to software architecture**. Ambriola & Tortola (eds.), Advances in Software Engineering & Knowledge Engineering, v. II, World Scientific Pub Co., Singapore, 1993, p.1–39.

GEOVRML. **GeoVRML**. Disponível em: <http://www.ai.sri.com/geovrml/>. Acesso em: 01 ago. 2014.

GGLAS. **Google Glass.** Disponível em: < https://www.google.com/glass/start/>. Acesso em: 24 Jan 2015.

GILL, N. S. Importance of Software Component Characterization for Better Software Reusability. ACM SIGSOFT Software Engineering Notes, Vol 31, No 1, Julho 2006.

GIMENES, I. M. S, & HUZITA, E.H.M. **Desenvolvimento Baseado em Componentes:** Conceitos e Técnicas. Rio de Janeiro: Ciência Moderna, 2005.

GINGA. **Middleware Aberto do Sistema Nipo-Brasileiro de TV Digital**. Disponível em: <http://www.ginga.org.br/>. Acesso em: 20 Out 2014.

GOAER, O.; TAMZALIT, D.; OUSSALAH, M. C.; SERIAI, A. D. **Evolution styles to the rescue of architectural evolution knowledge.** Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge, Leipzig, Germany, 2008.

GOMAA, H. **Designing software product lines with UML**: from use cases to pattern-based software architectures. Addison-Wesley, 2004.

GONÇALVES, A. **Reconstrução de ambientes históricos utilizando VRML:** o caso do Fórum Flaviano de Conimbriga. Dissertação de mestrado em Eng. Informática. Universidade de Coimbra. 2002.

GONZAGA, R. S. **Uma abordagem arquitetural para a orquestração dinâmica de serviços**. Santo André, 48p. Dissertação de Mestrado – Universidade Federal do ABC (UFABC), 2014.

GOOGLE. **Google Chrome**. Disponível em: <http://www.google.com/intl/pt-BR/chrome/browser/>. Acesso em: 01 ago. 2014.

GOOGLECODE. **Quake II and the Quake logo are trademarks of id Software.** Disponível em: <https://code.google.com/p/quake2-gwt-port/>. Acesso em: 17 out. 2014.

GOUVEIA, V.A.G. **Aplicação de uma linha de produtos de software (LPS) no contexto de uma PME.** Funchal, 195f. Dissertação de Mestrado – Universidade Técnica de Lisboa, 2007.

GREENFIELD, J.; SHORT, K. **Software factories** – Assembling applications with patterns, models, frameworks, and tools. Wiley Publishing, Inc, 2004.

GRISS, M.L., FAVARO, J., D'ALESSANDRO, M. **Integrating feature modeling with the RSEB**. Proceedings of the fifth International Conference on Software Reuse – ICSR5, pp. 76-85, Victoria, British Columbia, Canada. 1998.

GUERRA, P. A. G. Uma abordagem arquitetural para tolerância a falhas em sistemas de software baseados em componentes. Tese de Doutorado. Instituto de Computação – UNICAMP, 2004.

GUIMARÃES, M. P; MARTINS, V.F. Desafios a serem superados para o uso de Realidade Virtual e Aumentada no cotidiano do ensino. **Revista de Informática Aplicada**, V9, n°1. 2013.

GUO, J., LUQI. **A. Survey of Software Reuse Repositories**. Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems. Edinburgh, Scotland, UK, pp. 92-100, 2000.

HEINZLE, R.; MONTIBELER, J. P.. HOGREFE, L. C. **Estudo de caso:** Uma ferramenta para a construção amigável de ontologias para sistemas baseados em conhecimento. Revista Interdisciplinar Científica Aplicada, Blumenau, v.7, n.1, p.1- 14, TRI I. 2013. ISSN 1980-7031 1.

HENNINGER, S. **An Evolutionary Approach to Construction Effective Software Reuse Repositories.** ACM Transactions on Software Engineering and Methodologies, v.16, n.2, pp. 111-138, 1997.

INFO. Revista INFO Exame. Disponível em: <http://info.abril.com.br/noticias/ciencia/realidade-virtual-ajuda-a-aliviar-dor-fisica-15092009-29.shl>. Acesso em: nov. 2014.

INOUE, K.; YOKOMORI, R.; FUJIWARA, H.; YAMAMOTO, T.; MATSUSHITA, M.; KUSUMOTO, S. **Component Rank:** Relative Significance Rank for Software Component Search. International Conference on Software Engineering, 2003.

JANSEN, W.; GRANDE, T. **Guidelines on security and privacy in public cloud computing**. NIST special publication, pp. 800-144, 2011.

JAVA. **O que é Java.** Disponível em: <https://www.java.com/pt_BR/download/whatis_java.jsp> Acesso em: 01 ago. 2014.

JAVA3D. **Java 3D**. Disponível em <https://java3d.java.net/>. Acesso em: 20 jul. 2014.

JOHNSON, R. E. **Documenting Frameworks using patterns.** Conference proceedings on Object-oriented programming systems, languages, and applications. Vancouver, British Columbia, Canada: ACM Press, 1992.

JOHNSON, R. E. **Frameworks = (Components + Patterns)**. Communications of the ACM, vol. 40 n. 10, 1997.

JOHNSON, R. E.; FOOTE, B**. Designing reusable classes.** Journal of Object Oriented Program- ming, v. 1, n. 2, p. 22–35, 1988.

JOSUTTIS, N. M. **SOA na prática** – A Arte da Modelagem de Sistemas Distribuídos. 1st. ed. Oreill'y, 2008.

JUNIOR, A. A. O. **Visualizador 3D para dados de Radar Meteorolóogico Usando Webgl.** Dissertação de mestrado. Universidade Federal do Paraná (UFPR) , 133f. 2012.

KHRONOS. **WebGL** – OpenGL ES 2.0 for the Web. Disponível em: <https://www.khronos.org/webgl/>. Acesso em: 01 ago. 2014.

KICZALES, G.; LAMPING, J.; MEDDHEKAR, A.; MAEDA, A.; LOPES, C. V.; LOINGTIER, J. M.; IRWIN, J. **Aspect-Oriented Programming. Proceedings of the European Conference on Object-Oriented Programming** (ECOOP), Finland. Springer-Verlag LNCS, 1997.

KILNER, J. **Realidade Virtual**. Disponível em: < http://www.cin.ufpe.br/~if687/frame/turmas/turma_2011_1/aula_30_05_2011.pdf>. Acesso em: 01 jul. 2014.

KIRNER, C. **Prototipagem Rápida de Aplicações Interativas de Realidade Aumentada**. Tendências e Técnicas em Realidade Virtual e Aumentada, v. 1, p. 29-54, 2011.

KIRNER, C., PINHO, M.S. **Introdução a Realidade Virtual.** Mini-Curso, JAI/SBC, Recife, PE, 1996.

KIRNER, C.; KIRNER, T. T. **Evolução e Tendências da Realidade Virtual e da Realidade Aumentada**. Realidade Virtual e Aumentada: Aplicações e Tendências. v. 1, p. 8-23, 2011.

KIRNER, C.; SISCOUTTO, R. **Fundamentos de Realidade Virtual e Aumentada.** Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações. Eds. Cláudio Kirner e Robson Siscoutto. Livro do Pré-Simpósio, IX Symposium on Virtual And Augmented Reality, Petrópolis – RJ, 2007.

KITCHENHAM, B.; MENDES, E.; TRAVASSOS, G. H. **Cross Versus Within-Company Cost Estimations Studies:** A Systematic Review. IEEE Transactions on Software Engineering, v 3, n.5. 2007.

KRAFZIG, D.; BANKE, K.; SLAMA, D. **Enterprise Soa:** Service-Oriented Architecture Best Practices. The coad series. Prentice hall, 408 P. 2004.

KRUEGER, C. W. **Software reuse.** ACM Computing Surveys (CSUR), pp.131-183, 1992.

KUMAR, B. V.; NARAYAN, P., NG, T. **Implementando Soa Usando Java EE.** 1° Ed, Rio De Janeiro. Editora Altabooks, 368f, 2012.

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet:** Uma Abordagem Top- Down. 5ª ed. São Paulo, Brasil: Editora Pearson 592 p., 2010.

LA, H. J; KIM, S. D. **A conceptual framework for provisioning context-aware mobile cloud services.** IEEE 3rd International Conference on Cloud Computing (CLOUD). IEEE, Jul. 2010, pp. 466– 473. 2010.

LAGERGREN, M. **Modeling as a tool to assist in managing problems in health care**. D. Boldy, J. Braithwaite and I. Forbes (Eds.), Evidence Based Management in Health Care: The Role of Decision Support Systems, Australian Studies in Health Service Administration, 92, pp 17- 36, 2002.

LAU, K. **Software Component Model**. Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

LEITE, D.R. **GSPROJECTS** – Um ambiente para simulação da gestão de projetos de software. João Pessoa, 54f. Dissertação de Mestrado – Universidade Federal da Paraíba (UFPB), 2013.

LEMOS, M.O.O.; MACHADO, L.S. **Novas Tecnologias para o Desenvolvimento de Conteúdo 3D para Web**. Symposium on Virtual and Augmented Reality 2012. 4p. Niterói/RJ - Brazil. 2012. Disponível em: <http://www.de.ufpb.br/~labteve/publi/2012_svr4.pdf>. Acesso em: 01 ago. 2014.

LEWIS. G; WRAGE. L. **Model Problems in Technologies for Interoperability:**

Web Services. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note CMU/SEI-2006-TN-021, 2006. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7939.

LI , M; WANG, S; HE, T. **A transformer substation simulation engine based on virtual reality.** International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE), vol. 2. IEEE, Aug. 2010, pp. 41–44. 2010.

LI, M.; WANG, S.; HE, T. A transformer substation simulation engine based on virtual reality. In **Computer, Mechatronics, Control and Electronic Engineering** (CMCE), 2010 International Conference on, vol. 2, pp. 41-44, 2010.

LIGHTHOUSE3D. **Introduction to the Tutorial.** Disponível em <http://www.lighthouse3d.com/vrml/tutorial/> Acesso em: 01 ago. 2014.

LINDEN, F. J. D., ROMMES, E. e SCHMID, K**. Software Product Lines in Action.** Editora Springer. Primeira Edição. Lucrédio, 2007.

LLORA, X; ACS, B; AUVIL,L. S; CAPITANU, B; WELGE,M. E; GOLD- BERG, D. E. Meandre. **Semantic-driven data-intensive flows in the clouds.** IEEE Fourth International Conference on eScience. eScience ’08. IEEE, Dec. 2008, pp. 238–245. 2008.

MACKENZIE, C. M.; LASKEY, K.; MCCABE, F.; BROWN, P. F.; HAMILTON; METZ., R. **Oasis Reference Model For Service Oriented Architecture 1.0**. Disponível em: < http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html> 2006. Acesso em: 05 abr. 2015.

MARCHÃO, J; REIS, L. **Serviços em Cloud na Ótica de Utilização Empresarial.** Revista de Ciências da Computação. Portuguese Journal of Computer Science.V8, N8. ISSN: 2182-1801. 2013. Disponível em <http://inqueritos.lead.uab.pt/OJS/index.php/RCC/issue/view/9>. Acesso em: 10 ago. 2014.

MARINS, V., HAGUENAUER, C., CUNHA, G.G. **Realidade Virtual em Educação Criando Objetos de Aprendizagem com VRML.** Revista Colabora, Vol. 4. N. 15, ISSN. 1519-8529. 2007.

MARTINS, A. **Fundamentos de Computação Nuvem para Governo.** XXX Congresso da Sociedade Brasileira de Computação – SCBC, Workshop de Computação Aplicada em Governo Eletrônico (WCGE). 2010. Disponível em: <http://www4.serpro.gov.br/wcge2010/artigos-selecionados>. Acesso em: 12 ago. 2014.

MARTINS, V.; OLIVEIRA, A.; GUIMARÃES, M. **Implementação de um Laboratório de Realidade Virtual de Baixo Custo:** Estudo de Caso de Mon*tag*em de um Laboratório para o Ensino de Matemática. Revista Brasileira de Computação Aplicada, Rio Grande do Sul, Brasil, v. 5, n. 1, p. 98 - 112, abr. 2013.

MARZULLO, F. P. **SOA na prática** – Inovando seu negócio por meio de soluções orientadas a servicos. 1° ed. Novatec, 392f, 2009.

MASS, Y; HERZBERG, A. **VRCommerce** – Electronic Commerce in Virtual Reality. ACM Conference on Electronic Commerce. 1999.

MATTSSON, M.**Object-oriented Frameworks** – A survey. Tese de Doutorado. Departamento de Ciência da Computação. Lund University, CODEN:LUTEDX/(TECS-3066)/1-130/(1996), also as Technical Report, LU-CS-TR:96-167. 1996.

MCILROY, M. D. Mass Produced Software Components, In: **NATO Software Engineering Conference Report**, Garmisch, Germany, October, 1968, pp. 79-85. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acesso em: 26 jan. 2015.

MELL, P.; GRANCE, T. **The NIST definition of cloud computing.** NIST Special Publication, 2011.

MERRIAM, S. S. **Qualitative Research:** A Guide to Design and Implementation . Jossey-Bass Higher & Adult Education Series, 2009.

MEYER, B. **Object-Oriented Software Construction**. Prentice Hall, 2 edition, 1296f, 2000.

MICROSOFT. O que é um driver? . Disponível em: <http://windows.microsoft.com/pt-br/windows/what-is-driver#1TC=windows-7>. Acesso em: 05 ago. 2014.

MILES, R.; HAMILTON, K. **Learning UML 2.0**. O'Reilly Media, Inc. ISBN 0596009828. 2006.

.

MOZILLA. **Firefox**. Disponível em < https://www.mozilla.org/pt-BR/firefox/new/>. Acesso em: 01 ago. 2014.

MOREIRA, A. A. **Reuso de IHC orientado a padrões concretos de interação e dirigido por casos de uso.** 2007. 181 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.2007.

MOURA, D. S. **Software Profile RAS:** Estendendo a Padronização do Reusable Asset Specification e Construindo um Repositório de Ativos. Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR– RS. 2013.

MUELLER-WITTIG, W, JEGATHESE, R., SONG M., QUICK, J., WANG, H & ZHONG,Y. **Virtual Factory** – Highly Interactive Visualization for Manufacturing. Proceedings of the 2002 Winter Simulation Conference, 2002.

MUNSHI, A.; GINSBURG, D.; SHREINER, D. **OpenGL ES 2.0**. Boston, MA USA: Addison-Wesley, ISBN 978-0-321-50279-7. 2009.

MURAKAMI, E. **Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão**. Tese de Doutorado, 192f. Universidade de São Paulo (USP). 2006.

MUSSE, S. Human Crowd Modelling with Various Levels of Behaviour Control. Tese de Doutorado. Lausanne: EPFL. 2000.

NAKAMURA, L; VASCONCELOS, H. **Utilização de web semântica para seleção**

de informações no registro UDDI uma abordagem com qualidade de serviço. Dissertação de Mestrado. Universidade de São Paulo. 148f, São Carlos. 2011.

NASCER, J. F. R. **Titan Architect:** Um Wizard para Instanciação de Gerenciadores de Conteúdo utilizando o Titan Framework. Trabalho de Conclusão de Curso, 124f. Universidade Federal de Matro Grosso do Sul (UFMS). 2009.

NETO, A. R; CUNHA, G. G; LANDAU, L. **Desenvolvimento de Simuladores de Equipamentos Portuários Utilizando Realidade Virtual**. Journal Virtual Reality JVR. Volume 6, P1-16. 2013.

NIJHOLT, A.; HULSTIJN, J. **Multimodal Interactions with Agents in Virtual Worlds**. Kasabov, N. (ed.): Chapter 8 - Future Directions for Intelligent Information Systems and Information Science, Physica-Verlag: Studies in Fuzziness and Soft Computing, pp. 148-173, 2000.

OCULUS. **Oculus VR.** Disponível em: <https://www.oculus.com/>. Acesso em: jan. 2015.

OLIVEIRA , A. C. M. T. G; NUNES , F. D. L. D. S. **Building an Open Source framework for virtual medical training**. Journal of digital imaging : the official journal of the Society for Computer Applications in Radiology, vol. 23, no. 6, 2010.

OLIVEIRA, A. J. G; MARTINS, V. F. **Realidade Virtual:** Projeto de Baixo Custo para Criação de Ambientes Virtuais na Área Educacional. Trabalho de Conclusão de Curso. Universidade Presbiteriana Mackenzie. 2010.

OLIVEIRA, J. P. F.; BRITO. T; RABELO, S ; ELIAS, G . **Um Serviço de Repositório Compartilhado e Distribuído para Suporte ao Desenvolvimento Baseado em Componentes.** Simpósio Brasileiro de Engenharia de Software. ANAIS – XXI Simpósio Brasileiro de Engenharia de Software, João Pessoa , 2007.

OLIVEIRA, M. J; SABBADINI,F; CHAMOVITZ, I. **Uma plataforma de simulação visual em 3D orientada para o ciclo de vida das entidades.** Revista Eletrônica de

Engenharia de Produção e Correlatas – ISSN 1676-1901, 2009.

OLIVEIRA, M; CROWCROFT, J; SLATER, M. **An innovative design approach to build virtual environment systems**. Proceedings of the workshop on Virtual environments 2003, ser. EGVE ’03. Zurich, Switzerland: ACM, p. 143–151, ACM ID: 769970. 2003.

OPENGL. **Vertex Especifications**. Disponível em: <http://www.opengl.org/wiki/Vertex_Specification>. Acesso em: 20 jul. 2014.

PALAZZO, L. A. M. Modelos Proativos para Hipermídia Adaptativa. Tese de Doutorado. Instituto de Informática da UFRGS. Programa de Pós-graduação em Computação. Porto Alegre, 2000.

PAPAZOGLOU, M. P.; GEORGAKOPOULOS, D. **Service-oriented computing**. Communications of the ACM. October 2003/Vol. 46, No. 10, 2003.

PASLEY, J. **How BPEL and SOA are changing *Web Services* development.** IEEE Internet Computing, v.9, n.3, p.60-67. 2005.

PEREIRA, T.A.B. Uma Abordagem para Recomendação de Módulos para Projetos de Desenvolvimento Distribuído de Linhas de Produto de Software. João Pessoa, 115f. Dissertação de Mestrado – Universidade Federal da Paraíba (UFPB), 2011.

PESSIN, G., OSÓRIO, F. S., MUSSE, S. R. **Simulação virtual da evolução de estratégias e do controle inteligente em sistemas multi-robóticos**. In Anais do X SVR, 2008.

PIRES, P. A. R. Framework para a construção de “portais de negócio” para gestão de solicitações de consumidores IaaS na HP Cloud. Lisboa, Portugal. Dissertação de Mestrado - Universidade Nova Lisboa. 2013.

POHL, K., BÖCKLE, G.; LINDEN, F. J. Software product line engineering: foundations, principles and techniques. Nova York: Springer-Verlag, 2005.

POLYS, N. F.; VISAMSETTY, S. S. S.; TILEVICH , E. **Design Patterns in Componentized Scenegraphs.** Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 5th Workshop. Orange County, CA. Doi: 10.1109/SEARIS.2012.6231170. 2012.

PREE, W. Design Patterns for Object-Oriented Software Development. Addison-Wesley, 1995.

PRESSMAN, R. S. **Engenharia de Software:** uma abordagem profissional. 7ªed. Porto Alegre: Editora Bookman, 2011.

PROTA, T.M. **Moondo:** um framework para desenvolvimento de aplicações declarativas no sbtvd. Trabalho de Conclusão de Curso – Universidade Federal de Pernambuco, Centro de Informática. Recife, 2009.

PURE-SYSTEMS. **Get pure:** variants Community. Disponível em: <http://www.pure-systems.com/pure_variants_Community.55+M5dcda8b8b9c.0.html>. Acesso em: 18 ago. 2014.

QUEIROZ, P. G. G. **Uma abordagem de desenvolvimento de linha de produtos com uma arquitetura orientada a serviços.** São Carlos SP, 140f. Dissertação de Mestrado, Universidade de São Paulo-USP; 2009.

REICHER, T; WILLIAMS, M. A; BRUGGE, B; KLINKER, G. **Results of a study on software architectures for augmented reality systems.** The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings. IEEE, Oct. 2003, pp. 274– 275. 2003.

REYNOLDS, C. W. **Interaction with Groups of Autonomous Characters.** Proceedings of Game Developers Conference 2000, CMP Game Media Group, San Francisco, CA, pp. 449-460, 2001.

RED HAT. **OpenShift**. Disponível em: <https://www.openshift.com>. Acesso em: 20 mai. 2015.

RICKEL, J.; MARSELLA, S.; GRATCH, J.; HILL, R.; TRAUM, D.; SWARTOUT W. **Toward a New Generation of Virtual Humans for Interactive Experiences**. IEEE Intelligent Systems, vol. 17, no. 4, Special issue on AI in Interactive Entertainment, 2002.

RIZZO, A. A.; BOWERLY, T.; BUCKWALTER, J. G.; SCHULTHEIS, M.; MATHEIS, R.; SHAHABI, C.; Neumann, U.; Kim, L.; Sharifzadeh, M. **Virtual Environments for the Assessment of Attention and Memory Processes:** The Virtual Classroom and Office. Proceedings of the International Conference on Disability, Virtual Reality and Associated Technology, Vesaprem, Hungary, September, 2002.

ROCHA , D. C., FILHO, A. L. F., LEITE, D. A, VILELA, F. G., LIBARDI, P. O., SOUZA.A. F. L., LIMA, L. V. O., CALIXTO , L. G. **Desenvolvimento de uma plataforma auxiliar para ensino de máquinas elétricas empregando realidade virtual.** Anais do SBIE. 2013. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/2574>. Acesso em: 06 abr. 2015.

RODELLO, I. A. ; BREGA, J. R. F. Realidade Virtual e Aumentada em Ações de Marketing. In: RIBEIRO, M. W. S. ; ZORZAL, E. . (Org.). **Realidade Virtual e Aumentada: Aplicações e Tendências.** 1 ed. Uberlândia: Editora SBC - Sociedade Brasileira de Computação, 2011, v. 1, p. 44-57.

RODELLO,I. A.; SANCHES, S. R. R.; SEMENTILLE, A. C.; BREGA, J. R. F. Realidade Misturada: Conceitos, Ferramentas e Aplicações. **Revista Brasileira de Computação Aplicada**, v.2, n.2 , 2010. Disponível em:<http://www.upf.br/seer/index.php/rbca/article/view/941/776> Acesso em: 10 jun. 2014.

RUIZ, J. G. **The Impact of e-Learning in Medical Education.** Academic Medicine , 81 (3), 207-212. 2006.

SAMETINGER, J. **Software Engineering with Reusable Components**. Springer – Verlag New York, Inc, 1997.

SANTANA, F. S. **Uma Infraestrutura Orientada A Serviços Para A Modelagem**

De Nicho Ecológico. São Paulo, 141f. Tese De Doutorado - Escola Politécnica Da Universidade De São Paulo (USP), 2009.

SANTOS, C. T.; OSÓRIO, F. S. **An intelligent and adaptive virtual environment and its application in distance learning**. Proceedings of the Working Conference on Advanced Visual Interfaces. ACM Press, Gallipoli, Italy. Pp. 362 – 365, 2004.

SANTOS, W.R.;TOCZEK, J.R. B. E. C. T. **A utilização dos recursos EAD como apoio ao ensino presencial na educação básica**. 12 P, Revista Brasileira de Ensino de Ciência e Tecnologia.DOI:10.3895/S1982-873X2014000100006. 2014.

SEACORD, R. **Software Engineering Component Repository.** International Workshop on the Engineering of Computer Based Systems, Los Angeles, 1999.

SECONDLIFE. **Second Life.** Disponível em: <http://secondlife.com/>. Acesso em: jan. 2015.

SEI. **A framework for software product line practice.** Versão 5.0. Software Engineering Institute, Carnegie Mellon University. Disponível em: <http://www.sei.cmu.edu/productlines/index.html>. Acesso em: 01 ago. 2014.

SHAO, G; MCGRAW , R. **Service-oriented simulations for enhancing situation awareness.** Proceedings of the Spring Simulation Multiconference, ser. SpringSim ’09. San Diego, California: Society for Computer Simulation International, 2009, p. 48:1–48:7, ACM ID: 1639859. [Online]. Disponível em: <http://portal.acm.org/citation.cfm?id=1639809.1639859>. Acesso em: 06 abr. 2015.

SHERMAN, W.R., Craig, A.B. **Understanding Virtual Reality.** Morgan kaufmann, 2003.

SILVA, A. F. R; OLIVEIRA, M. R. M**. Projeto de um framework de desenvolvimento de interfaces para TV digital.** Trabalho de Conclusão de Curso - Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Ciência da Computação. Brasília, 2006.

SILVA, A. P. **Uma Linha de Produto de Software baseada na Web Semântica para Sistemas Tutores Inteligentes.** Tese de Doutorado. 185. Campina Grande. Universidade Federal de Campina Grande (UFCG), 2011.

SILVA, A. P.; COSTA, E.; BITTENCOURT, I. G. **Uma Linha de Produto de Software baseada na Web Semântica para Sistemas Tutores Inteligentes**. Revista Brasileira de Informação na Educação, v. 20, p. 87, 2012.

SILVA, R. C. E SILVA, A. R. **Tecnologias para Construção de Mundos Virtuais:** Um Comparativo Entre As Opções Existentes No Mercado. FAZU EM REVISTA, P. 211–215, 2011.

SILVA, R. P. **Suporte ao desenvolvimento e uso de frameworks e componentes.** Tese de Doutorado(UFRGS) Universidade Federal do Rio Grande do Sul – Programa de Pós-Graduação em Computação, 263f , 2000.

SINGHAL, S; ZYDA, M. **Networked Virtual Environment:** Design and Implementation. ACM Press, 331p, Siggraph Series, New York , 1999.

SINGHAL, S; ZYDA, M. **Networked Virtual Environment:** Design and Implementation. ACM Press, pp.331, Siggraph Series, New York, 1999.

SMARAGDAKIS, Y.; BATORY, D. Application Generators. **Encyclopedia of Electrical and Electronics Engineering**, j.G. Webster (ed.), Jhon Wiley and Sons, 2000.

SOFTWARE ENGINEERING: **A Practitioner's Approach**. 6. edition. New York, USA: McGraw Hill, 2004.

SOMMERVILLE, I. **Engenharia de Software**. 8ª ed. São Paulo: Addison-Wesley, 2007.

SOMMERVILLE, I. **Engenharia de Software.** 9ª edição, São Paulo: Person Prentice Hall, 529 p. 2011.

SOURCEFORGE. **X3DOM.** Disponível em: < http://sourceforge.net/projects/x3dom/>. Acesso em: 01 ago. 2014.

SOUSA, F. R.; MOREIRA, L. O.; DE MACÊDO, J. A. F.; MACHADO, J. C. **Gerenciamento de dados em nuvem:** Conceitos, sistemas e desafios. Tópicos em sistemas colaborativos, interativos, multimídia, web e bancos de dados, Sociedade Brasileira de Computação, pp.101-130, 2010.

SOUZA, D.A.C.M. **AvComponent**: Um Framework para Desenvolvimento de Componentes de Realidade Virtual em Infraestrutura de Compartilhamento de Componentes em Nuvem. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Pernambuco, 2014.

SOUZA, D.A.C.M.; FUJIOKA, R.C.; VASCONCELOS, C.R.; AZEVEDO, R.R.; FREITAS, F.; LIMA, T.;BARROS, H. **Uma Proposta de Categorização de TouchSensors na Definição de um Ambiente Extensível Aplicado à Área de Engenharia de Software.** WRVA'08 5 Workshop de Realidade Virtual e Aumentada, Bauru-SP, 2008.

SOUZA, D. A. C. M.; VASCONCELOS, C. R.; AZEVEDO, R.; FUJIOKA, R. C.; ALMEIDA, M. J. S. C.; FREITAS, F. **Honey:** Um Ambiente Virtual Baseado em Agentes para Apoiar o Ensino de Engenharia de Software. XIX Simpósio Brasileiro de Informática na Educação, 2008.

SURVEY OF CLOUD COMPUTING 2013. Disponível em: <http://www.northbridge.com/2013-cloud-computing-survey>. Acesso em: 06 mai. 2015.

SZYPERSKI, C. **Component Software.** Addison-Wesley, 2 edition, 2002.

SZYPERSKI. C, E. A. **Sumary of the Second International Workshop on Component** – Oriented Programming. Jyvaskyla.1997.

TAO, G.; ZHANG, X.; ZHANG, Q.**Virtual Operation of Motor Based on the Virtual Reality Technology.** Second International Conference on Intelligent Human-

Machine Systems and Cybernetics. 2010.

TARR, P.; OSSHER, H. **Multi-Dimensional Separation of Concerns and the Hyperspace Approach.** Proceedings Architectures and Component Technology: The State-of-the-Art in Software Development, 2000.

TAVARES, T. C. **Caracterização de cargas de trabalho para avaliação de desempenho em *Web Services*.** Dissertação de mestrado, Universidade de São Paulo, São Carlos, SP, 2009.

THOMAS, J. P.; THOMAS, M.; GHINEA, G**. Modeling of *Web Services* flow**. Proceedings of the IEEE International Conference on E-Commerce (CEC'03), IEEE Computer Society, 2003.

TORI, R.; KIRNER, C.; SISCOUTO, R. **Fundamentos e Tecnologia de Realidade Virtual e Aumentada**. Pará, Brasil: Editora SBC, 412 p, 2006.

TORQUATO, J. R. C. **Ubiproject:** uma infra-estrutura para Redes Sociais de projetos compatível com o OAI-PMH. João Pessoa: UFPB. Dissertação (mestrado) – UFPB/CCEN, 2009.

TOYOHARA, R. K. T. **Construindo *Web Services* para avaliação de desempenho de uma arquitetura orientada a serviços com qos**. Qualificação de mestrado, USP - Universidade de São Paulo, São Carlos, SP, 2009.

UNITY3D. **Crie jogos que você aprecia com o Unity.** Disponível em: <http://unity3d.com/pt/unity>. Acesso em: 01 ago. 2014.

VASCONCELOS, A. **Uma Abordagem de apoio à Criação de Arquitetura de Referência de Domínio baseadas na Análise de Sistemas Legados.** Rio de Janeiro: UFRJ, 2007. Tese (Doutorado em Engenharia de Sistemas e Computação), Faculdade de Engenharia de Sistemas, Universidade Federal do Rio de Janeiro, 2007.

VERBRAECK, A; VAN HOUTEN, S. T. **From simulation to gaming:** an object-

oriented supply chain training library. Simulation Conference, Proceedings of the Winter. IEEE, Dec. 2005.

VERNADAT, F.B. Enterprise modeling and integration: principles and applications. London, Chapman & Hall. 1996.

VINCE, J. **Virtual Reality Systems.** Addison-Wesley, 1995.

VINCE, J. **Introduction to Virtual Reality**. 2ª ed. Springer-Verlag, 2004.

VLISSIDES, J. **Pattern Hatching:** Design Patterns Applied. Addison-Wesley Longman Ltd., Essex, UK, 1998.

W3Ca. ***Web Services*** **Description Language** (Wsdl) 1.1. Disponível em:

W3Cb. HTTP - Hypertext Transfer Protocol. Disponível em: < http://www.w3.org/Protocols/> Acesso em: 01 ago. 2014.

W3Cc. **Standards**. Disponível em: <http://www.w3.org/standards//>. Acesso em: 01 ago. 2014.

W3Cd. VRML **Virtual Reality Modeling Language**. Disponível em: <http://www.w3.org/MarkUp/VRML/>. Acesso em: 01 ago. 2014.

W3Ce. **HTML5**. <http://www.w3.org/TR/html5/>. Acesso em: 20 set. 2014.

W3SCHOOLSa. **SOAP Introduction**. Disponível em: <http://www.w3schools.com/webservices/ws_soap_intro.asp/>. Acesso em: 01 ago. 2014.

W3SCHOOLSb. WSDL and UDDI. Disponível em: <http://www.w3schools.com/webservices/ws_wsdl_uddi.asp/>. Acesso em: 01 ago. 2014.

WALCZAK, K.; CELLARY, W. **Building database applications of virtual reality with X-VRML**. Proceedings of the 7th International Conference on 3D Web Technology, 111–120, 2002.

WEB3Da. **X3D**. Disponível em: < http://www.web3d.org/x3d>. Acesso em: 01 ago. 2014.

WEB3Db. **Document Number ISO/IEC14772-1:1997/Amd**. 1:2002(E). Disponível em < http://www.web3d.org/documents/specifications/14772-1/V2.1/index.html>. Acesso em: 01 ago. 2014.

WEB3Dc. **The Virtual Reality Modeling Language**. Disponível em: <http://www.web3d.org/documents/specifications/14772-1/V2.1/index.html>. Acesso em: 01 ago. 2014.

WEB3Dd. **Stantards**. Disponível em: <http://www.web3d.org/standards>. Acesso em: 01 ago. 2014.

WEB3De. **SAI.** Disponível em <http://www.web3d.org/documents/specifications/19777-2/V3.0/>. Acesso em: 01 ago. 2014.

WEB3Df. **Extensible 3D (X3D) language bindings Part 2**: Java. Disponível em <http://www.web3d.org/documents/specifications/19777-2/V3.0/>. Acesso em: 01 ago. 2014.

WEB3Dg. **Extensible 3D (X3D) Part 1:** Architecture and base components: Java. Disponível em: <http://www.web3d.org/documents/specifications/19775-1/V3.3/index.html>. Acesso em: 01 ago. 2014.

WEB3Dh. **X3D Profiles.** Disponível em: <http://www.web3d.org/x3d/profiles>. Acesso em: 01 ago. 2014.

WEISS, D. M.; LAI, C. T. R. **Software product-line engineering:** a family-based software development process. Addison-Wesley, 1999.

WERNER, C. M. L.; Braga, R. M. M. **Desenvolvimento Baseado em componentes**. Anais do XIV Simpósio Brasileiro de Engenharia de Software, pp.35-44, 2000.

WESCHTER, E. O.; TURINE, M. A. S. **Arquitetura de um Gerador de Aplicações Web Baseado no Framework Titan.** Dissertação de Mestrado em Ciência da Computação. Departamento de Computação e Estatística. Universidade Federal de Mato Grosso do Sul. Campo Grande, MS, Brasil. Julho 2008.

WIRFS-BROCK, R.J.; JOHNSON, R.E.; Surveying Current Research in Object-Oriented Design. Communications of the ACM, 33, 1990.

X3DOM. **About**. Disponível em: <http://www.x3dom.org/?page_id=2>. Acesso em: 01 ago. 2014.

X3DGRAPHICS. **X3D Extensible 3D Graphics for web Authors.** Disponível em < http://x3dgraphics.com/ >. Acesso em 20 fev. 2015.

XML. **XML.** Disponível em: <http://www.xml.com/>. Acesso em: 01 ago. 2014.

YE, Y. Supporting Component-Based Software Development with Active Component Repository Systems. Tese de Doutorado. Universidade do Colorado, 2001.

YIN, R.K. **Estudo de caso: planejamento e métodos**. 3. ed. Porto Alegre: Bookman, 2005.

YIN, ROBERT. **Estudo de caso** – planejamento e métodos. 3. ed. Porto Alegre: Bookman, 2005.

ZIVIANI, A. Implementacão de um Ambiente Virtual Distribuído Baseado em Java e VRML. Trabalho De Conclusão De Curso. UFRJ. 1998.

APÊNDICE A – Protocolo da revisão sistemática realizada

1. OBJETIVO

Investigar e identificar o emprego de técnicas de reuso no contexto dos ambientes virtuais que utilizam Realidade Virtual e Aumentada para representação do seu conteúdo.

2. UTILIDADE POTENCIAL

- Identificação de técnicas de reuso empregadas em ambientes de realidade virtual e aumentada;
- Constatação da utilização de técnicas distintas para resolução de problemas correlatos; e
- Identificação das principais abordagens utilizadas bem como a tendência de técnicas específicas.

3. NATUREZA, QUALIDADE E AMPLITUDE DA PESQUISA (SÍNTAXE DAS QUESTÕES DE PESQUISA)

3.1. QUESTÃO DE PESQUISA

Quais as técnicas de reuso são empregadas diretamente no contexto dos Ambientes Virtuais que tem seu conteúdo construídos com Realidade Virtual e Realidade Aumentada?

3.2. POPULAÇÃO DA PESQUISA

A população de pesquisa do estudo formar-se-á, em essência, por publicações originais e eletronicamente divulgadas com ênfase na utilização de técnicas de que dão suporte a distribuição e desenvolvimento de sistemas com Realidade Virtual e Aumentada.

3.3. FONTES DE BUSCA

Para a busca das fontes primárias, serão realizadas, com palavras-chaves pré-definidas, consultas aos seguintes sites especializados na publicação de conteúdo acadêmico:

- Academic Search Premier[64];
- Citeseer[65];
- Google Scholar[66];
- Ieee Xplore[67];
- Wiley Interscience[68];
- Sciencedirect[69];
- Biblioteca de teses e dissertações das Universidades USP, UFPE[70].

3.4. PALAVRAS-CHAVE

Na revisão sistemática, serão utilizados, para a seleção preliminar de fontes de leitura, operadores lógicos de concatenação de resultados de buscas, sites especializados em divulgação de publicações de conteúdo acadêmico e o conjunto de expressões a seguir:

("software reuse" OR "reuso de software" OR Reuse OR repository OR Framework)

AND

("Realidade Virtual" OR "Virtual Reality" OR "augmented reality" OR "mixed reality" OR "realidade Aumentada" OR "Realidade Misturada")

Na string anterior, apresentam-se os termos que serão utilizados nas buscas[71] pelas fontes primárias, empregando-se em sua construção operadores lógicos e símbolos especiais. Dado o caráter genérico da citada string, é importante ressaltar, para cada um dos repositórios mencionados na Seção 3.3 do Apêndice A, a necessidade de efetivação de eventuais adaptações antes do seu uso.

A escolha dos termos e suas relações foram definidas em função do objetivo principal da revisão, que é identificar a utilização direta de técnicas de reuso na construção e distribuição de ava 3D.

---

[64] Repositório digital de textos científicos acessível no endereço http://search.ebscohost.com.
[65] Biblioteca digital de produções científicas disponível em http://citeseerx.ist.psu.edu
[66] Portal de busca de publicações acadêmicas encontrado em http://scholar.google.com.
[67] Biblioteca digital disponível no endereço http://www.ieeexplore.ieee.org
[68]Serviço on-line de acesso a publicações científicas acessível através do endereço http://www3.interscience.wiley.com
[69] Repositório on-line de artigos e capítulos de textos científicos disponível em http://www.sciencedirect.com
[70] Repositórios on-line da USP e UFPE em http://www.teses.usp.br/ e
[71] Serão empregados operadores equivalentes à adição e à alternatividade lógica (operadores *and* e *or*), na forma especificamente definida em cada fonte de recursos pesquisada.

3.5. IDIOMA DE PESQUISA

Foi considerado de forma preferencial os artigos na língua inglesa, pois é amplamente aceita internacionalmente, bem como possui um maior volume de publicações. No entanto, artigos relevantes encontrados na língua portuguesa, publicados em eventos nacionais relacionados à área de pesquisa, também serão considerados.

3.6. DATA DE PUBLICAÇÃO E PERÍODO

Para realizar a RS referente à questão de pesquisa são considerados artigos publicados a partir do **ano 2001 até dezembro de 2014**. Mas eventualmente podem-se encontrar fontes clássicas com definições (livros com conceitos clássicos) que também serão considerados.

3.7. EFEITOS E RESULTADOS ALMEJADOS

Com a condução da revisão proposta, almeja-se a obtenção de lis*tag*em das técnicas de reuso utilizadas em sistemas de realidade virtual e aumentada.

4. CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

4.1. CRITÉRIOS DE INCLUSÃO

i. Descrições de sistemas de software que dão suporte ao desenvolvimento de aplicações de RV e RA em nuvem e web 2.0;

ii. Descrições de plataformas de software que dão suporte ao compartilhamento de componentes de RV e RA;

iii. Descrições de linguagens e representações que possibilitem a construção de

aplicações de RV e RA sobre arquiteturas, *frameworks* ou plataformas de *software;*

iv. Descrições de aplicações de RV e RA desenvolvidas a partir de reuso de componentes de software; e

v. Descrições de arquiteturas/*frameworks*/plataformas que dão suporte ao desenvolvimento e distribuição de aplicações de RV e RA;

4.2. CRITÉRIOS DE EXCLUSÃO

i. Trabalhos que simplesmente descrevam aplicações de RV e RA sem uma abordagem de reuso ou generalização de elementos de software;

ii. Artigos que abordam o assunto de RV e RA em nuvem, com foco em questões ligadas à infraestrutura da nuvem, questões de performance, questões de segurança e balanceamento de carga;

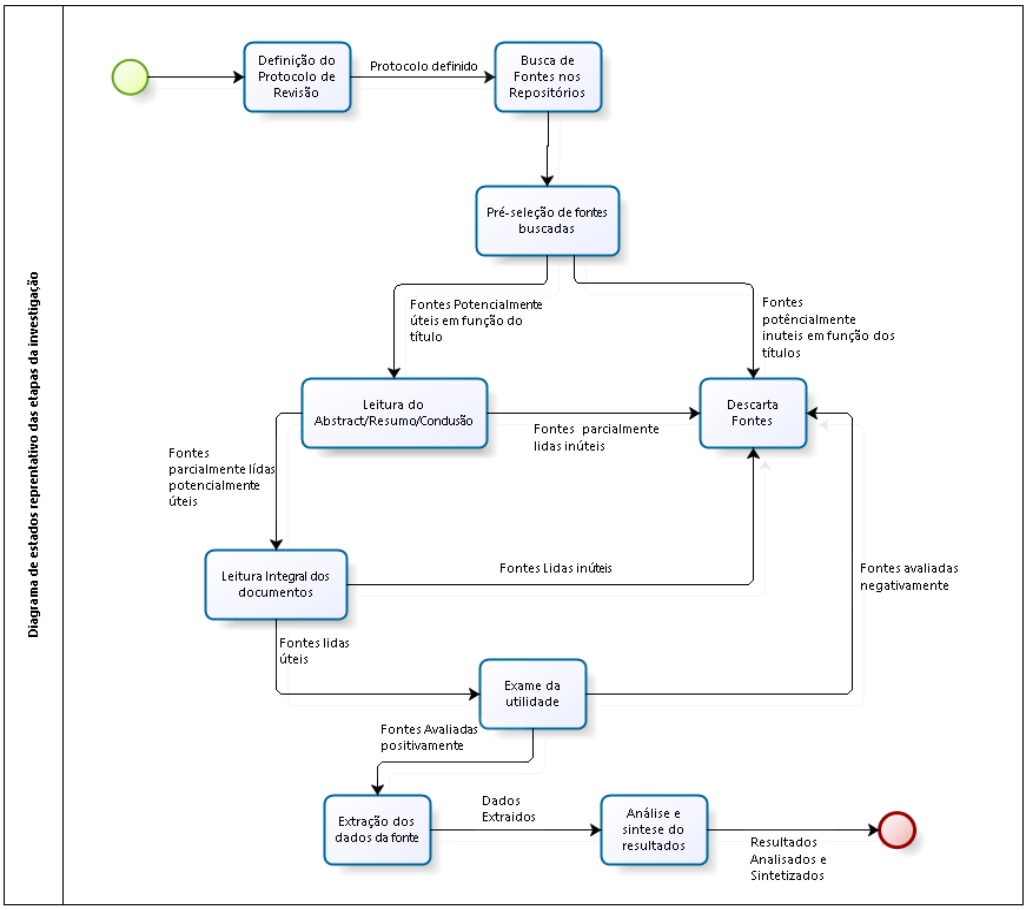
iii. Artigos cujos arquivos não estejam disponíveis online; e

iv. Artigos cujo foco não corresponda à questão de pesquisa.

5. ETAPAS DA REVISÃO

A Figura A.4.1 apresenta o diagrama de etapas da investigação realizada na rs.

Figura A.4.1 – Diagrama de estado representativo das etapas da investigação[72].



Fonte: O Autor

[72] Representação construída para ilustração da sistemática adotada para condução dos trabalhos.

APÊNDICE B – Tecnologias, imagens e configurações de desenvolvimento

1. Imagens das configurações do Scollabcore

Figura B.1.1 – Configuração dos aspectos no Spring Framework para as classes do Scollabcore.

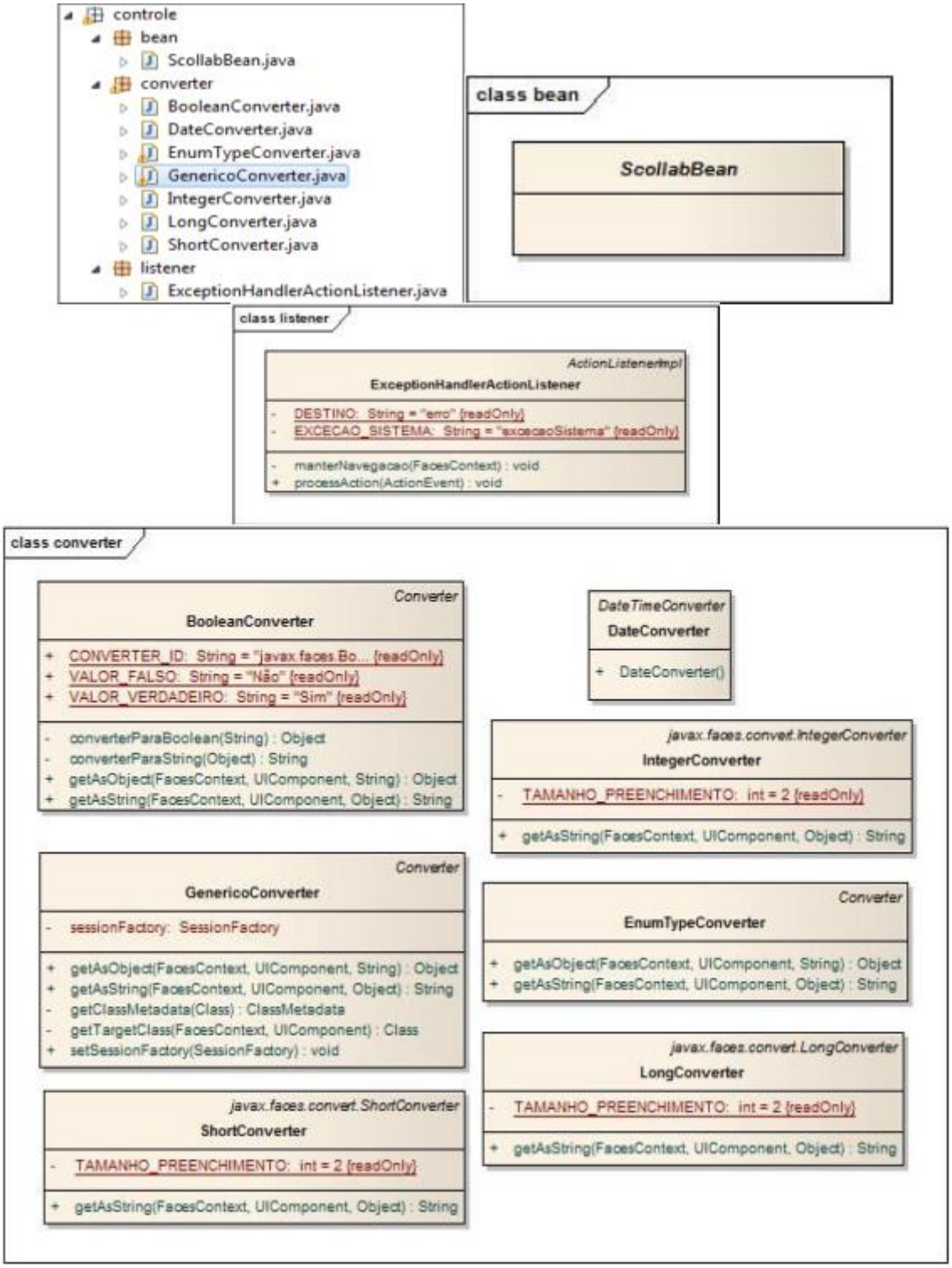
```
<!--#####################  CONFIGURAÇÃO DOS ASPECTOS  ###################-->
<aop:aspectj-autoproxy />
<aop:config>
    <aop:pointcut id="serviceOperation"
        expression="execution(* br.ufpb.di.labes.focos..*Service.*(..))" />
    <aop:aspect ref="profilerAspect">
        <aop:around pointcut-ref="serviceOperation" method="log" />
    </aop:aspect>
    <aop:aspect ref="profilerAspect">
        <aop:around pointcut-ref="daoOperation" method="log" />
    </aop:aspect>

    <aop:aspect ref="exceptionTranslator">
        <aop:around pointcut-ref="serviceOperation"
            method="tratarExcecao" />
    </aop:aspect>
</aop:config>
<!--#######################      ASPECTOS        ######################-->

<bean id="profilerAspect"
    class="br.ufpb.di.labes.infra.aspects.ProfilerAspect" />
```

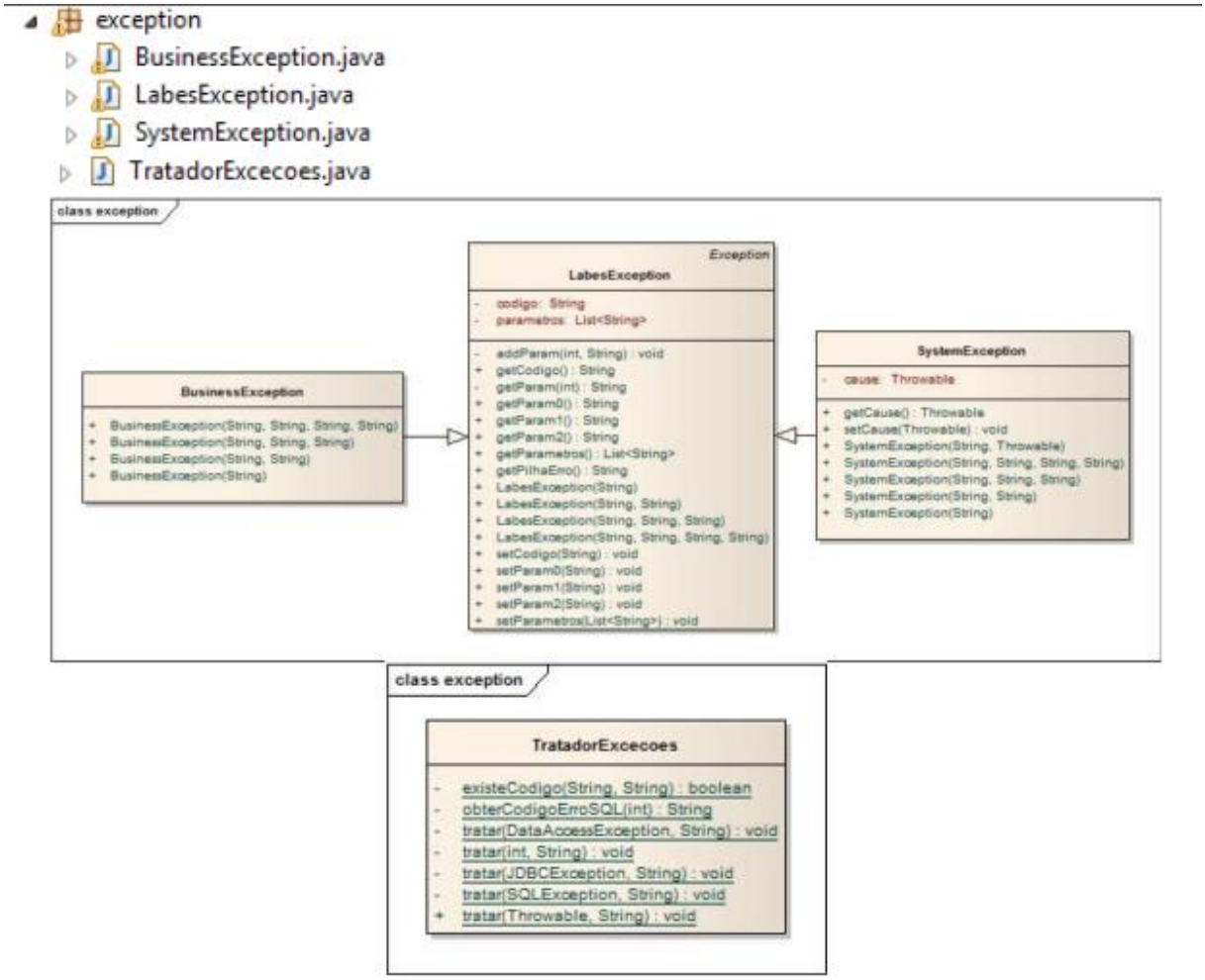
Fonte: Adaptado de Fujioka (2011).

Figura B.1.2 – Pacote Controle



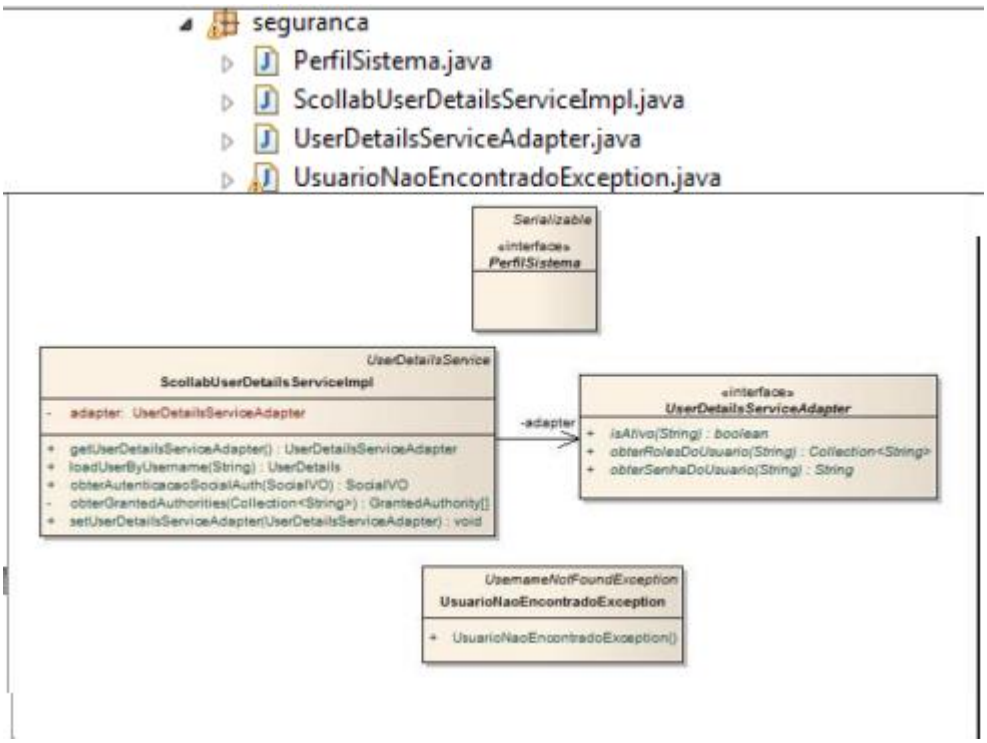
Fonte: Fujioka (2011).

Figura B.1.3 – Pacote Exception



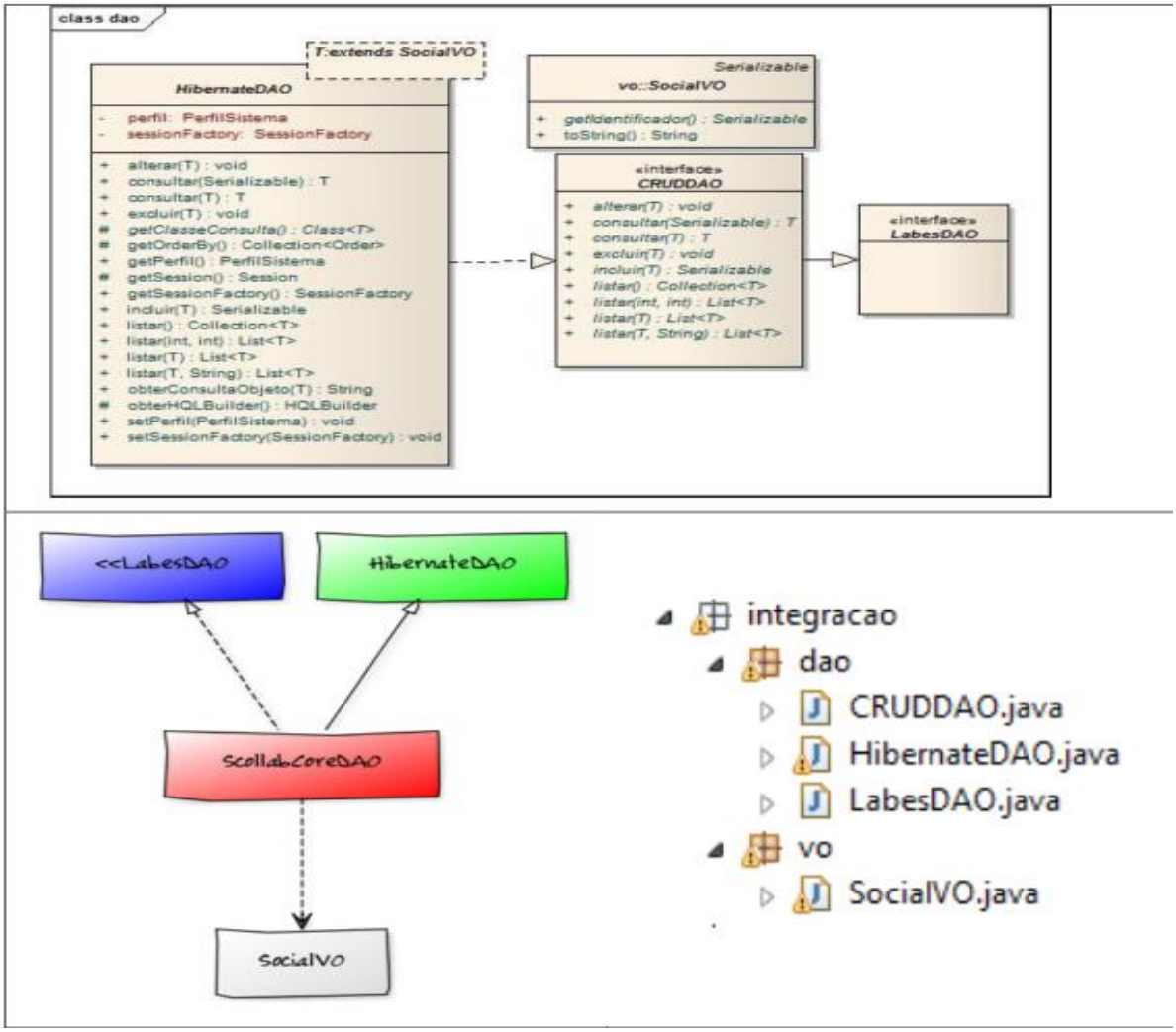
Fonte: Fujioka (2011)

Figura B.1.4 – Pacote de Segurança



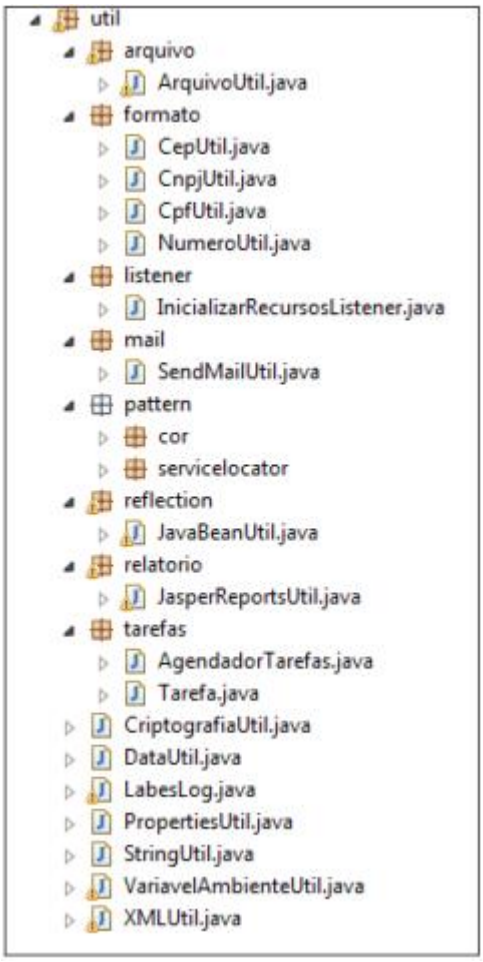
Fonte: Fujioka (2011).

Figura B.1.5 – Pacote de integração scollabcore



Fonte: Fujioka (2011).

Figura B.1.6 – Pacote útil Scollabcore



Fonte: Fujioka (2011).

2. IMAGENS DAS CONFIGURAÇÕES DOS ESTUDOS DE CASO

Figura B.2.1 – WSDL de chamada a serviços parte 1

```
- <definitions targetNamespace="http://ws.scollabcore.rodrigofujioka.com/" name="RepositorioServiceService">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://ws.scollabcore.rodrigofujioka.com/" schemaLocation="http://digitalocean.rodrigofujioka.com:8080/scollabcore/ws/AVWS?xsd=1"/>
      </xsd:schema>
    </types>
  - <message name="listaComponentes">
      <part name="parameters" element="tns:listaComponentes"/>
    </message>
  - <message name="listaComponentesResponse">
      <part name="parameters" element="tns:listaComponentesResponse"/>
    </message>
  - <message name="ConsomeComponentes">
      <part name="parameters" element="tns:ConsomeComponentes"/>
    </message>
  - <message name="ConsomeComponentesResponse">
      <part name="parameters" element="tns:ConsomeComponentesResponse"/>
    </message>
  - <message name="buscaCompoenentes">
      <part name="parameters" element="tns:buscaCompoenentes"/>
    </message>
  - <message name="buscaCompoenentesResponse">
      <part name="parameters" element="tns:buscaCompoenentesResponse"/>
    </message>
  - <portType name="RepositorioService">
    - <operation name="listaComponentes">
        <input wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/listaComponentesRequest" message="tns:listaComponentes"/>
        <output wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/listaComponentesResponse" message="tns:listaComponentesResponse"/>
      </operation>
    - <operation name="ConsomeComponentes">
        <input wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/ConsomeComponentesRequest" message="tns:ConsomeComponentes"/>
        <output wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/ConsomeComponentesResponse" message="tns:ConsomeComponentesResponse"/>
      </operation>
    - <operation name="buscaCompoenentes">
        <input wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/buscaCompoenentesRequest" message="tns:buscaCompoenentes"/>
        <output wsam:Action="http://ws.scollabcore.rodrigofujioka.com/RepositorioService/buscaCompoenentesResponse" message="tns:buscaCompoenentesResponse"/>
      </operation>
    </portType>
```

Fonte: o Autor

Figura B.2.2 – WSDL de chamada a serviços parte 2

```
- <binding name="RepositorioServicePortBinding" type="tns:RepositorioService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  - <operation name="listaComponentes">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="ConsomeComponentes">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="buscaCompoenentes">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
- <service name="RepositorioServiceService">
  - <port name="RepositorioServicePort" binding="tns:RepositorioServicePortBinding">
      <soap:address location="http://digitalocean.rodrigofujioka.com:8080/scollabcore/ws/AVWS"/>
    </port>
  </service>
</definitions>
```

Fonte: o Autor

3. TECNOLOGIAS UTILIZADAS

A linguagem de programação utilizada foi Java e como ambiente de desenvolvimento integrado (IDE) foi utilizada a IDE Eclipse[73], que possui um conjunto de plugins disponíveis na Web que, se bem organizados, oferecem uma série de componentes gráficos que auxiliam no desenvolvimento de aplicações. Porém, como o Eclipse não é o foco desta Tese e também não é a única ferramenta que pode ser utilizada para desenvolvimento de clientes para essa Arquitetura, não entraremos em muitos detalhes com relação ao ambiente de desenvolvimento em si utilizado.

O ScollabCore e as ferramentas dessa Tese foram desenvolvidos utilizando a plataforma *Java Enterprise Edition* (JEE), distribuído em várias camadas, mais o padrão MVC (GAMMA et al. 1995). Na camada de modelo e integração foi empregado a *Java Persistence API* (JPA) e o *Hibernate*. Na camada Visão foram utilizados os *Frameworks Java Server Faces* (JSF) e *facelets* e para integração desses foi utilizado o Spring.

3.1. JAVA SERVER FACES

A JSF[74] é uma tecnologia desenvolvida pela Oracle Sun Microsystems[75] que consiste em um *framework* para representar componentes de interface com o usuário e gerenciar o seu estado. A tecnologia JSF provê um conjunto pré-definido de componentes para atender às necessidades mais comuns, como caixas de textos, formulários e botões. Ela também implementa o padrão de projeto mvc que irá garantir a separação entre o modelo e visão, considerada uma boa prática de *design* de software, e entre visão e controlador, considerada importante em se tratando de interfaces Web (GAMMA et al., 1995; FOWLER, 2002).

Apesar do conjunto padrão de componentes do JSF cobrir diversos casos recorrentes

---

[73] ECLIPSE é uma ferramenta integrada de desenvolvimento que pode ser utilizada para construção de aplicações em diversas linguagens. Disponível em <http://www.eclipse.org/> . Acesso em 12 de fev 2015

[74] Disponível em <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html> acesso em 12 de fev 2015.

[75] Disponível em <www.oracle.com/> acesso em 12 de fev 2015.

na composição de interfaces Web, este pode ser estendido acrescentando novos componentes. Existem diversos projetos como o **Primefaces**[76] que fornecem gratuitamente componentes adicionais. Por ser um padrão estabelecido por um *Java Community Process* (JCP), processo colaborativo onde as especificações da tecnologia Java são criadas por membros da própria comunidade, cujo grupo de especialistas é composto por várias empresas que desenvolvem ferramentas de desenvolvimento como, por exemplo, Borland Software Corporation, IBM, Oracle e BEA Systems, é de se esperar que cada vez mais JSF seja incorporado a ferramentas de produtividade. Além disso, o JSF foi incorporado à mais recente versão da JEE.

3.2. HIBERNATE

A maioria das aplicações tem que se preocupar com a persistência de seus dados a qual pode ser implementada de várias formas. Uma das características presentes em várias aplicações é que diversas classes precisam acessar o banco de dados para recuperar informações, o que pode trazer problemas de manutenção, já que uma mudança na base de dados se propaga para todas as classes que utilizam dados provenientes do banco. Para solucionar este problema, as aplicações desenvolvidas nessa Tese utilizam o padrão já implementado no ScollabCore, o *Data Access Object* com base no DAO genérico provido pelo ScollabCore. Estes componentes possuem uma estratégia que define como o acesso ao banco de dados é realizado (JOHNSON, 2004; ALUR, 2004).

Assim, optou-se por se utilizar o *Framework Hibernate* como estratégia. Ele é responsável por mapear objetos em tabelas de um banco de dados relacional, estreitando o gap conceitual que existe entre os dois. Ele também fornece uma linguagem própria de seleção, o *Hibernate Query Language*[77] (HQL) similar ao SQL, mas que possibilita a seleção de objetos ao invés de linhas. O mapeamento objeto-tabela é feito através de arquivos XML ou de

[76] Disponível em <http://www.primefaces.com> acesso em 12 de fev 2015.
[77] Disponível em < https://docs.jboss.org/hibernate/orm/3.5/reference/pt-br/html/queryhql.html> acesso em 12 de fev 2015.

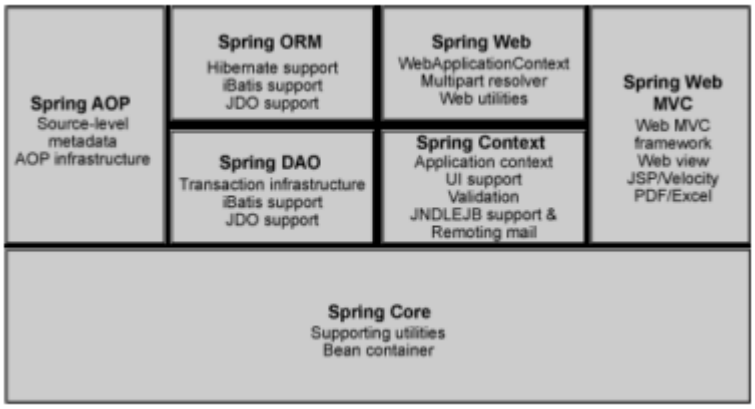
anotações nas próprias classes que representam o modelo. A configuração das características do banco de dados no Hibernate pode ser feita de diversas formas como, por exemplo, por meio da própria interface do *Framework* ou por meio de arquivos de configuração XML. As operações de criação, atualização, remoção e seleção de objetos são feitas através da API do *Hibernate* que, consultando seus arquivos de mapeamento e configuração, envia os comandos corretos ao banco de dados configurado. Desta forma, a aplicação fica portável entre bancos de dados.

Apesar de o *Hibernate* estreitar o gap conceitual entre objetos e tabelas, uma implementação que o utilize está sujeita a erros de programação como, por exemplo, o esquecimento de sessões abertas. Para evitar estes erros, optou-se por usar o Framework Spring pois o mesmo fornece um módulo de mapeamento objeto-relacional que integra-se ao *Hibernate*, controlando a abertura e o fechamento de sessões, evitando o aparecimento de erros como o descrito anteriormente.

3.3. SPRING FRAMEWORK

O Spring Framework é um projeto de código aberto que tem como objetivo simplificar a construção de aplicações utilizando a tecnologia JEE. O Spring foi criado por Rod Johnson e teve seu código aberto em 2003. Spring é dividido em módulos que podem ser usados separadamente ou em conjunto conforme a necessidade do projeto. É importante destacar de forma resumida a função de cada pacote do Framework ilustrada na Figura B.3.1.

Figura B.3.1 – Módulos do Spring



Fonte: adaptado de Fujioka (2011)

Pacote Core: Este pacote é a base do framework e provê a inversão de controle (*inversion of control: ioc*) e injeção de dependências.

- **Pacote Context**: construído na base do pacote core, fornece meios de acessar objetos. Oferece suporte à internacionalização, carregamento de recursos, propagação de eventos e criação transparente de contextos.
- **Pacote DAO**: provê uma camada de abstração ao JDBC. Também provê meios de gerenciar transações de forma programática ou declarativa.
- **Pacote ORM**: provê camadas de integração para APIS de mapeamento objeto relacional populares, incluindo JPA e Hibernate.
- **Pacote AOP**: provê uma implementação da programação orientada a aspectos, permitindo que se defina, por exemplo, adendos e pontos de junção para desacoplar funcionalidades que deveriam ficar separadas da regra de negócio.
- **Pacote WEB**: provê funcionalidades básicas para desenvolvimento Web, como upload de arquivos e a inicialização do container IoC usando servlets.
- **Pacote MVC**: provê separação entre código do domínio e formulários Web, permitindo ainda que todas as outras funcionalidades do Spring *framework* sejam utilizadas.

Outros módulos utilizados no scollab core são o de *dependency injection* (DI) e o de programação orientada a aspectos. O termo *dependency injection* foi cunhado em Fowler (2004) como uma forma mais específica de inversão de controle.

No Spring o gerenciamento, criação dos componentes e a configuração de

dependências são descritos em um arquivo em formato XML. Dessa forma, consegue-se um baixo acoplamento entre os módulos dos sistemas que o utilizam.

A programação orientada a aspectos é um paradigma que propõe uma forma de tratar interesses transversais, os aspectos, que costumam se espalhar pelo código em outros paradigmas como o da orientação a objetos. É importante ressaltar que a programação orientada a aspectos atua de forma complementar à orientação a objetos. Na arquitetura desenvolvida nesta tese ela é utilizada para realizar logging de métodos que estejam configurados no arquivo de configuração do Spring Framework. O controle de transações também é realizado pela programação orientada a aspectos, permitindo que pojos (*plain old java objects*) utilizem um modelo de transações declarativas similar ao modelo dos *enterprise java beans*.