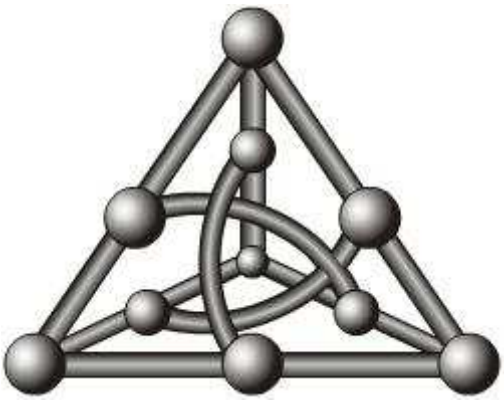
Hidra - Uma Biblioteca Java para o Desenvolvimento de Repositórios de Ativos de Software Orientado a Serviços baseada no modelo RAS

Danielli Urbieta e Pedro Souza

Trabalho de Conclusao de Curso apresentada à
Faculdade de Computação da
Universidade Federal de Mato Grosso do Sul



Orientador: Prof. MSc. Geraldo Barbosa Landre

Campo Grande, Junho de 2015

Resumo

Escreva o resumo aqui.

Palavras-chave: Java, SOA, Webservices, Repositório, Ativos Reusáveis de Software.

Abstract

Write the abstract here.

Keywords: Java, SOA, Webservices, Repository, Software Reusable Assets.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Texto.

1.1 Seção

Caso seja necessário dividir em seções.

Capítulo 2

Embasamento Teórico e Trabalhos Relacionados

2.1 Modelo de Especificação de Ativos Reutilizáveis

Um ativo de software segundo a definição da OMG (Object Management Group - 2005), é algo que visa a solução de algum problema em um determinado contexto, possui pontos de variabilidades e regras de como seu reuso pode ser realizado. A figura a seguir ilustra a composição de um ativo de software.

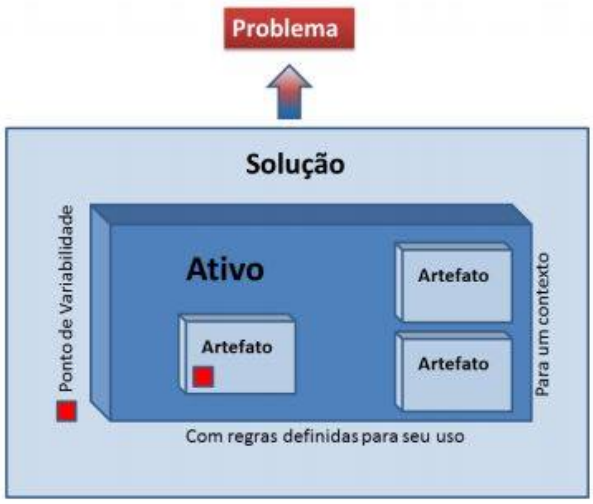


Figura 2.1: Ilustração de um ativo de software

Ou seja, um ativo de software reutilizável é o conjunto que compõe a solução de um determinado problema, sendo composto por diferentes artefatos, que podem ser tanto códigos fontes, quanto documentação de projeto, ferramentas adotadas, entre outros artefatos.

A OMG utiliza três pontos-chave para descrever os ativos reutilizáveis, que apresentam-se como: Granularidade, Variabilidade e Articulação. A seguir descrevemos a definição de cada um destes.

- Granularidade: descreve a quantidade de problemas específicos ou soluções alternativas um pacote de artefatos pode resolver. Ativos mais simples, podem compreender um único problema bem definido. Porém com o aumento da complexidade do ativo em conjunto com o aumento de seu tamanho, a sua capacidade de resoluções de problemas aumenta junto a sua granularidade.

- Variabilidade: condiz com a capacidade do ativo fornecer variabilidade. Ou seja a possibilidade de alteração de suas caracteristicas, a OMG exibe as caracteristicas de variabilidade de um ativo por meio de um conjunto formado por quatro tipos:
  - **Ativos caixa-preta:** a sua implementação interna não é conhecida e não pode ser alterada, por exemplo, componentes binários.
  - **Ativos caixa-branca:** a sua implementação interna é conhecida e podem ser alterada, por exemplo, códigos fonte.
  - **Ativos caixa-clara:** a sua implementação é conhecida, mas não podem ser modificados, por exemplo, documentação, modelos, fragmentos de código.
  - **Ativos caixa-cinza:** a sua implementação é conhecida, porém apenas determinados subconjuntos de artefatos podem ser modificados, por exemplo, serviços disponibilizados, que em geral, permitem manipulação por meio de parâmetros.
- Articulação: condiz com a capacidade do ativo de prover a solução do problema. Um ativo mais completo, constituído por exemplo de documentação, código fonte e testes que fornecem uma solução, possui um alto nível de articulação.

No âmbito da Engenharia de software, a utilização de ativos de softwares reutilizáveis são de grande interesse, visto que a prática de sua utilização fornece vantagens ao desenvolvimento de softwares como melhoria de qualidade do produto final, aumento de produtividade as equipes e redução de custos. A biblioteca desenvolvida neste trabalho, possui relação com o modelo de especificação de ativos reutilizáveis (RAS OMG-2005), pois contempla o desenvolvimento de repositórios de ativos de software que estejam de acordo com as orientações do padrão citado. Esta seção explora os principais conceitos que abragem as orientações especificadas no modelo RAS adotadas neste trabalho.

2.1.1 Pacotes de Ativos

Todo ativo reusável deve conter pelo menos um arquivo mafinesto, esse arquivo é um documento XML que valida o ativo, em relação a um dos *schemas* XML RAS conhecidos. Segundo a *OMG* um pacote de ativo é uma coleção de artefatos incluindo um arquivo manifesto e o *schema* que representa o perfil do ativo. Esses pacotes podem ser armazenados de duas maneiras distintas, das quais são:

- **Empacotado como um arquivo único**: abordagem comumente utilizada em ambientes de desenvolvimentos em equipes, os artefatos que compõe o ativo e o arquivo mafinesto são comprimidos em um único arquivo Zip, o que facilita a distribuição do ativo, e mantém o caminho de cada artefato em relação a sua raiz do documento.
- **Desempacotado com artefatos em localizações originais**: coloca o arquivo manifesto sob versionamento e mantém os artefatos em suas localizações originais, fornecendo flexibildiade na composição de componentes por meio de meta-informação

A biblioteca *Hidra* aborda especificamente a segunda forma de armazenamento de ativos reutilizáveis de software.

2.1.2 Reusable Asset Specification (RAS)

O RAS é descrito em duas principais categorias, o núcleo e seus Perfis. O núcleo representa os elementos fundamentais para a especificação de uma ativo como por exemplo: sua Classificação, Solução e Utilização, e os perfis descrevem extensões desses elementos fundamentais. O perfil de um ativo não altera a definição semântica de um elemento descrito no núcleo. O núcleo RAS não é

instaciavel, assim como uma classe abstrata, portanto um ativo deve possur um perfil particular, esse perfil pode estender o núcleo RAS, ou estender um outro perfil como mostrado na figura 4.2.

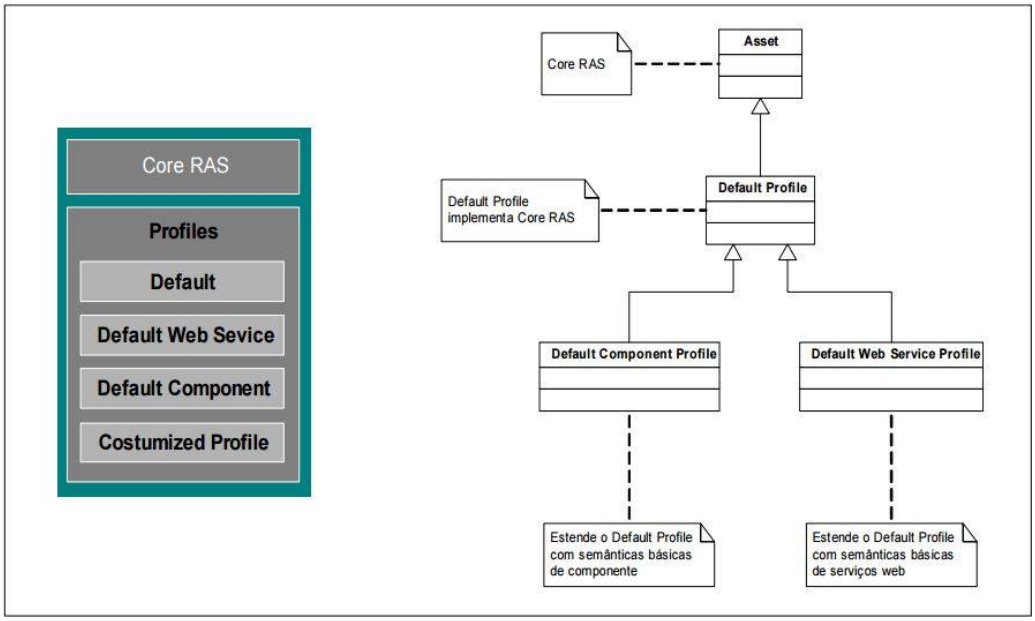


Figura 2.2: Estrutura de um Ativo de Softawe [1]

De acordo com a (*OMG*) um ativo de software é composto pela sua Classificação *(Classification)*, Solução *(Solution)*, Utilização *(Usage)*, pelo seu Perfil *(Profile)*, e caso esteja relacionado a outros ativos também possuirá (Related Assets). A figura 4.3 apresenta em um modelo conceitual os atributos do RAS que são definidos a segui:

- Classificação *(Classification)*: : um conjunto de descritores (descreve as qualidades e características do ativo.) que são usados para fazer a classificação do ativo, como descrever os contextos nos quais a reutilização do ativo é relevante.
- Solução *(Solution)*: contém as descrições de um ou mais artefatos que fazem parte do ativo.
- Utilização *(Soltuion)*: contém as regras para a instalação, customização e uso do ativo, ou seja, possui as instruções correspondentes a cada artefato que o ativo possui e um contexto de referência.
- Ativos Relacionados *(Related Assets)*: condiz com as descrições dos relacionamentos existentes entre os ativos.
- Perfil *(Profile)*: representa o perfil do ativo.

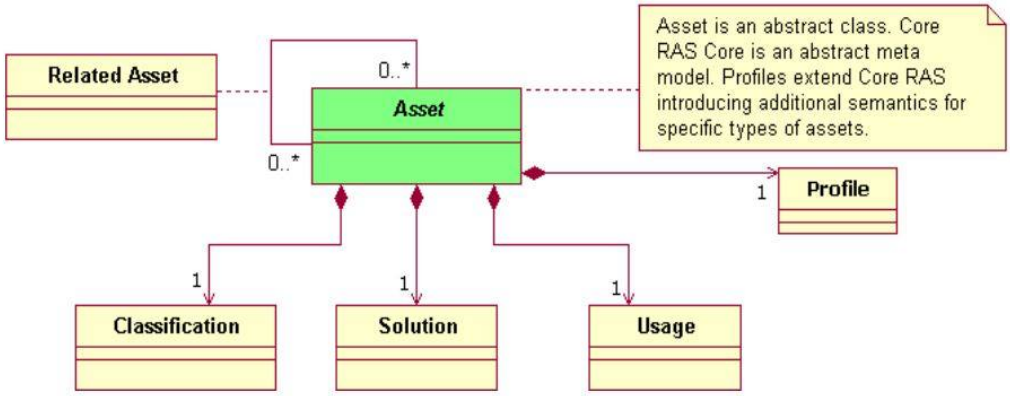


Figura 2.3: Estrutura de um ativo de Software (OMG, 2005)

Na figura 2.3 é apresentado um ativo de Software Reusável, sua representação no formato RAS daria se da seguinte forma.

```
<?xml version="1.0" encoding="UTF-8"?>
<asset name="" id="" short-description="">
    <profile name="" id-history="" version-major="" version-minor=""></profile>
    <solution>
        <artifacts>
            <artifact name="" type="" reference="" id="" version=""></artifact>
        </artifacts>
    </solution>
    <classification>
        <contexts>
            <context name="" id="" description="">
                <descriptionGroup name="" reference="" description=""></descriptionGroup>
            </context>
        </contexts>
        <descriptionGroups>
            <descriptionGroups name="" reference="" description=""></descriptionGroups>
        </descriptionGroups>
    </classification>
    <usage>
        <artifactActivities>
            <artifactActivy artifactId="" contextId="">
                <activities id="" task="" reference="" role="" taskRole="">
                    <variability id="" bindingRule=""></variability>
                </activities>
            </artifactActivy>
        </artifactActivities>
        <contextReferences>
            <contextReference contextId="">
                <activities id="" task="" reference="" role="" taskRole="">
                    <variability id="" bindingRule=""></variability>
                </activities>
            </contextReference>
        </contextReferences>
    </usage>
</asset>
```

Figura 2.4: Representação de um ativo no formato RAS

Nesta seção apresentamos as principais definições e orientações do modelo RAS utilizado como fonte de informação para desenvolvimento deste trabalho. Um dos aspectos abordados pela biblioteca Hidra é a validação, aceitação e persistência, em repositórios de ativos de software, condizentes com o padrão RAS descrito nesta seção.

Capítulo 3

Tecnologias e Ferramentas

3.1 Descrição das tecnologias e ferramentas utilizadas

Esta seção apresenta as ferramentas e tecnologias utilizadas no desenvolvimento do projeto *Hidra*.

- **NetBeans IDE (Integrated Development Environment)**. Ambiente de desenvolvimento utilizado. Ferramenta para que permite escrever, compilar, depurar e instalar programas. O IDE é completamente escrito em Java, mas também da suporte a qualquer linguagem de programação. possui um grande número de módulos para extender suas funcionalidades. O NetBeans IDE é livre, sem restrições à sua forma de utilização.
- **JDK 8 (Java Development Kit)**. Inclui ferramentas úteis para desenvolver e testar os programas escritos na linguagem de programação Java e em execução na plataforma Java.
- **JAXB (Java Architecture for XML Binding)**. Auxilia na manipulação de documentos XML, através da linguagem de programação Java. Ao invés de percorrer o documento, separá-lo logicamente em partes discretas, e repassar o conteúdo para a aplicação Java (ORT, 2003), assim como o SAX (SAX, 2005) ou o DOM (W3C, 2005), o JAXB funciona em duas etapas.
  - Um esquema XML é compilado em códigos fonte de classes.
  - Após o mapeamento (*binding*) de um esquema XML em um conjunto de classes, então, pode-se, através de uma aplicação que utilize as bibliotecas do JAXB e as classes geradas pelo compilador:
    - criar uma árvore de objetos em memória a partir de um documento XML (*unmarshalling*);
    - criar um documento XML a partir de objetos Java (*marshalling*);
- **JGit**. Biblioteca Java do sistema de controle de versão Git, na qual contém.
  - rotinas de acesso ao repositório
  - protocolos de rede
  - principais algoritmos de controle de versão
- **Apache Maven**. Ferramenta de automação para compilação utilizada em projetos Java, Similar à ferramenta Ant, hospedada pela Apache Software Foundation. Utiliza-se um arquivo XML (*POM*) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e plug-ins necessários. O maven Importa bibliotecas Java e seus plug-ins dinamicamente de um ou mais repositórios online, como o *Maven 2 Central Repository*.

- **java.util.Properties**. Classe Java que permite armazenar informações indexadas por uma palavra chave. A partir dela é possível fazer uma rápida pesquisa para recuperar informações necessárias em aplicativos Java SE e Java EE. As informações armazenadas por esta classe podem permanecer em memória ou ser persistida em um arquivo texto, sendo por este motivo muito utilizada para armazenar propriedades de aplicativos (caminho de servidor, Logins, Senhas de usuários, caminho de diretórios). Uma lista de propriedades pode conter outra lista de propriedades contendo outros "padrões"; Nesta segunda lista de propriedades será pesquisado se a chave de propriedade não for encontrada na lista de propriedades original.

- **Padrões de Projeto Grasp e Grop**. [2] Utilização de padrões Facade e Especialista na Informação.

  Facade: Classe Facade para simplificar o acesso a subconjuntos de métodos e classes, diminuição do acoplamento, e melhor manutenibilidade.

  Especialista na Informação: Padrão utilizado para atribuição de responsabilidades. As responsabilidades serão atribuidas a quem realmente detêm a informação necessária para preencher os requisitos daquela responsabilidade.

Capítulo 4

Hidra

O controle e versionamento de componentes e artefatos de software, é uma das importantes etapas da engenharia de software, e o desenvolvimento em equipes necessita de um gerenciamento de ativos capaz de atender a demanda dos processos ligados ao desenvolvimento de software.

O projeto *Hidra* consiste de uma biblioteca para apoiar o desenvolvimento de repositórios de ativos de software padronizados. Baseada nos conceitos e funcionalidades do famoso e bem sucedido controlador de versões GIT, a biblioteca provê mecanismos úteis para o armazenamento de informações relevantes para a gestão de ativos, e favorece a utilização de suas funcionalidades por meio de serviços integrados. De modo a garantir a qualidade da análise e desenvolvimento deste projeto, as especificações da biblioteca *Hidra* serão apresentada na forma de documentação técnica e documentação funcional.

Neste capitulo são apresentados os principais componentes para o desenvolvimento da biblioteca tex*Hidra*. Na seção 4.1 é apresentada a derivação dos requisitos da arquitetura de referência. Na seção 4.2 são apresentados os Requisitos Funcionais e Não Funcionais. A seção 4.3 é composta pelo glossário. A seção 4.4 apresenta o diagrama de componentes ao qual representa o projeto da biblioteca tex*Hidra*. Por fim a seção 4.5 apresenta o diagrama de classes que compõe o projeto.

4.1 Derivação dos Requisitos da Biblioteca Hidra a partir dos Requisios Arquiteturais da Cambuci

Os requisitos funcionais e não-funcionais da biblioteca *Hidra* foram elicitados a partir da derivações dos requisitos arquiteturais da arquitetura de referência Cambuci [3], bem como, a partir da identificação de novos requisitos voltados diretamente para a definição da biblioteca *Hidra*.

A tabela a seguir apresenta os requisitos arquiteturais da arquitetura de referência Cambuci (colunas ID e Requisito Original), juntamente com os requisitos respectivamente derivados à biblioteca *Hidra* (coluna Requisitos Derivados). Os requisitos específicos da biblioteca *Hidra* foram identificados adotando como padrão as siglas RF para os requisitos funcionais e RN para a requisitos não-funcionais.

Tabela 4.1: Tabela de Requisitos Hidra

ID	**Requisito Original**	**Requisito Derivado**
RA-AS[1]	A arquitetura de referência deve possibilitar que repositórios de ativos de software incluam um novo ativo, que pode ser composto por vários artefatos.	RF-01 e RF-02
RA-AS[2]	A arquitetura de referência deve possibilitar que repositórios de ativos de software forneçam mecanismo para aceitação e certificação de ativos.	RF-03 e RF-04
RA-AS[3]	A arquitetura de referência deve possibilitar que repositórios de ativos de software desativem ativos que não serão mais utilizados.	RF-05
RA-AS[4]	A arquitetura de referência deve possibilitar que repositórios de ativos de software permitam a classificação de um ativo e também informar o contexto de sua utilização.	RF-06 e RF-07
RA-AS[5]	A arquitetura de referência deve possibilitar que repositórios de ativos de software registrem a dependência entre ativos.	RF-08
RA-AS[6]	A arquitetura de referência deve possibilitar que repositórios de ativos de software notifiquem os interessados sobre mudanças que aconteçam no ativo.	RF-09
RA-AS[7]	A arquitetura de referência deve possibilitar que repositórios de ativos de software permitam realizar buscas e recuperação dos ativos	RF-10, RF-11, RF-12
RA-AS[8]	A arquitetura de referência deve possibilitar que repositórios de ativos de software permitam a navegação entre ativos	Não derivado para a versão atual da *Hidra* (Trabalho Futuro).
RA-AS[9]	A arquitetura de referência deve possibilitar que repositórios de ativos de software aceite múltiplas fontes de origem de ativos, com o objetivo de facilitar a integração entre equipes e entre repositórios diferentes.	RN-01
RA-AS[10]	A arquitetura de referência deve possibilitar que repositórios de ativos de software criem e armazenem múltiplas versões de um mesmo ativo.	RN-02
RA-AS[11]	A arquitetura de referência deve possibilitar que repositórios de ativos de software gerencie a configuração, como por exemplo, a definição dos itens do ativo que são configuráveis, o controle de mudanças dos itens do ativo que são configuráveis.	RN-03
RA-AS[12]	A arquitetura de referência deve possibilitar que repositórios de ativos de software permita o registro de impressões dos usuários a respeito da versão do ativo que eles utilizaram.	Não derivado para a versão atual da *Hidra* (Trabalho Futuro).
RA-AS[13]	A arquitetura de referência deve possibilitar que repositórios de ativos de software registrem métricas coletadas sobre a utilização do ativo.	Fora do escopo da *Hidra* (Trabalho Futuro).

Continua na página seguinte

Tabela 4.1 – *Tabela de Requisitos Arquiteturais Cambuci*

ID	**Requisito Original**	**Requisito Derivado**
RA-AS[14]	A arquitetura de referência deve possibilitar que repositórios de ativos de software ofereçam informações relativas ao reúso, iniciativas de reúso, ativos mais usados, etc.	Fora do escopo da *Hidra* (Trabalho Futuro).
RA-AS[15]	A arquitetura de referência deve possibilitar que repositórios de ativos de software permitam o acesso de acordo com o papel que o usuário assume.	Não derivado para a versão atual da *Hidra* (Trabalho Futuro).
RA-AS[16]	A arquitetura de referência deve possibilitar que repositórios de ativos de software garantam a integridade dos ativos, ou seja, que eles não sofram alterações não autorizadas.	RN-04
RA-AS[17]	A arquitetura de referência deve possibilitar que repositórios de ativos de software realizem o gerenciamento de transação, garantindo a atomicidade, consistência, isolamento e durabilidade.	RF-13
RAS[1]	A arquitetura de referência de possibilitar que repositórios de ativos de software desenvolvidos para persistir diferentes tipos de ativos possam ser facilmente integrados.	RF-14
RAS[2]	A arquitetura de referência deve possibilitar que repositórios de ativos de software implementados em linguagens de programação distintas e sob diferentes plataformas possam ser facilmente integrados.	RF-15
RAS[3]	A arquitetura de referência deve prover mecanismos para que repositórios de ativos de software na forma de serviços possam ser publicados e posteriormente descobertos por aplicações cliente.	RN-01
RAS[4]	A arquitetura de referência de prover mecanismos para que repositórios de ativos de software orientados a serviço possam ser compostos por processos de negócio ou utilizados por aplicações cliente.	RN-01 e RN-04
RAS[5]	A arquitetura de referência deve viabilizar o desenvolvimento de repositórios de ativos de software que disponibilizem informações sobre suas características e direções normativas de uso, por meio de descrições padronizadas.	RF-16
RAS[6]	A arquitetura de referência deve viabilizar o desenvolvimento de repositório de ativos de software que disponibilizem descrições semânticas, permitindo assim sua classificação nos repositórios de serviço.	RF-17
RAS[7]	A arquitetura de referência deve viabilizar o desenvolvimento de repositório de ativos de software que tenham à disposição informações e documentos relacionados às suas características de qualidade.	RF-18

Continua na página seguinte

Tabela 4.1 – *Tabela de Requisitos Arquiteturais Cambuci*

ID	**Requisito Original**	**Requisito Derivado**
RAS[8]	A arquitetura de referência deve prover mecanismos para a captura, monitoramento, registro e sinalização do não cumprimento de requisitos de qualidade estabelecidos entre serviços provedores e serviços clientes.	RF-19
RAS[9]	A arquitetura de referência deve viabilizar o desenvolvimento de repositório de ativos de software escalável, capaz de evoluir de maneira incremental, por meio da composição de novas funcionalidades disponíveis na forma de serviços.	RN-05
RAS[10]	A arquitetura de referência deve possibilitar que serviços de repositório de ativos de software e composições desses serviços sejam tratados uniformemente, ou seja, possam ser publicados, localizados e utilizados da mesma forma.	RN-01
RAS[11]	A arquitetura de referência deve possibilitar que serviços do repositório de ativos de software possam interagir diretamente ou por meio do uso de barramentos de serviço.	RN-01

4.2 Requisitos Funcionais Hidra

A partir da Tabela 4.1 os seguintes requisitos funcionais da biblioteca Hidra foram criados.

Tabela 4.2: Requisitos Funcionais Hidra

ID	**Requisito**	**Cambuci**	**Solução**
RF-01	A biblioteca *Hidra* deve possibilitar a **inclusão de ativos de software**, levando em consideração a composição de um ativo por diferentes artefatos.	RA-AS[1]	Os ativos reusáveis de software são armazenados no repositório em forma de diretórios, por meio, da segunda forma de armazenamento RAS [**?**].
RF-02	A biblioteca *Hidra* deve fornecer mecanismos a fim de **listar artefatos que compõem um ativo de software** armazenado no repositório.	RA-AS[1]	Requisito implementado por meio dos métodos *Asset.getSolution()* e *Asset.setSolution()*.
RF-03	A biblioteca *Hidra* deve possuir uma **estrutura padronizada de representação, comunicação e armazenamento** de ativos de software.	RA-AS[2]	Foi adotado o padrão RAS atualmente em sua versão 2.2.

Continua na página seguinte

Tabela 4.2 – *Requisitos Funcionais Hidra*

ID	**Requisito**	**RA Cambuci**	**Solução**
RF-04	A biblioteca *Hidra* deve possibilitar que todo novo ativo de software seja **validado e certificado** de acordo com o padrão adotado.	RA-AS[2]	As regras especificadas no padrão RAS, expressas em forma de um XSD *NomeArquivoXSD.xsd*, são consultadas ao validar e certificar um Ativo antes de qualquer atualização ou inserção (método *Asset.validate()*).
RF-05	A biblioteca *Hidra* deve possibilitar que ativos de software, que não forem mais utilizados, **sejam removidos do repositório**.	RA-AS[3]	Requisito implementado por meio do método *Repository .removeAsset(Asset asset).*
RF-06	A biblioteca *Hidra* deve possibilitar a adição de informações para **classificação de um ativo** e também o contexto de sua utilização.	RA-AS[4]	Requisito implementado por meio dos métodos *Asset.getClassification()* e *Asset.setClassification().*
RF-07	A biblioteca *Hidra* deve possibilitar a adição de informações sobre **regras para instalação, personalização, e utilização** do ativo.	extensão do requisito RA-AS[4] baseando-se no padrão RAS.	Requisito implementado por meio dos métodos *Asset.getUsage()* e *Asset.setUsage().*
RF-08	A biblioteca *Hidra* deve possibilitar o **registro de dependência entre ativos**.	RA-AS[5]	Requistio implementado por meio dos métodos *Asset.getRelatedAssets()* e *Asset.setRelatedAsset().*
RF-09	A biblioteca *Hidra* deve fornecer mecanismos a fim de **oferecer informações relevantes a todos os interessados, sobre mudanças** que aconteçam no ativo de software: data de alteração, autor da alteração, o que foi alterado e descrição sobre a alteração.	RA-AS[6]	Requisito implementado por meio do método *Asset.getLog()*
RF-10	A biblioteca *Hidra* deve fornecer mecanismos a fim de **listar ativos armazenados no repositório**.	RA-AS[7]	Requisito implementado por meio do método *Repository.listAssets()*

Continua na página seguinte

Tabela 4.2 – *Requisitos Funcionais Hidra*

ID	**Requisito**	**RA Cambuci**	**Solução**
RF-11	A biblioteca *Hidra* deve fornecer mecanismos a fim de **recuperar um ativo armazenado no repositório** (*download*).	RA-AS[7]	Requisito implementado por meio do método *Repository .retrieveAsset()*
RF-12	A biblioteca *Hidra* deve fornecer mecanismos a fim de **buscar ativos armazenados no repositório**.	RA-AS[7]	Requisito não implementado na versão atual da biblioteca *Hidra* (Trabalho Futuro).
RF-13	A biblioteca *Hidra* deve fornecer mecanismos a fim de **garantir a atomicidade, consistência e isolamento de transações de controle** de ativos de software	RA-AS[17]	A biblioteca provê recursos para que as transações sejam controladas considerando os aspectos citados: os recursos da API jGit e a camada de serviços. Mas essa implementação deverá ser realizada diretamente no repositório.
RF-14	A biblioteca *Hidra* deve fornecer mecanismos a fim de **permitir a persistência de diferentes tipos de ativos**.	RAS[1]	O padrão RAS, adotado na biblioteca *Hidra* para a implementação dos requisitos relacionados ao Ativo de Software, permite a persistência de diferentes tipos de ativos, desde que a estrutura de cada ativo seja descrita em sua solução (métodos *Asset.getSolution()* e *Asset.setSolution()*).

Continua na página seguinte

Tabela 4.2 – *Requisitos Funcionais Hidra*

ID	**Requisito**	**RA Cambuci**	**Solução**
RF-15	A biblioteca *Hidra* deve fornecer mecanismos a fim de **permitir a persistência de ativos implementados em diferentes linguagens de programação**.	RAS[2]	O padrão RAS, adotado na biblioteca *Hidra* para a implementação dos requisitos relacionados ao Ativo de Software, permite a persistência de ativos implementados em diferentes linguagens de programação, desde que as regras para instalação, personalização, e utilização de cada ativo seja descrita na especificação de seu uso (métodos *Asset.getUsage()* e *Asset.setUsage()*).
RF-16	A biblioteca *Hidra* deve fornecer uma **camada de serviços (*Webservice*)** com informações sobre suas características e direções normativas de uso, por meio de descrições padronizadas seguindo o padrão DNS para descoberta de serviços.	RAS[5]	Requisito não implementado na versão atual da biblioteca *Hidra* (Trabalho Futuro).
RF-17	A biblioteca *Hidra* deve viabilizar o desenvolvimento de um repositório de ativos de software com uma camada *webservice* com descrições semânticas, permitindo assim sua classificação nos respositórios de serviços.	RAS[6]	Requisito não implementado na versão atual da biblioteca *Hidra* (Trabalho Futuro).
RF-18	A biblioteca *Hidra* deve viabilizar o desenvolvimento de repositório de ativos de software que tenham à disposição informações e documentos relacionados às suas características de qualidade.	RAS[7]	Requisito não implementado na versão atual da biblioteca *Hidra* (Trabalho Futuro).
RF-19	A biblioteca *Hidra* deve prover mecanismos para a captura, monitoramento, registro e sinalização do não cumprimento de requisitos de qualidade estabelecidos entre serviços provedores e serviços clientes.	RAS[8]	Requisito não implementado na versão atual da biblioteca *Hidra* (Trabalho Futuro).

4.2.1 Requisitos Não Funcionais Hidra

A partir da Tabela 4.1 os requisitos não funcionais da biblioteca Hidra foram criados.

Tabela 4.3: Requisitos Não-Funcionais Hidra

ID	Requisito	Cambuci	Solução
RN-01	A biblioteca *Hidra* deve fornecer mecanismos para que repositórios de ativos de software aceitem múltiplas fontes de origem de ativos, por meio de serviços web que possam ser publicados, localizados e utilizados de maneira uniforme.	RA-AS[9], RAS[3], RAS[4], RAS[10]	Requisito é atendido por meio da camada de serviço provida pela biblioteca, que segue o padrão REST, que permitirá ao repositório desenvolvido, tendo como base a *Hidra*, fácil acesso e integração a múltiplas ferramentas.
RN-02	A biblioteca *Hidra* deve fornecer mecanismos de versionamento aos ativos de software.	RA-AS[10]	Requisito é atendido por meio da camada de persistência provida pela biblioteca, que utiliza a API jGit para manipulação das operações de Gerenciamento de Configuração sobre um repositório Git, e permitirá que o repositório armazene múltiplas versões de um mesmo ativo.
RN-03	A biblioteca *Hidra* deve oferecer mecanismos para gerenciamento da configuração de ativos de software.	RA-AS[11]	Requisito é atendido por meio da camada de persistência provida pela biblioteca, que utiliza a API jGit para manipulação das operações de Gerenciamento de Configuração sobre um repositório Git, e permitirá que o repositório gerencie configurações de ativos de software.
RN-04	A biblioteca *Hidra* permitir que repositórios garantam que seus ativos de software não sofram alterações não autorizadas.	RA-AS[16], RAS[4]	Requisito é atendido por meio do controle de usuários da camada de persistência provida pela biblioteca, que utiliza a API jGit para manipulação das operações de Gerenciamento de Configuração sobre um repositório Git. Na versão inicial, a biblioteca utiliza um usuário padrão informado no seu arquivo de propriedades (hidra.properties).

Continua na página seguinte

Tabela 4.3 – *Requisitos Não-funcionais Hidra*

ID	**Requisito**	**RA Cambuci**	**Solução**
RN-05	A biblioteca *Hidra* deve ser extensível de modo a viabilizar o desenvolvimento de repositório de ativos de software escalável, capaz de evoluir de maneira incremental, por meio da composição de novas funcionalidades disponíveis na forma de serviços.	RAS[9]	Requisito é atendido por meio dos padrões adotados para a implementação da biblioteca: i) Padrão RAS para representação e manipulação de Ativos de Software Reusáveis; ii) Divisão dos recursos providos em camadas (jGit para persistência, Hidra para regras de negócio, HidraService para fornecimento de serviços; iii) Padrões de Projeto (tanto padrões GRASP quanto padrões GoF) adotados na implementação, como por exemplo, Singleton, Facade, Strategy, Especialista na Informação).

Capítulo 5

Conclusão

Escreva a conclusao aqui.

Referências Bibliográficas

[1] Henrique Rocha de Faria and Reginaldo Arakaki. Um Modelo de Processo de Apoio ao Desenvolvimento de Software Baseado em Componentes, Orientado a Qualidade, e Centrado em um Repositório, 2005.

[2] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall, 2004.

[3] Márcio Osshiro and Maria Istela Cagnin. Reutilização De Modelagem De Negócios Baseada Em Visões, 2014.