UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL – UFMS
FACULDADE DE COMPUTAÇÃO – FACOM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCIANO ÉDIPO PEREIRA DA SILVA

ONTO-ResAsset: Ontologia de Ativos Reutilizáveis

Campo Grande - MS

Setembro - 2013

LUCIANO ÉDIPO PEREIRA DA SILVA

ONTO-ResAsset: Ontologia de Ativos Reutilizáveis

Dissertação apresentada, defendida e aprovada pelo Programa de Mestrado *Stricto Sensu* em Ciência da Computação da Faculdade de Computação, mantido pela Universidade Federal do Mato Grosso do Sul, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação (Área de Concentração: Engenharia de *Software*).

Orientadora:
Profª. Drª. Maria Istela Cagnin Machado

Campo Grande - MS
Setembro - 2013

Dedico este trabalho às minhas mães.

AGRADECIMENTO

À minha família, pelos ensinamentos, pelo apoio, amor, carinho e alegria que me proporcionaram durante a minha vida. Agradeço em especial aos meus tios, Devanir e Jessé que contribuíram de forma significativa durante minha mudança para Campo Grande me acolhendo em seu lar.

Aos amigos, que sempre estiveram ao meu lado (mesmo distantes fisicamente) e me deram forças para continuar essa caminhada, e que incentivaram e apoiaram durante toda esta etapa de minha vida, em especial ao Marcelo Benites e a Rosana Takehara, pessoas especiais que conheci no mestrado, me ajudaram de forma sublime e grandiosa, serei sempre grato.

À minha orientadora, Profa. Dra. Maria Istela Cagnin Machado, pelos seus ensinamentos, conselhos e atenção que proporcionaram o desenvolvimento deste trabalho, e principalmente pela amizade e dedicação oferecida.

Aos meus colegas de trabalho na GENDOC e na EAD-UFMS, especialmente ao Humberto (Begnini), que foi primordial para que continuasse em Campo Grande e realizasse este trabalho.

A todos os colegas, amigos e professores do programa de mestrado e do LEDES pelo apoio oferecido.

À CAPES pelo auxílio financeiro.

Silva, L. E. P. **ONTO-ResAsset: Ontologia de Ativos Reutilizáveis**. Dissertação (Mestrado em Ciência da Computação) – Faculdade de Computação. Universidade Federal do Mato Grosso do Sul-MS, Campo Grande, 2013.

RESUMO

A reutilização de software, apesar de ter sido criada há diversos anos, ainda é considerada um importante mecanismo para melhorar a prática de Engenharia de Software. De acordo com a literatura, repositórios de reuso colaboram para o sucesso de programas de reutilização, porém outros mecanismos devem estar associados para que o reuso seja realizado de maneira sistemática, como é o caso da mudança cultural. Apesar do código fonte ser o ativo mais comumente reutilizado pelas empresas de software, outros tipos de ativos também são reutilizados, como modelos de processos de negócios, soluções de análise e de projeto, casos de teste, dentre outros. Devido a isso, é necessário utilizar geralmente diversos tipos de repositórios tornando a prática do reuso uma tarefa custosa, propensa a erros e redundante no armazenamento de informações em diversos locais e em formatos diferentes; ressaltando a importância do uso de um único repositório. Porém, a seleção ou o desenvolvimento de um único repositório adequado às necessidades da empresa não é uma tarefa trivial, visto que os interessados podem não ter conhecimento suficiente para isso. Nesse contexto, é importante que o conhecimento sobre os ativos reutilizáveis de software e sobre a gestão desses ativos seja difundido. Sob essa perspectiva, este trabalho de mestrado define uma ontologia de ativos reutilizáveis, denominada ONTO-ResAsset, tendo em vista que ontologia é uma técnica robusta de conceitualização do conhecimento. Para apoiar o desenvolvimento da ONTO-ResAsset é utilizada a metodologia *Methontology* e sua avaliação é conduzida sob o ponto de vista de especialistas no domínio e não especialistas no domínio (ou seja, voluntários de uma equipe de desenvolvimento de software de uma empresa real e alunos de pós-graduação da Facom/UFMS). A avaliação dos especialistas no domínio permitiu o refinamento da ONTO-ResAsset principalmente em relação a representação dos conceitos no modelo conceitual e a revisão dos glossários de termos; e a avaliação dos não especialistas no domínio permitiu observar o compartilhamento e a difusão do conhecimento sobre especificação e gestão de ativos reutilizáveis por meio da ontologia.

Palavras-chave: Ativo de Software, Gestão de Ativo de Software, Repositório, Ontologia, Reutilização de Software.

Silva, L. E. P. **ONTO-ResAsset: Ontology for Reusable Assets**. Dissertation (Mestrado em Ciência da Computação) – Faculdade de Computação. Universidade Federal do Mato Grosso do Sul, Campo Grande-MS, 2013.

ABSTRACT

The software reuse, in despite to be created several years ago, is still regarded as an important mechanism in the improvement of Software Engineering practice. According to the literature, there are several factors that affect the success of software reuse programs. Although the source code have been the most reused asset on software organizations, other types of assets are also reused as business process models, analysis and project solutions, test cases, among others. Therewith, it is frequently necessary to use several kinds of repositories, becoming the reuse activity an expensive task that is error prone, redundant on information storage; what emphasis the importance of a single repository. Nevertheless, the development or selection of an adequate unique repository that meets the enterprise needs is not a trivial task, since the stakeholders may not have the sufficient knowledge to do it. In this context, it is important that the knowledge about the reusable software assets, and about the management of its assets be widespread. In this perspective, regarding ontology as a strong technique for knowledge conceptualization, this master dissertation defines an ontology of specification and management of reusable assets called ONTO-ResAsset. In order to support the Onto-ResAsset development, the *Methongology* methodology is applied and its evaluation is conducted by expert's domain viewpoints, and by a non-expert in the domain (software development team of a real development enterprise and postgraduate students of Facom/UFMS). The experts' evaluation enables the improvement of ONTO-ResAsset, mainly on relation to the concepts representation of the conceptual model and the revision of the terms glossary; and the evaluation of the non-experts allowed the observation of the sharing and the knowledge widespread about the specification and the management of the reusable assets with the ontology.

Keywords: Software Asset, Software Asset Management, Software Repository, Ontology, Software Reuse.

LISTA DE FIGURAS

Figura 2.1 - Exemplo de ontologia representada por meio de mapa conceitual 8
Figura 2.2 - Passos da *Methontology* (adaptado de Férnandez-lópez *et. al*., (1997)) 11
Figura 2.3 - Artefatos que compõem o glossário de termos da *Methontology* (adaptado de Férnandez-lópez *et. al*., (1997)) 13
Figura 3.1 – Visão geral da especificação de ativo (adaptado de OMG (2005)) 16
Figura 3.2 - Pontos-chave dos ativos reutilizáveis (adaptado de OMG (2005)) 17
Figura 3.3 - Modelo de especificação de ativo (adaptado de Ezran *et al*. (2002)) 20
Figura 3.4 - Primeiro nível da documentação do RAS (OMG, 2005) 22
Figura 3.5 - Segundo nível da documentação do RAS (OMG, 2005) 23
Figura 3.6 - Tipos de atividades de desenvolvimento de software associadas à solução do RAS (OMG, 2005) 24
Figura 4.1 - Processo de produção-gerenciamento-consumo (adaptado de Apperly (2001)) 27
Figura 5.1 - Iterações para a definição dos termos do domínio de interesse 37
Figura 5.2 - Representação dos relacionamentos do RAS refinados na ONTO-ResAsset 52
Figura 5.3 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <AtividadeConstrucao> 53
Figura 5.4 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <Solucao> 55
Figura 5.5 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <Artefato> 56
Figura 5.6 - Tipos de <Artefato> 57
Figura 5.7 - Parte do modelo conceitual com foco no conceito <Ativo> 59
Figura 5.8 - Parte do modelo conceitual com foco no conceito <PlanoGestao> 62
Figura 6.1 - Perfil dos avaliadores do grupo 1 (desenvolvedores) 70
Figura 6.2 - Tempo de experiência em relação às fases do desenvolvimento de software do grupo 1 (desenvolvedores) 71
Figura 6.3 - Apresentação das taxas de acerto das questões da avaliação - grupo 1 (desenvolvedores) 71
Figura 6.4 - Taxa de acerto por questão grupo 1 (desenvolvedores) 72
Figura 6.5 - Opinião sobre a ONTO-ResAsset - grupo 1 (desenvolvedores) 73
Figura 6.6 - Perfis dos avaliadores do grupo 2 (Alunos) 74
Figura 6.7 - Tempo de experiência em relação às fases do desenvolvimento de software do grupo 2 (alunos) 75
Figura 6.8 - Apresentação das taxas de acerto das questões da avaliação grupo 2 (alunos) 75
Figura 6.9 - Taxa de acerto por questão grupo 2 (alunos) 76
Figura 6.10 - Opinião sobre a ONTO-ResAsset para grupo 2 (alunos) 77
Figura C.1 - Modelo Conceitual da ONTO-ResAsset 99

LISTA DE QUADROS

Quadro 4.1 - Resumo do processo gestão de ativos reutilizáveis 26
Quadro 4.2 - Comparação entre funcionalidades de repositórios de projetos e de componentes de software com requisitos de repositórios de reuso 32
Quadro 5.1 - Atividades para o desenvolvimento de software 39
Quadro 5.2 - Glossário dos termos das atividades fundamentais de desenvolvimento de software 40
Quadro 5.3 - Glossário dos Artefatos de software identificados 42
Quadro 5.4 - Glossário dos termos da especificação de ativos reutilizáveis 43
Quadro 5.5 - Glossário dos termos da gestão de reuso ativos 46
Quadro 5.6 – Glossário dos novos conceitos definidos 48
Quadro 5.7- Alterações de relacionamentos entre elementos do RAS 51
Quadro 5.8 - Relacionamento entre conceitos com foco em <AtividadeConstrucao> 54
Quadro 5.9 - Relacionamento entre os conceitos com foco em <Solucao> 55
Quadro 5.10 – Conceitos que representam os tipos de <Artefato> 56
Quadro 5.11 - Relacionamento entre os conceitos com foco no <Artefato> 58
Quadro 5.12 - Relacionamento entre os conceitos com foco no <Ativo> 60
Quadro 5.13 - Relacionamento entre os conceitos com foco no <PlanoGestao> 63
Quadro 6.1 - Lista de documentos utilizados durante a avaliação (etapas 1 e 2) 66
Quadro A.1 – Funcionalidades comuns aos repositórios de projeto de software 90
Quadro A.2 - Artefatos e metadados comuns aos repositórios de projeto de software 91
Quadro A.3- Funcionalidades comuns aos repositórios de componentes de software 92
Quadro A.4 - Metadados dos Componentes de Software 93
Quadro B.5 - Lista de conceitos da ONTO-ResAsset em ordem alfabética 94

SUMÁRIO

CAPÍTULO 1. INTRODUÇÃO 1
1.1. Contextualização 1
1.2. Motivação e justificativa 2
1.3. Objetivos 4
1.4. Organização do texto da dissertação 4

CAPÍTULO 2. ONTOLOGIA 6
2.1. Considerações iniciais 6
2.2. Definições 6
2.3. Desenvolvimento de ontologia 9
2.3.1. Metodologia *Methontology* 10
2.4. Considerações finais 14

CAPÍTULO 3. REUTILIZAÇÃO DE SOFTWARE 15
3.1. Considerações iniciais 15
3.2. Histórico, definições e conceitos 15
3.3. Especificações de ativos reutilizáveis 20
3.4. Considerações finais 24

CAPÍTULO 4. GESTÃO DE ATIVOS REUTILIZÁVEIS 25
4.1. Considerações iniciais 25
4.2. Processo de gestão de reuso de ativos 25
4.3. Repositórios de reuso 27
4.4. Requisitos de repositórios de reuso 29
4.5. Ontologias no contexto de gestão de ativos reutilizáveis 32
4.6. Considerações finais 33

CAPÍTULO 5. DESENVOLVIMENTO DA ONTOLOGIA ONTO-RESASSET 34
5.1. Considerações iniciais 34
5.2. Planejamento da ontologia proposta 34
5.3. Passo: Especificar e capturar conhecimento 38
5.3.1. Mapeamento das atividades fundamentais do ciclo de desenvolvimento de software 38
5.3.2. Mapeamento dos artefatos de software 40
5.3.3. Mapeamento da especificação de ativos reutilizáveis 42
5.3.4. Mapeamento do processo de gestão de ativos reutilizáveis 45
5.4. Passo: Conceitualizar e implementar 47
5.4.1. Alterações e inclusão de novos conceitos 48
5.4.2. Conceitualização e implementação das atividades fundamentais de desenvolvimento de software 52
5.4.3. Conceitualização e implementação dos ativos reutilizáveis 54
5.4.4. Conceitualização e implementação da gestão de ativos 61
5.5. Considerações finais 64

CAPÍTULO 6. AVALIAÇÃO DA ONTO-RESASSET 65
6.1. Considerações iniciais 65
6.2. Planejamento da avaliação 65
6.3. Primeira etapa da avaliação: Especialistas 66
6.3.1. Sugestões dos especialistas atendidas 67

6.3.2. Sugestões dos especialistas não atendidas 68
6.4. Segunda etapa da avaliação: não especialistas 69
6.4.1. Resultados do grupo 1 – desenvolvedores de software 70
6.4.2. Resultados do grupo 2 – alunos de pós-graduação 74
6.5. Considerações finais 77

CAPÍTULO 7. CONCLUSÃO 79
7.1. Considerações iniciais 79
7.2. Contribuições do trabalho 79
7.3. Dificuldades e limitações 80
7.4. Trabalhos futuros 81

REFERÊNCIAS 83

APÊNDICE A ANÁLISE DE DOMÍNIO DE REPOSITÓRIOS 89
A.1 Análise de Domínio de Repositórios de Projeto de Software 89
A.2 Repositório de Componentes 92

APÊNDICE B GLOSSÁRIO GERAL DE TERMOS DA ONTO-RESASSET 94

APÊNDICE C MODELO CONCEITUAL DA ONTO-RESASSET 99

APÊNDICE D QUESTIONÁRIO PARA AVALIAÇÃO DA ONTO-RESASSET POR ESPECIALISTA NO DOMÍNIO 100

APÊNDICE E QUESTIONÁRIO PARA AVALIAÇÃO DA ONTO-RESASSET POR NÃO ESPECIALISTA NO DOMÍNIO 102
E.1 Questionário avaliação da ONTO-ResAsset 102
E.2 Questionário de Perfil 105

APÊNDICE F QUESTIONÁRIO PARA AVALIAÇÃO DA ONTO-RESASSET POR NÃO ESPECIALISTA NO DOMÍNIO (REFINADO GRUPO 2) 108

Capítulo 1. Introdução

1.1. Contextualização

É notória a importância do reuso na Engenharia de Software, que surgiu no final da década de 60, para superar a crise de software da época (McILROY, 1968). Assim, a reutilização de software se utilizada de maneira planejada e sistemática promove um produto com maior qualidade e correção, uma vez que o ativo reutilizado já foi utilizado e testado previamente. Adicionalmente, promove maior rapidez no tempo de execução de um projeto de software e, consequentemente, redução do seu custo (SINGH, 2010, JAMWAL, 2010).

Salienta-se que a reutilização de software não se limita apenas à fase de implementação do desenvolvimento de software (LUCRÉDIO *et al*. (2008)), mas também pode ser aplicada nas fases anteriores relacionadas ao levantamento de requisitos, análise e projeto, podendo se estender na fase de manutenção e também nas atividades de apoio (gerência de projeto, documentação, verificação, validação e teste, inspeção, dentre outras).

Mesmo após 40 anos do surgimento do termo, a reutilização de software ainda é considerada um importante mecanismo para melhorar a prática de Engenharia de Software. Devido a isso, diversas técnicas de reutilização têm sido propostas na literatura (KRUEGER, 1992; ALMEIDA *et al.*, 2007; SINGH, 2010; SOMMERVILLE, 2011).

Sendo assim é importante a gestão de reuso de ativos (ABNT NBR ISO/IEC 12207, 2009), que geralmente é auxiliada por repositórios de reuso[1] (BERNSTEIN e DAYAL, 1994; FRAKES e FOX, 1995; MORISIO *et al*., 2002; ALMEIDA *et al.*, 2007), os quais apoiam a execução de programas de reuso nas organizações. Porém, é imprescindível também a mudança cultural da equipe de desenvolvimento para que possa levar em consideração o desenvolvimento para o reuso e baseado em reuso (LUCRÉDIO *et al.*, 2008; MORISIO, *et al.*, 2002).

Sob essa perspectiva, existem diversos tipos de repositórios, de acordo com o tipo específico de ativo que gerenciam, por exemplo, repositórios de código fonte (CHACON, 2009), repositórios de requisitos (IBM, 2008), repositórios de modelos (IBM, 2006),

[1] O termo repositório ou repositório de reuso é utilizado neste trabalho para designar repositório de um determinado tipo de ativo e o termo repositório de ativos reutilizáveis é utilizado para indicar um repositório de qualquer tipo de ativo.

repositórios de teste (JORGE, 2012), repositórios de componentes (WERNER *et al.*, 2007), repositório de serviços (SOA Software Inc., 2012), dentre outros; sendo o repositório de código fonte o mais conhecido e utilizado (LUCRÉDIO *et al.*, 2008). Dessa forma, observa-se a importância de um repositório de ativos reutilizáveis, que seja capaz de gerenciar os diferentes tipos de ativos, de acordo com as necessidades e evolução do programa de reuso da empresa, visando facilitar o reuso sistemático e minimizar os custos associados.

Diante do exposto, dada a relevância dos ativos reutilizáveis e da gestão de reuso de ativos, provida pelos repositórios de reuso, é importante que o conhecimento sobre eles seja compartilhado para todos os interessados, ou seja, diretores de empresas, gerentes de projetos, desenvolvedores de softwares (analista de sistemas, projetista, programador, testador) e desenvolvedores de repositórios de reuso.

Assim, é importante o uso de técnicas adequadas para propiciar a documentação, divulgação, compartilhamento e reuso do conhecimento sobre os ativos reutilizáveis e sobre a gestão de reuso dos mesmos para os diferentes tipos de interessados visando obter as vantagens supracitadas, como é o caso de ontologias (GRUBER, 1995; FENSEL, 2003; e DALTIO, 2007), que é a técnica utilizada neste trabalho.

1.2. Motivação e justificativa

Apesar do conhecimento sobre ativos reutilizáveis e da gestão de reuso de ativos ser importante para a prática do reuso e, consequentemente, para obter os seus benefícios, conforme já discutido na seção anterior, esse conhecimento está espalhado e disperso em vários tipos de documentos (por exemplo, artigos, normas, padrões, manuais, etc), sob diferentes formatos e disponível de várias formas. Por exemplo, o conhecimento sobre especificação de ativos reutilizáveis (*Reusable Asset Specification* - RAS) pode ser obtido especialmente em OMG (2005), Ezran *et al*. (2002) e Lucrédio *et al*. (2008); o conhecimento sobre gestão de ativos reutilizáveis e repositórios de reuso podem ser obtidos em Burégio (2006), ABNT NBR ISO/IEC 12207 (2009) e Almeida *et al*. (2007), bem como a partir dos próprios repositórios de reuso existentes.

Ezran *et al.* (2002) e Lucrédio *et al.* (2008) tratam da especificação de ativos de software em um alto nível de abstração e, consequentemente muito simplista, impedindo a obtenção de informação suficiente para representar o conhecimento sobre os ativos de software de forma satisfatória. Por outro lado, o RAS detalha a especificação de ativos em vários níveis

de granularidade e é a fonte de informação mais completa sobre especificação de ativos se comparada com qualquer outra fonte descrita neste trabalho.

Apesar disso, o RAS apresenta algumas inconsistências entre as definições dos elementos da especificação com as suas representações, ilustradas por meio de modelos na notação do diagrama de classes da UML (*Unified Modeling Language*) (BOOCH *et al*., 2006). Além disso, em alguns casos, há incompatibilidade entre os próprios modelos apresentados na especificação.

A gestão de reuso de ativos, provida pelos repositórios de reuso, está implícita nas funcionalidades providas pelos repositórios. Almeida *et al*. (2007) apresentam um conjunto de requisitos que deve ser considerado na construção de um repositório de reuso, tomando como base as suas experiências no desenvolvimento e no uso de repositórios no estado da prática em projetos do grupo *RiSE*[2] (*Reuse in Software Engineering*), porém não explicitam como essas funcionalidades estão associadas aos requisitos de um programa de reuso de ativos.

A ABNT NBR ISO/IEC 12207 (2009) possui um processo bem específico sobre a gestão de reuso de ativos, apresentando um conjunto de tarefas e ações que devem ser realizadas para a execução de uma gestão de ativos coerente e correta.

Não foram encontrados trabalhos que explicitem a especificação dos ativos reutilizáveis e o relacionamento seu relacionamento com um processo de gestão de reuso de ativos levando em consideração o reuso durante as atividades de desenvolvimento de software.

Do ponto de vista de ontologias na área de conhecimento de interesse deste trabalho, existem na literatura trabalhos e pesquisas baseados em ontologias sobre elementos específicos de repositórios de reuso, como é o trabalho de Ma *et al*. (2007), que definem uma ontologia com o intuito de utilizar um mesmo mecanismo de busca para diferentes repositórios, tornando a integração transparente entre eles; e dos trabalhos de Yao e Etzkorn (2004) e Braga *et al*. (2001), que usam ontologias de conceitos de domínios para melhorar a precisão nas buscas e a recuperação de componentes. No entanto, não foi encontrada na literatura uma ontologia cujo escopo seja a especificação e a gestão de ativos reutilizáveis durante o desenvolvimento de software.

---

[2] http://www.rise.com.br/, Acessado em: agosto/2013.

1.3. Objetivos

O objetivo geral deste trabalho é construir uma ontologia semiformal (Seção 2.2 do Capítulo 2) sobre a especificação e gestão de ativos reutilizáveis, denominada ONTO-ResAsset, para que qualquer interessado possa adquirir conhecimento sobre esse domínio por meio de uma única fonte de informação. Assim, a ontologia ONTO-ResAsset pode ser utilizada como base para outras ontologias, para apoiar a definição de critérios para a seleção de repositórios nas empresas, para apoiar o desenvolvimento de repositórios de ativos reutilizáveis, dentre outros.

Os objetivos específicos do trabalho são:

- Definir a ontologia ONTO-ResAsset, de maneira incremental, visando refiná-la a cada ciclo de desenvolvimento;
- Avaliar a ontologia ONTO-ResAsset sob o ponto de vista de especialistas no domínio e de desenvolvedores de software de uma equipe de desenvolvimento de uma empresa real;
- Possibilitar que os interessados no domínio da ONTO-ResAsset obtenham todo o conhecimento a respeito da especificação e da gestão de ativos por meio de uma única fonte de informação; e
- Fornecer subsídios, por meio da ontologia proposta, para que os interessados possam definir critérios para selecionar o repositório de reuso existente mais adequado ou identificar os requisitos necessários para o desenvolvimento de um repositório de reuso.

1.4. Organização do texto da dissertação

O texto desta dissertação está organizado em mais seis capítulos. No Capítulo 2 são apresentados conceitos e definições sobre ontologias para facilitar o entendimento do trabalho proposto, assim como uma descrição sobre a metodologia *Methontology* utilizada para apoiar o desenvolvimento da ONTO-ResAsset.

No Capítulo 3 são abordadas definições de reutilização de software e são apresentados conceitos relevantes para o entendimento deste trabalho. No Capítulo 4 são apresentados conceitos sobre gestão de ativos e sobre repositório de reuso.

No Capítulo 5 é apresentado o desenvolvimento incremental da ONTO-ResAsset com o apoio de passos da metodologia *Methontology*.

No Capítulo 6 é apresentada a avaliação da ontologia ONTO-ResAsset sob a perspectiva de especialistas no domínio, visando identificar a representação adequada dos principais conceitos do domínio de especificação e gestão de ativos; e sob a perspectiva de não especialistas no domínio (desenvolvedores de software e alunos de pós-graduação), com o objetivo de analisar se o objetivo da ontologia, que é divulgar o conhecimento do domínio, foi atendido.

Por fim, no Capítulo 7, são apresentadas as principais contribuições decorrentes da execução deste trabalho, as principais limitações observadas e os possíveis trabalhos futuros que podem ser derivados a partir do trabalho desenvolvido.

Capítulo 2. Ontologia

2.1. Considerações iniciais

As ontologias estão cada vez mais presentes em diversas áreas do conhecimento e são citadas em diversos trabalhos na literatura, principalmente na área de Engenharia do Conhecimento.

Neste capítulo são apresentados os principais conceitos relacionados ao entendimento e ao desenvolvimento de ontologias, pois são de interesse deste trabalho. Na Seção 2.2 são apresentadas as principais definições de ontologias e uma descrição dos principais conceitos envolvidos. Na Seção 2.3 é descrita a metodologia de desenvolvimento de ontologias *Methontology*, utilizada para apoiar a construção da ONTO-ResAsset, proposta neste trabalho. Por fim, na Seção 2.4, são apresentadas as considerações finais deste capítulo.

2.2. Definições

Uma das definições de ontologia, frequentemente utilizada, é a de Gruber (1995), o qual define ontologia como sendo "uma especificação formal explícita de uma conceitualização compartilhada". Daltio (2007) afirma que o termo conceitualização é a consideração mais importante dessa definição, que corresponde a uma representação conceitual de um determinado domínio. Complementarmente, Fensel (2003) declara que conceitualização refere-se a um modelo abstrato de algum fenômeno, pelo qual é possível identificar os conceitos relevantes desse fenômeno.

Fensel (2003) também explica os outros conceitos da definição de Gruber (1995), sendo: especificação e*xplícita* porque os tipos de conceitos usados e as restrições sobre o uso dos mesmos devem ser definidos explicitamente na especificação da ontologia; especificação *formal*[3], pois uma ontologia deve ser compreendida por uma máquina e conceitualização c*ompartilhada*, uma vez que uma ontologia captura um conhecimento consensual, não sendo restrita a apenas um indivíduo. Adicionalmente, Fensel (2003) afirma que o papel de uma

[3] A definição de Gruber (1995) apesar de utilizar o termo *formal* não limita ou excluí ontologias que não sejam formalizadas com prova ou axiomas.

ontologia é facilitar a construção de um modelo de domínio, provendo um vocabulário de termos e os relacionamentos existentes entre eles.

De acordo com Daltio (2007), o conhecimento do domínio é formalizado por pelo menos quatro componentes, mencionados a seguir, apesar de haver uma ampla diversidade de elementos para se representar uma ontologia. Salienta-se que outros trabalhos (CORCHO *et al.*, 2003; GRUBER, 1995) utilizam uma nomenclatura diferente para alguns desses componentes.

1) **Classes**: representam conjuntos ou tipos de objetos (conceitos ou categorias de domínios) organizados, em geral, por taxonomias[4]. Por exemplo, `Código Fonte`, `Requisito` e `Componente` são subclasses da classe `Artefato`.

2) **Instâncias**: modelam os objetos de domínio, que são representados por instâncias das classes. Por exemplo, o código-fonte de algum software, um documento de requisitos, um componente de um determinado sistema.

3) **Propriedades**: representam as características das classes e das instâncias por meio de atributos, que definem o que uma classe deve possuir ou como as classes e instâncias se relacionam entre si. No trabalho de Corcho *et al.*, (2003) propriedades são chamadas de relações, por exemplo, a classe `Artefato` possui a relação `é-parte-de` com a classe `Ativo`, assim como um `Ativo possui` um ou mais `Artefatos`.

4) **Restrições**: são definições abstratas que utilizam as propriedades para descrever os conceitos dos domínios usando condições. Por exemplo, no domínio de especificação e de gestão de ativos reutilizáveis, todo `Ativo` deve ser composto por no mínimo um `Artefato`, gerando a restrição: $\forall\, Ativo\ \exists\, Artefato\ (\min = 1)$. As restrições também podem ser representadas por meio de axiomas, que são utilizados para modelar sentenças sempre verdadeiras (Gruber, 1995).

Um exemplo simples de representação gráfica de ontologia é apresentado por meio de um mapa conceitual, utilizando os componentes supracitados conforme ilustrado na Figura 2.1. Na figura estão representados alguns dos conceitos que fazem parte do domínio de especificação e gestão de ativos reutilizáveis, de interesse deste trabalho, por exemplo *Artefato, Ativo* e *Repositório*. Adicionalmente, são representados os relacionamentos ou propriedades entre os conceitos, por exemplo, *possui*, *éUma, éAuxiliadoPor*.

---

[4] Refere-se a uma estrutura de metadados rotulados, organizada hierarquicamente, que permite o gerenciamento de informação (GILCHRIST e KIBBY (2000) *apud* MARTÍNEZ *et al.* (2004), p. 107).

Na Figura 2.1 é ilustrado apenas um exemplo dos componentes instância e restrição. O trecho de código representa uma instância de um *CódigoFonte* e a restrição $\forall\ Ativo\ \exists\ Artefato\ (min = 1)$ descreve a restrição da propriedade *possui* entre *Ativo* e *Artefato*, indicando que para qualquer *Ativo* existe pelo menos um *Artefato*. Ressalta-se que as restrições, em geral, não são expressas de forma gráfica. Entretanto, é ilustrada uma restrição na Figura 2.1 com o intuito de exemplificar a ontologia.

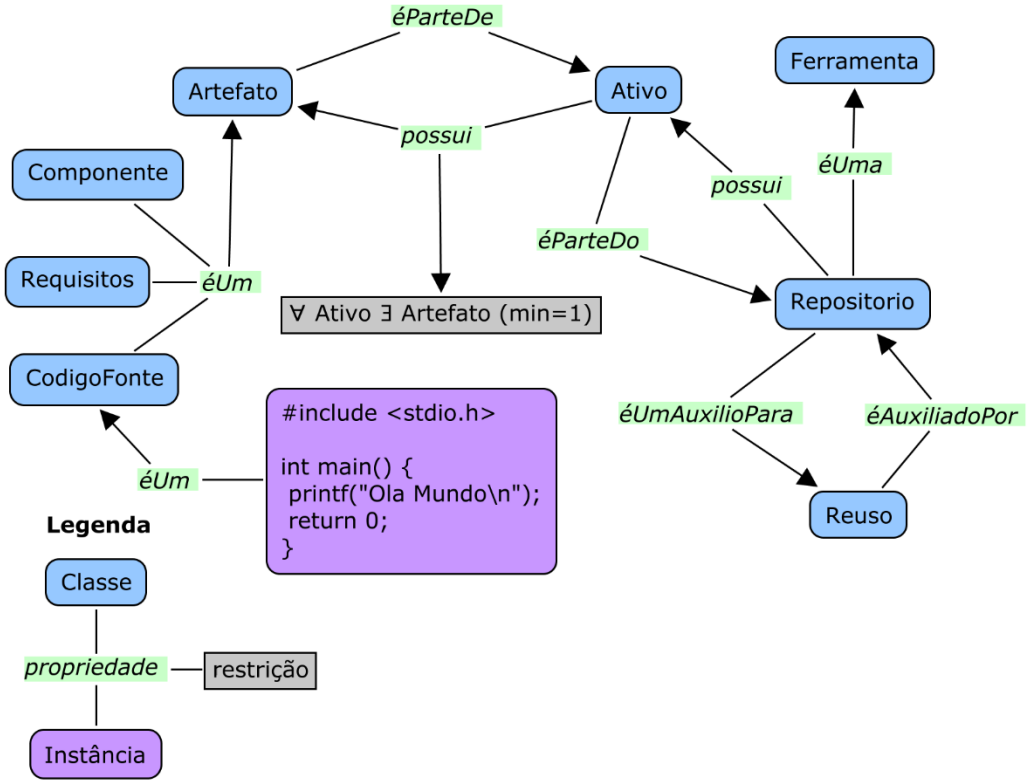


Figura 2.1 - Exemplo de ontologia representada por meio de mapa conceitual

Segundo Uschold (1996), uma ontologia pode ser classificada de acordo com o nível de formalismo, sendo considerada altamente informal quando expressa livremente em linguagem natural, semi-informal quando expressa em linguagem natural mas de maneira estruturada e restrita, semiformal quando expressa em uma linguagem formalmente definida, e rigorosamente formal quando a ontologia é expressa em termos definidos por meio de semântica formal e comprovada por meio de teoremas e provas. Para a representação da ONTO-ResAsset é utilizada uma linguagem formalmente definida, a UML, e também linguagem natural de forma estruturada e restrita, portanto, a ONTO-ResAsset pode ser classificada como semi-informal e semiformal.

2.3. Desenvolvimento de ontologia

Mizoguchi (2004) afirma que o desenvolvimento de ontologias é difícil e está mais para uma arte do que para uma tecnologia. O autor comenta que o desenvolvimento ontológico, em geral, é feito de acordo com as necessidades de quem desenvolve a ontologia e apesar de não existir uma metodologia madura, há diversas propostas na literatura.

Mizoguchi (2004) e Corcho *et al.* (2003) apresentam comparativamente algumas propostas existentes de metodologias para o desenvolvimento de ontologias, como Metodologia TOVE (GRUNNINGER e FOX, 1994), AFM: *Activity-First Method* (MIZOGUCHI *et al.*, 1995), EO – *Enterprise Ontology* (USCHOLD e KING, 1995; USCHOLD, 1996), *Methontology* (FÉRNANDEZ-LÓPEZ *et al.*, 1997) e *On-To-Knowledge* (STAAB *et al.*, 2001). Observa-se que as metodologias existentes foram definidas principalmente na década de 90 e que não há trabalhos mais atuais sobre engenharia ontológica, pois as metodologias propostas se encontram em um estágio maduro e, portanto, nenhuma nova metodologia expressiva foi desenvolvida recentemente por pesquisadores da área de ontologia.

Dentre as metodologias estudadas por Mizoguchi (2004) e Corcho *et al.* (2003), a metodologia *Methontology* foi a selecionada neste trabalho para apoiar a construção da ONTO-RESAsset por ser bem documentada e possuir fases bem distintas e definidas, principalmente por quem não é especialista em ontologia. A *Methontology* também não possui foco no formalismo, como ocorre em metodologias (*TOVE* (GRUNNINGER E FOX, 1994), *AFM: Activity-First Method* (MIZOGUCHI et al., 1995), *EO – Enterprise Ontology* (USCHOLD e KING, 1995; USCHOLD, 1996) e por *On-To-Knowledge* (STAAB et al., 2001)). Para simplificar o entendimento do desenvolvimento da ONTO-RESAsset, apresentado no Capítulo 5, essa metodologia é descrita na seção a seguir.

As ontologias podem ser implementadas por meio de linguagens ontológicas. Essas linguagens permitem o compartilhamento e a representação do conhecimento provido pelas ontologias e possibilitam o seu reconhecimento e entendimento por máquinas ou outros softwares (STUDER *et al.*, 1998). De acordo com Antoniou e Harmelen (2009), uma linguagem ontológica deve possuir os seguintes requisitos: sintaxe bem definida, semântica bem definida (formal), apoio eficiente ao raciocínio (automatizado) e capacidade para representar as expressões ontológicas de forma conveniente.

A necessidade de uma sintaxe bem definida é importante para que a linguagem seja interpretável e reconhecida por uma máquina. A semântica formal, segundo Antoniou e

Harmelen (2009), descreve precisamente o significado do conhecimento para não haver qualquer interpretação subjetiva e diferente entre pessoas e máquinas.

De acordo com Antoniou e Harmelen (2009), o apoio eficiente ao raciocínio pelas linguagens ontológicas (ANTONIOU e HARMELEN, 2009) pode ser entendido como um possível mecanismo de inferência de conhecimento. A importância desse requisito é que um mecanismo automático de inferência de conhecimento pode ser útil em grandes ontologias, bem como pode auxiliar na integração e no compartilhamento de diferentes ontologias a partir de diversas fontes.

As linguagens ontológicas que surgiram mais recentemente e foram influenciadas pela Web Semântica (SU e ILEBREKKE, 2006), são aquelas baseadas em padrões de troca de informações na Internet como XML (*e**X**tensible **M**arkup **L**anguage*) (W3C, 2008) e *RDF* (*Resource Description Framework*) (W3C, 2004). As linguagens *RDF Scheme* (W3C, 2004b), OWL (W3C, 2004a; W3C, 2009), OIL (FENSEL *et al*., 2000) e DAML+OIL (HARMELEN *et al*., 2001) são exemplos dessas linguagens ontológicas mais atuais.

Neste trabalho optou-se por utilizar a notação do diagrama de classes da UML para a implementação da ontologia semiformal ONTO-ResAsset, pois foi possível representar os conceitos e as propriedades presentes no escopo definido para essa ontologia. Outro motivo que levou à escolha do diagrama de classes da UML foi que a principal referência sobre especificação de ativos utilizada como fonte de informação neste trabalho, ou seja, o RAS da OMG (2005), utiliza essa notação para a representação dos elementos que compõem essa especificação. Além disso, pela UML ser uma notação com sintaxe bem definida provê a representação de uma ontologia de maneira semiformal, de acordo com Cranefield e Purvis (1999), Kogut *et al.* (2002) e Guizzardi, *et al*, (2004).

2.3.1. Metodologia *Methontology*

A metodologia *Methontology* (FÉRNANDEZ-LÓPEZ *et al*., 1997) foi desenvolvida pela Universidade Politécnica de Madri e os seus passos foram definidos baseando-se em Uschold e King (1995) e no processo de desenvolvimento do ciclo de vida de software, com o intuito de cobrir o *planejamento, especificação, conceitualização, formalização, integração, implementação, avaliação, documentação e manutenção* de uma ontologia, conforme ilustrado na Figura 2.2.

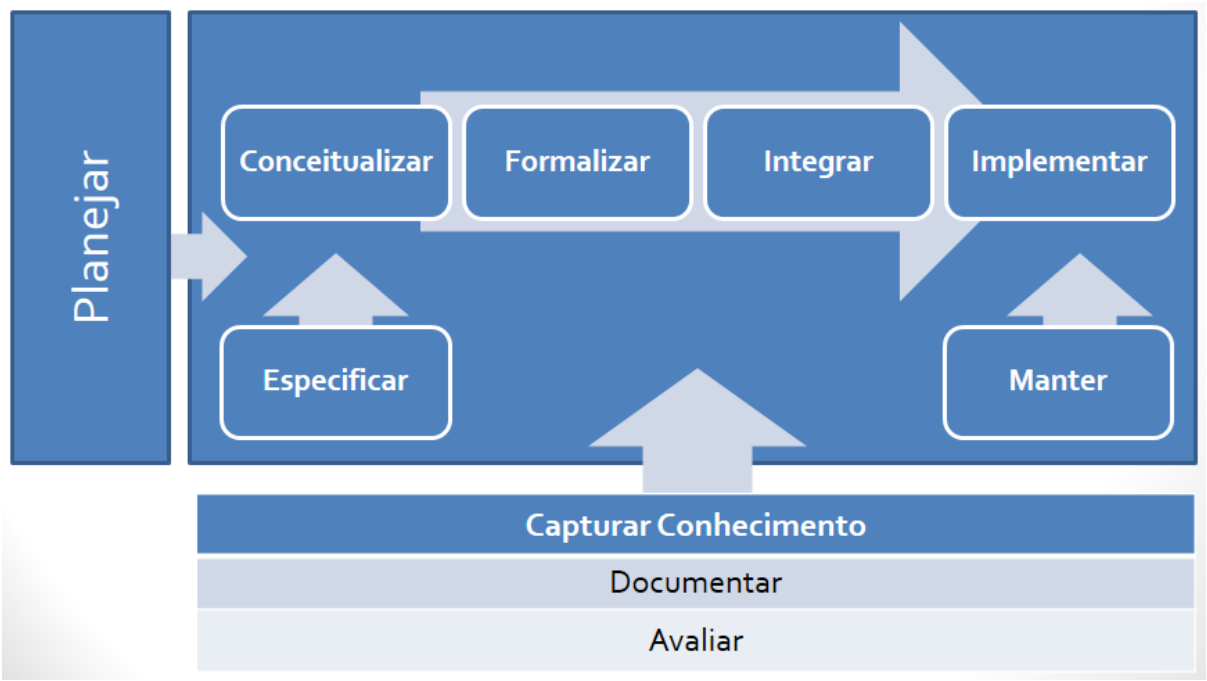


Figura 2.2 - Passos da *Methontology* (adaptado de Férnandez-lópez *et. al*., (1997))

A metodologia *Methontology* é dividida em sete passos principais (Planejar, Especificar, Conceitualizar, Formalizar, Integrar, Implementar e Manter) e três outros passos (Captura de conhecimento, Documentar e Avaliar), que são executados de maneira paralela aos passos principais. Cada um desses passos está descrito a seguir.

1) Planejar: nesse passo devem-se explicitar algumas informações com a finalidade de planejar a construção da ontologia, como o propósito, o nível de formalidade e o escopo da ontologia; bem como podem ser especificados também o tempo e os recursos necessários para o desenvolvimento da ontologia (software, hardware e recursos humanos).

2) Especificar: nesse passo é determinado o porquê da criação da ontologia, as suas finalidades e os seus usuários finais. Adicionalmente, define-se o conjunto de termos que será incluído na ontologia, mesmo se o significado de cada um dos termos ainda não estiver caracterizado. Aconselha-se o agrupamento dos conceitos em árvores de classificação de conceitos. Essa representação intermediária possibilita a identificação precoce de termos relevantes ainda não incluídos, e então realizar sua inclusão ainda durante a fase de especificação; assim como a remoção de termos identificados como sinônimos ou não relevantes à ontologia. Cada termo deve ser conciso, relevante e não representar outro termo já definido, ou seja, não pode haver duplicatas.

3) Conceitualizar: nesse passo é feita a conceituação por meio da criação do glossário de termos do domínio, o qual pode incluir conceitos, instâncias, verbos e propriedades. Nesse

passo o conhecimento do domínio é estruturado em um modelo conceitual que descreve o domínio por meio dos termos já identificados na fase de especificação. Nesse passo, o glossário de termos pode ser elaborado por meio de diversos artefatos descritos a seguir, de acordo com o tipo de conteúdo (conceitos ou verbos), conforme ilustrados na Figura 2.3. Salienta-se que os verbos representam as ações em um domínio.

- Dicionário de Dados: utilizado para a representação dos conceitos do domínio, seus significados, atributos, instâncias, etc;
- Tabela de Atributos de Instância: provê informações sobre os atributos dos conceitos ou os valores das instâncias;
- Tabela de Atributos de Classe: descreve o conceito por si só, na ONTO-ResAsset o equivalente são os quadros apresentados na etapa de especificação e mostrados na Seção 5.3;
- Tabela de Constantes: usadas para especificar informações sobre o domínio que são constantes e não podem mudar;
- Tabela de Instâncias: define as instâncias da ontologia;
- Árvore de Classificação de Atributos: exibe graficamente os atributos e constantes em uma sequência de inferência a partir de um atributo raiz, bem como uma sequência de fórmulas ou regras para executar a inferência de atributos;
- Dicionário de Verbos: expressa o significado dos verbos de uma maneira declarativa; e
- Tabela de Condições: especifica um conjunto de condições que devem ser satisfeitas antes da execução de uma ação, ou um conjunto de condições que devem ser atendidas após a execução de uma ação.

Complementarmente, para a elaboração do glossário de termos, o conteúdo dos artefatos supracitados pode ser utilizados para criar dois outros artefatos:

- Tabela de fórmulas e tabela de regras: que expressam as restrições da ontologia, por exemplo, por meio de axiomas.

Durante o desenvolvimento da ONTO-ResAsset são elaborados apenas os artefatos Dicionários de Dados, Tabela de Atributos de Classe, Dicionário de Verbos e Tabela de Condições definidos neste passo, principalmente devido a caracterização da ontologia proposta, ou seja, semiformal. Como não há necessidade de formalizar a ontologia proposta,

os artefatos Tabela de fórmulas e Tabela de regras são elaborados em linguagem natural estruturada.

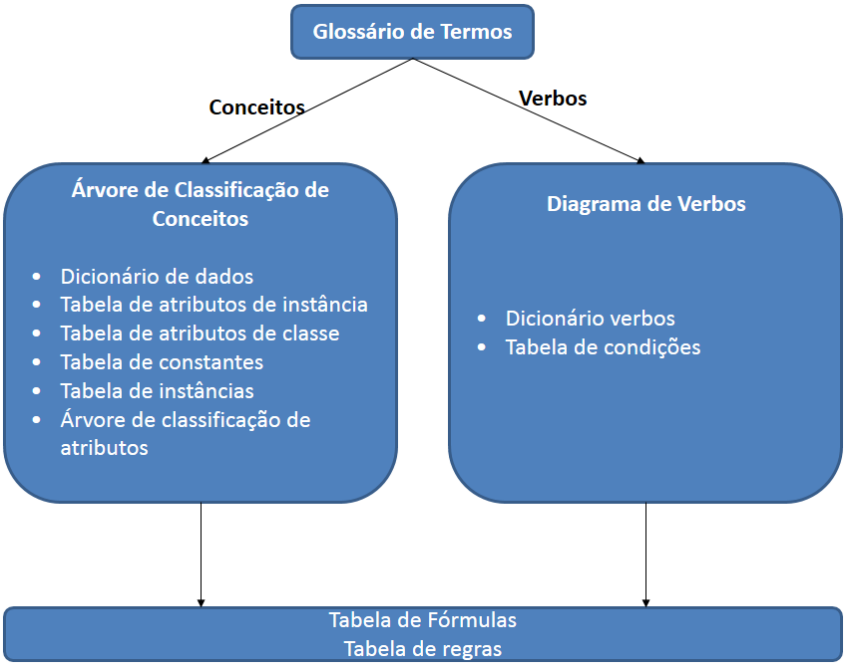


Figura 2.3 - Artefatos que compõem o glossário de termos da *Methontology* (adaptado de Férnandez-lópez *et. al*., (1997))

4) Integrar: nesse passo é realizada a integração de ontologias. Para isso verifica-se a existência de uma ontologia definida para o domínio ou subdomínio de interesse, e a integra, caso necessário, à ontologia em desenvolvimento;

5) Formalizar: nesse passo a ontologia é representada por meio de uma linguagem formal ou matemática, permitindo a inferência por ferramentas computacionais e a verificação e eliminação de contradições e redundâncias na ontologia.

6) Implementar: esse passo é responsável pela implementação da ontologia utilizando alguma linguagem formal;

7) Manter: passo responsável pela manutenção e utilização da ontologia.

8) Capturar conhecimento: esse passo permeia todas as outras fases do processo de desenvolvimento de ontologia, sendo sua execução mais frequente durante o passo especificar e menos frequente na execução dos outros passos subsequentes. Para apoiar esse passo indica-se o uso de várias técnicas: i) *brainstorming,* entrevistas informais com especialistas e inspeção de ontologias semelhantes: para facilitar a criação inicial do glossário de termos relevantes; ii) análise informal de textos: para buscar um conjunto intermediário de representações (conceitos); iii) análise de texto formal: para identificar as estruturas a serem detectadas

(afirmação, definição, etc.), o tipo de conhecimento e a contribuição de cada um deles para a concepção da ontologia (conceitos, atributos, valores e relacionamentos); iv) entrevistas estruturadas com especialistas: para obter conhecimento específico e detalhado sobre conceitos, suas propriedades e seus relacionamentos; bem como avaliar o modelo conceitual elaborado no passo apresentado na Seção 6.3.

9) Avaliar: esse passo apoia a avaliação, verificação e validação da ontologia implementada, no entanto, na documentação da *Methontology* não é apresentado como esse pode deve ser realizado.

10) Documentar: esse passo é responsável pela documentação da ontologia em desenvolvimento, devendo ser realizado por meio da utilização de diversos tipos de artefatos e em todos os passos da metodologia executados.

2.4. Considerações finais

Neste capítulo foram abordados aspectos gerais sobre ontologias, em especial, alguns conceitos básicos e necessários para o seu entendimento, pois são de interesse deste trabalho.

Adicionalmente, foi apresentada a metodologia de desenvolvimento de ontologias denominada *Methontology*, a qual foi a escolhida para apoiar o desenvolvimento da ONTO-ResAsset. Porém, nem todos os passos da *Methontology* são executados durante o desenvolvimento da ontologia proposta e nem todos os artefatos sugeridos são elaborados devido a característica semiformal da ONTO-ResAsset. Além disso, como a *Methontology* não provê detalhes de como realizar uma avaliação de ontologias de maneira sistemática, neste trabalho optou-se por realizar uma avaliação baseada nas visões de especialistas no domínio e de não especialistas no domínio (desenvolvedores de software e alunos de pós-graduação), com o apoio de questionários.

No próximo capítulo é apresentada a revisão bibliográfica sobre a reutilização de software, com enfoque em seu histórico, definições e conceitos, e também são apresentadas duas propostas de especificação de ativos reutilizáveis.

Capítulo 3. Reutilização de software

3.1. Considerações iniciais

Mesmo após 40 anos do surgimento do termo reutilização de software, ele ainda é de interesse tanto da comunidade acadêmica quanto da comunidade industrial devido aos benefícios inerentes a sua prática, como melhoria de qualidade, aumento de produtividade e redução de custos. Pelo fato do interesse deste trabalho estar relacionado à especificação e gestão de ativos reutilizáveis, faz-se necessário um melhor entendimento a respeito do tema.

Na Seção 3.2 é apresentado um breve histórico, bem como as definições e os conceitos comumente encontrados na literatura sobre reutilização de software. Adicionalmente, são apresentados nesta seção os principais benefícios e desvantagens do reuso de software do ponto de vista de alguns autores. Na Seção 3.3 é apresentada a especificação de ativos reutilizáveis usada como uma das principais fontes de informação para a execução deste trabalho e, por fim, na Seção 3.4 são discutidas as considerações finais deste capítulo.

3.2. Histórico, definições e conceitos

O termo reutilização de software surgiu em 1968 durante a *NATO Software Engineering Conference*[5] como uma maneira de superar a crise do software que estava ocorrendo na época. O problema dessa crise estava em como criar softwares em larga escala de maneira controlada e de forma rentável. O trabalho de Mcilroy (1968), apresentado na NATO e denominado *Mass Produced Software Components*, foi o precursor sobre reuso na área de Engenharia de Software. Nesse trabalho foi proposta uma biblioteca de componentes reusáveis e técnicas para automatizar e customizar os componentes em diferentes níveis de precisão e robustez.

Segundo Krueger (1992), no início da década de 90, muitos cientistas da computação ainda consideravam o reuso como um poderoso meio de melhorar a prática da Engenharia de Software e como um mecanismo de amortizar o esforço no desenvolvimento de software.

Apesar da percepção dos benefícios providos pela reutilização de software desde a década de 60, ela ainda não se tornou uma prática padrão no desenvolvimento de software. Em

[5] Primeira conferência da área de Engenharia de Software.

virtude disso, a comunidade científica renovou seu interesse em entender como e onde o reuso pode ser efetivo, e porque foi tão difícil, mesmo sendo aparentemente simples, inseri-lo de maneira sistemática no desenvolvimento de software (KRUGER, 1992).

Alguns autores como Ezran *et al.* (2002) e Caldiera e Basili (1991) apresentam definições similares sobre reutilização de software. Em ambos trabalhos, o reuso é definido como qualquer "coisa" que está ligada ao desenvolvimento de software e pode ser usada mais de uma vez. Ezran *et al.* (2002) complementam essa definição, indicando também o uso de conhecimento ou ativos de sistemas existentes para a construção de um novo sistema. Já a definição de Kruger (1992), que também é a adotada por Almeida *et al.* (2007), especifica o termo reutilização de software como "o processo de criação de sistemas de software a partir de softwares existentes ao invés de construí-los do zero".

É importante salientar que a reutilização de software é alcançada por meio de ativos reutilizáveis. Um ativo reutilizável, segundo a 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, conforme ilustrado na Figura 3.1.

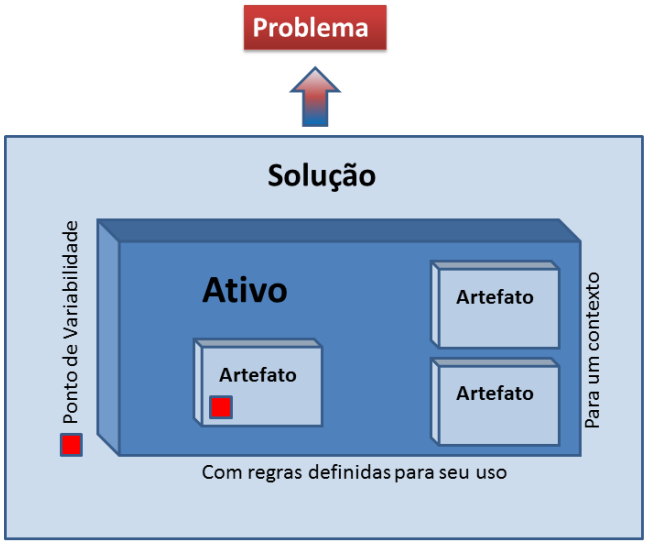


Figura 3.1 – Visão geral da especificação de ativo (adaptado de OMG (2005))

Mais especificamente, um ativo reutilizável é qualquer produto resultante de alguma fase do ciclo de desenvolvimento de software, por exemplo, estruturas de *design*, módulos, especificações, documentação, transformações, fragmentos de código fonte, casos de teste, entre outros. Para Krueger (1992), o código fonte é o ativo mais reusado. Na Seção 5.3 são apresentadas propostas de especificação de ativos reutilizáveis, que é de interesse deste trabalho.

Konda e Mandava (2010) realizaram uma revisão sistemática para identificar os ativos comumente reutilizados, sem levar em consideração o código fonte, pois já se sabe que é o ativo mais reutilizado; bem como os métodos e abordagens utilizados para reusá-los. A identificação dos ativos foi baseada em uma busca realizada em trabalhos publicados na literatura, a partir da qual observaram os seguintes ativos comumente reutilizados: algoritmos, arquitetura, dados, *designs*, documentação, *templates*, interfaces, conhecimento, modelos, módulos, planos de projeto, requisitos, serviços e testes.

Em seguida, Konda e Mandava (2010) estabeleceram uma correlação entre os ativos encontrados e as técnicas, métodos, modelos e métricas aplicados em cada um deles e observaram que ainda há uma distância entre os ativos encontrados nos trabalhos acadêmicos e entre os utilizados na indústria, principalmente em relação às métricas e em relação ao custo e economia atrelados ao reuso.

Os autores chegaram à conclusão que a indústria de software pode e deve utilizar os artefatos, métricas e abordagens de reuso identificados e apresentados no trabalho. Eles evidenciaram também que mesmo já existindo uma especificação de ativos, é necessário que a indústria de software a adote principalmente como suporte para a definição de novos modelos, métodos e abordagens de reutilização de software.

De acordo com a OMG (2005), todo ativo reutilizável é composto de um ou mais artefatos e dado o contexto do ativo, existem tipos específicos de artefatos que o compõe. Sendo assim, a OMG utiliza três pontos-chave para descrever os ativos reutilizáveis, conforme apresentados na Figura 3.2 e descritos a seguir.

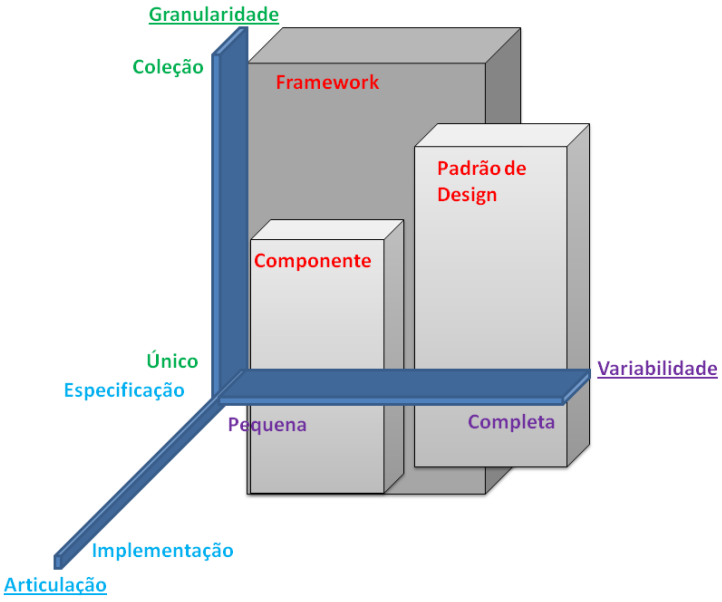


Figura 3.2 - Pontos-chave dos ativos reutilizáveis (adaptado de OMG (2005))

- **Granularidade**: refere-se à quantidade de problemas que um determinado artefato pode resolver. Por exemplo, um componente de software[6] resolve um problema com complexidade menor do que um *framework*[7].
- **Variabilidade:** refere-se à capacidade de um ativo oferecer pontos de variabilidade, ou seja, quanto maior a variabilidade maior é a capacidade do desenvolvedor personalizar o ativo reusado. Nesse aspecto, em especial, a OMG caracteriza os ativos em quatro tipos: i) ativos caixa-preta: sua implementação interna não é conhecida e não podem ser modificados, por exemplo, componentes binários; ii) ativos caixa-branca: sua implementação interna é conhecida e podem ser modificados, por exemplo, código fonte com permissão de alteração; iii) ativos caixa-clara: a implementação é conhecida, mas não podem ser modificados, por exemplo, documentação, modelos; iv) ativos caixa-cinza: a implementação é conhecida, porém apenas determinados subconjuntos de artefatos podem ser modificados, por exemplo, serviços ou *APIs* manipuladas, em geral, por meio de parâmetros.
- **Articulação:** refere-se a capacidade do ativo oferecer a solução, ou seja, quanto mais completo for o ativo para solucionar o problema maior é sua articulação, por exemplo, um ativo que provê os requisitos, casos de testes e a implementação (código) da solução, é mais articulado do que se contivesse apenas os requisitos.

Complementarmente, Almeida *et al.* (2007) classificaram os ativos reutilizáveis em dois tipos: (i) ativos verticais, que são específicos de um determinado domínio, por exemplo, classe `ContaBancária` no domínio Financeiro; e (ii) ativos horizontais, que representam elementos arquiteturais recorrentes e independem de domínios específicos, por exemplo, classes responsáveis pela camada de persistência.

---

[6] Um componente de software é um elemento de software que está de acordo com um modelo de componente e pode ser independentemente implantado e composto sem modificações, de acordo com um padrão de composição (COUNCILL e HEINEMAN, 2001).

[7] Um *framework* é uma aplicação semi-completa reusável que pode ser especializada para a produção de aplicações customizadas (FAYAD e SCHMIDT, 1997).

Do ponto de vista da reutilização de software, Singh *et al*. (2010) identificaram dois tipos de reuso: a) reuso oportunista: em que possíveis ativos que podem ser utilizados para a solução de um determinado problema são encontrados durante o desenvolvimento de um projeto de software; e b) reuso planejado: no qual a equipe de desenvolvimento cria, de forma estratégica, os ativos para que possam ser usados em projetos de software futuros. Os autores ainda subdividem o reuso oportunista em dois subtipos: reuso oportunista interno, em que a equipe reusa seus próprios ativos; e reuso oportunista externo, em que a equipe licencia e/ou utiliza ativos de terceiros integrando-os no desenvolvimento de software.

A maioria dos trabalhos sobre reutilização de software mostra as suas vantagens e desvantagens. Almeida *et al.* (2007) evidenciam que muitas vezes o código é reutilizado de maneira implícita durante o desenvolvimento de software. Consequentemente, não há um aproveitamento efetivo dos benefícios que o reuso planejado e sistemático pode fornecer.

De acordo com Singh *et al*. (2010), a adoção do reuso é capaz de prover diversos benefícios, por exemplo, aumento da confiabilidade no software desenvolvido, pois possibilita reduzir de maneira incremental potenciais falhas futuras visto que, em geral, os ativos reusados já foram testados e utilizados previamente.

Jamwal (2010) confirma os benefícios do reuso, afirmando que um processo de desenvolvimento de software baseado em reuso tem riscos mais reduzidos, pois como os ativos reutilizados já foram desenvolvidos, há menor grau de incerteza nos custos do que haveria se o desenvolvimento dos ativos fosse feito a partir do zero. Isso também reduz, segundo Singh *et al*. (2010), erros na estimativa de custo do projeto, principalmente quando uma grande parte do projeto provém de reuso. Isso torna o software "reusado" disponível em um tempo menor em relação ao software "sem reuso", o que para o mercado global pode ser mais importante do que o próprio custo do software (JAMWAL, 2010).

Outro benefício é que com a prática do reuso durante o desenvolvimento de software, há liberação de tempo que seria despendido pelos desenvolvedores, permitindo que o tempo de trabalho dos mesmos seja empregado na resolução de problemas mais específicos (JAMWAL, 2010).

Ezran *et al*. (2002) comentam que a prática sistemática do reuso no desenvolvimento de blocos com similaridade em requisitos ou arquitetura pode ser explorado para se obter um aumento substancial na produtividade, qualidade e desempenho durante o desenvolvimento futuro de software.

Por outro lado, a reutilização de software possui alguns problemas que devem ser conhecidos como, por exemplo, o custo da manutenção dos ativos reusados pode se tornar cada

vez mais alto. Além disso, os ativos podem se tornar cada vez mais incompatíveis com o sistema de software, ao longo do tempo, de acordo com a evolução do sistema (SINGH *et al.*; 2010; JAMWAL, 2010).

De acordo com Bill Frakes (MOORE, 2001), a dificuldade na reutilização de software se concentra na categorização, armazenamento e recuperação dos elementos que podem ser reutilizados, bem como na falta de compreensão sobre o domínio da aplicação em desenvolvimento. O pesquisador também ressalta que definir um modelo para um domínio muito amplo e complexo é complicado e difícil.

Outro problema está relacionado à restrição de ferramentas CASE, que permitem que o reuso seja realizado de maneira correta e com esforço reduzido. Atrelado a isso, é baixo o povoamento de bibliotecas de ativos reutilizáveis e a reutilização desses ativos é geralmente comprometida pela imaturidade das técnicas de catalogação e de recuperação de ativos que tais bibliotecas proveem (SINGH *et al.*, 2010).

Por fim, do ponto de vista dos desenvolvedores, a reutilização de software é afetada pela síndrome do *não-foi-feito-aqui* (SINGH *et al.*, 2010; Frakes e Fox, 1995), uma vez que eles preferem reescrever todo código a ter que confiar na solução criada e disponível para ser reusada.

3.3. Especificações de ativos reutilizáveis

Ezran *et al.* (2002) definiram um modelo de especificação de ativos reutilizáveis, representado por quatro elementos, conforme ilustrado na Figura 3.3.

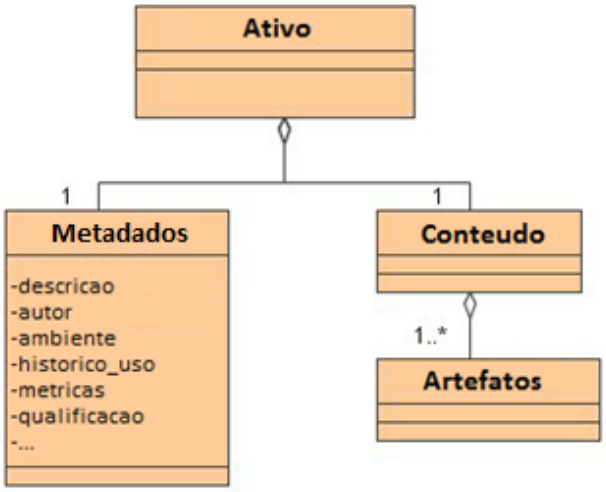


Figura 3.3 - Modelo de especificação de ativo (adaptado de Ezran *et al.* (2002))

O ativo é composto por um metadados e por um conteúdo. O conteúdo é composto por um ou mais artefatos, sendo que cada artefato pode representar a mesma parte do software em diferentes níveis de abstração. O metadado inclui todas as informações que um desenvolvedor necessita para que o ativo possa ser encontrado, recuperado e reusado.

O modelo de especificação de ativos de Ezran *et al.* (2002), apesar de citado por alguns trabalhos, é muito simples e não representa um modelo aceito e completo como um padrão, ao contrário da especificação formal de ativos reutilizáveis de software denominada RAS (*Reusable Assets Specification*) (OMG, 2005). Essa especificação foi definida pela OMG e provê recomendações sobre a estrutura, definição, conteúdo e descrição de ativos reutilizáveis. Essa especificação da OMG será usada como fonte de informação para o desenvolvimento da ontologia proposta neste trabalho.

A documentação da RAS é estruturada de maneira hierárquica, sendo que no primeiro nível são apresentados, de maneira abrangente, os principais elementos que definem um ativo reutilizável e que compõem o núcleo da especificação. Em seguida, cada um desses elementos é detalhado em vários níveis de abstração.

Neste trabalho, são considerados os dois primeiros níveis da hierarquia da documentação da RAS para o mapeamento dos termos da especificação de ativos reutilizáveis, pois fornecem o conhecimento necessário e adequado sobre ativos reutilizáveis dentro do escopo definido no planejamento da ontologia ONTO-ResAsset.

A seguir são apresentados os diagramas dos dois primeiros níveis da RAS utilizados neste trabalho.

Na Figura 3.4 é apresentado o primeiro nível da estrutura da documentação da RAS, denominado RAS *Core*. Ele contém os elementos fundamentais para a especificação de um ativo.

Um Ativo (*Asset*) é uma classe abstrata do metamodelo do *Core* RAS, o qual é composto pela sua Classificação (*Classification*), Solução (*Solution*), Utilização (*Usage*) e Perfil (*Profile*), podendo estar relacionado a vários outros ativos (Ativo Relacionado - *Related Asset*). Cada um desses elementos do ativo, representados no diagrama de classes da Figura 3.4, é discutido a seguir.

- **Classificação:** é um conjunto de descritores que são usados para fazer a classificação do ativo, bem como descreve os contextos nos quais a reutilização do ativo é relevante;

- **Solução:** contém as descrições de um ou mais artefatos que fazem parte do ativo;
- **Utilização:** 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; e
- **Ativo Relacionado:** contempla as descrições dos relacionamentos existentes entre os ativos.

De acordo com a OMG (2005), as informações da RAS são gerais e, para representar a diversidade de ativos, como serviços Web, padrões, componentes e *frameworks* e qualquer outro artefato, é necessário estender o *Core* RAS por meio de **Perfis** (*Profile*). Por exemplo, perfil de componente *default* e perfil de serviços Web. Os perfis derivados podem adicionar elementos e atributos, além de poder adicionar nova semântica aos existentes, mas não podem remover elementos e nem atributos da especificação original.

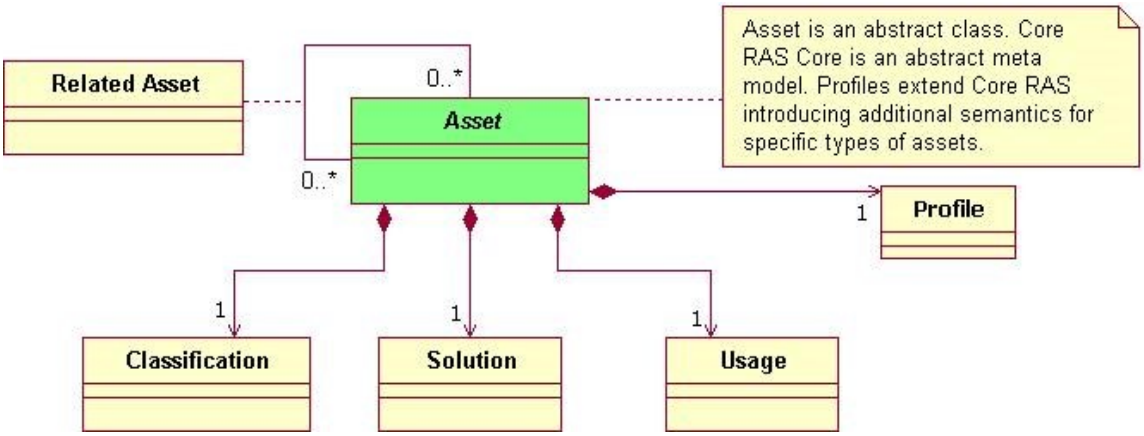


Figura 3.4 - Primeiro nível da documentação do RAS (OMG, 2005)

Um ativo, por ser uma classe abstrata, não é instanciado, portanto o ativo sempre é resultante da instanciação de um perfil específico, sendo que o perfil pode estender tanto o perfil *default* quanto outros perfis, por exemplo.

Todos os elementos desse primeiro nível foram considerados na definição da ONTO-ResAsset, exceto o elemento perfil (*profile*) pois a ontologia proposta representa uma instância de um perfil, no caso, contempla a especificação de ativos reutilizáveis durante as atividades básicas do desenvolvimento de software.

Na Figura 3.5 é apresentado o segundo nível da estrutura da documentação da RAS, denominado *Core+ Profile Default*, *que* contém os principais elementos que fazem parte de

um ativo. Nesse nível, os elementos do primeiro nível são expandidos e novos elementos necessários para a especificação mais detalhada desse segundo nível são apresentados.

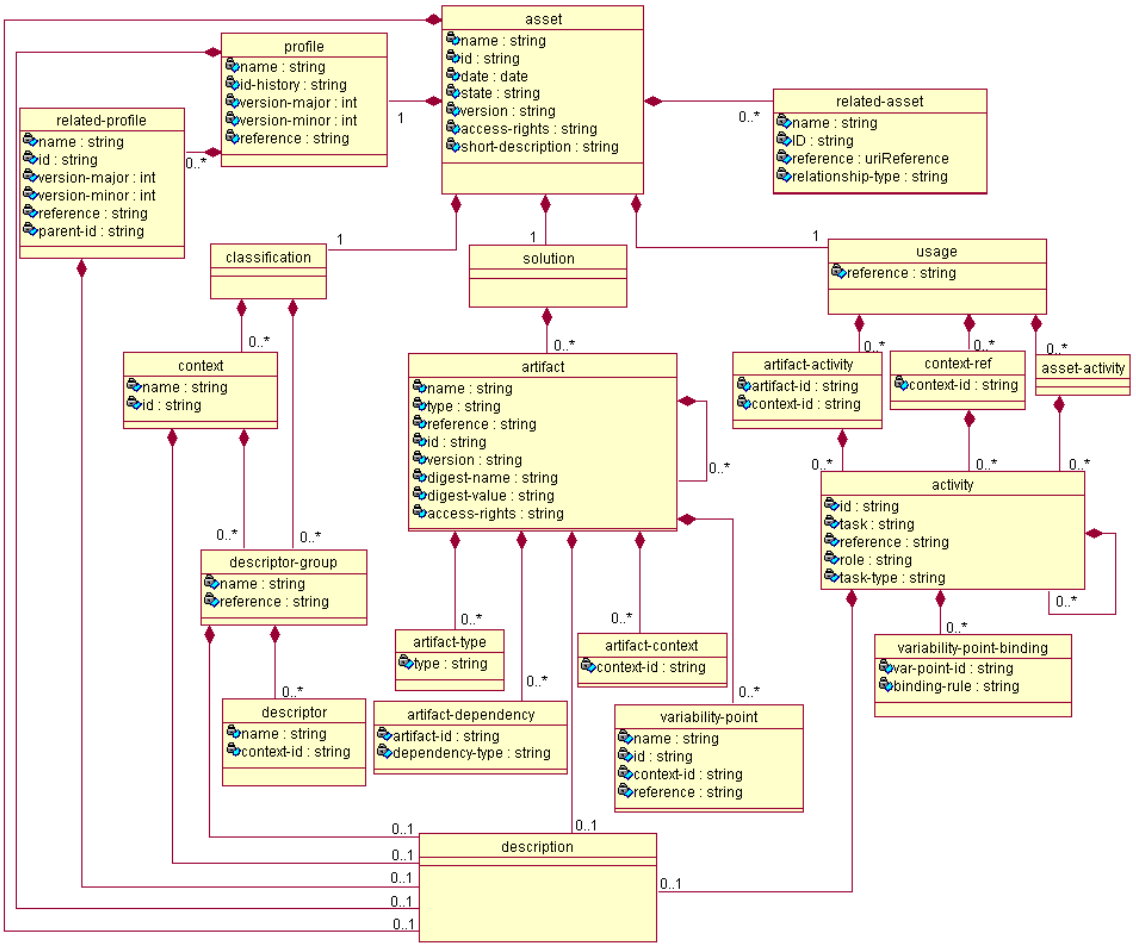


Figura 3.5 - Segundo nível da documentação do RAS (OMG, 2005)

Ainda quanto ao segundo nível da documentação do RAS apenas dois elementos não são considerados na concepção da ontologia proposta, os quais estão elencados a seguir. A descrição dos demais elementos é apresentada na Seção 5.3.3.

- *related-profile*: não é necessário já que o elemento perfil não é considerado na ONTO-ResAssset, conforme justificativa descrita anteriormente.
- *artifact-type*: não considerado pois os tipos de artefatos da ontologia proposta pertencem ao escopo do desenvolvimento de software e, portanto, cada tipo deve ser explicitado.

O elemento *description* permite a incorporação de informações semânticas a todos os elementos do RAS. Na ontologia ONTO-ResAssset esse elemento é utilizado apenas com a

intenção de representar as informações sobre a solução promovida pelo ativo, sendo desconsiderado nos demais conceitos da ontologia.

Adicionalmente, apesar do elemento *profile (*perfil) ser ignorado na construção da ontologia proposta neste trabalho, alguns elementos do perfil padrão do RAS são utilizados neste trabalho pois representam as atividades de desenvolvimento de software que podem estar associadas à solução que um ativo pode oferecer, conforme ilustrados na Figura 3.6. Tais elementos referem-se às atividades fundamentais de desenvolvimento, que estão diretamente ligadas aos objetivos deste trabalho e fornecem base para as definições dos conceitos da ONTO-ResAsset.

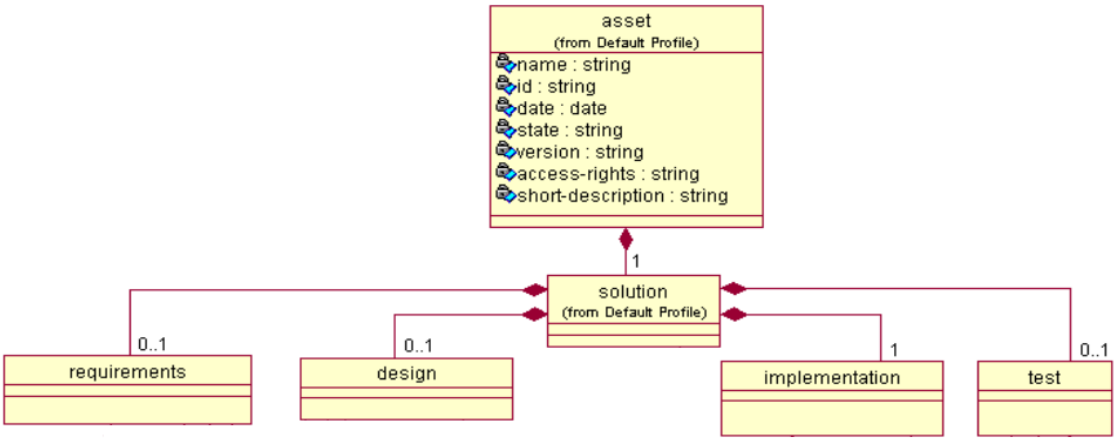


Figura 3.6 - Tipos de atividades de desenvolvimento de software associadas à solução do RAS (OMG, 2005)

3.4. Considerações finais

Neste capítulo foram abordados definições e conceitos relacionados à reutilização de software, que servirá como base para o melhor entendimento do próximo capítulo sobre gestão de ativos reutilizáveis. Adicionalmente, foram apresentadas especificações existentes de ativos reutilizáveis, das quais a especificação de ativos da OMG (2005) é a utilizada como fonte de informação para a concepção da ONTO-ResAsset.

Além do RAS, o trabalho de Konda e Mandava (2010) também é utilizado como fonte de informação neste trabalho pois provê informação sobre os ativos comumente reutilizáveis durante o desenvolvimento de software, bem como a respectiva descrição de cada um deles.

No próximo capítulo aborda-se sobre a gestão de ativos reutilizáveis, apresentando o processo de gestão de reuso da ABNT NBR ISO/IEC 12207 (2009). Adicionalmente, o capítulo discorre sobre repositórios de reuso e seus requisitos, que são ferramentas computacionais de apoio a gestão de reuso; e apresenta trabalhos relacionados sobre ontologias no contexto de gestão de ativos reutilizáveis.

Capítulo 4. Gestão de ativos reutilizáveis

4.1. Considerações iniciais

A reutilização eficiente e adequada de ativos durante o desenvolvimento de software deve ser apoiada por processos de apoio ao reuso, específicos para isso. Tais processos podem ser apoiados por ferramentas computacionais, como é o caso de repositórios que apoiam a gestão de reuso de ativos reutilizáveis.

Apesar de existir diversos trabalhos na literatura sobre repositórios, a maioria tem foco na busca e na recuperação de ativos específicos (componentes, modelos, código fonte, etc.), do que nos elementos necessários para se desenvolver um repositório de reuso, como é o caso de requisitos e arquitetura (BURÉGIO, 2006; ALMEIDA *et al.*, 2007).

Na Seção 4.2 deste capítulo é apresentado o processo de gestão de reuso de ativos da ABNT NBR ISO/IEC 12207 (2009), pois esse tipo de gestão é de interesse deste trabalho. Na Seção 4.3 são apresentadas as definições e os conceitos sobre repositórios, posteriormente na Seção 4.4 são apresentados os principais requisitos de repositórios, de acordo com a proposta de Burégio (2006). Na Seção 4.5 são apresentados trabalhos sobre ontologias no contexto de gestão e de especificação de ativos reutilizáveis a fim de evidenciar as carências observadas na literatura em relação ao tema de pesquisa deste trabalho de mestrado. Por fim, na Seção 4.6, são apresentadas as considerações finais deste capítulo.

4.2. Processo de gestão de reuso de ativos

O processo de gestão de reuso de ativos da norma ABNT NBR ISO/IEC 12207 (2009) possui como propósito o gerenciamento do ciclo de vida de ativos reutilizáveis desde a criação até a desativação dos mesmos. Os resultados, presentes na documentação do processo, descrevem os resultados esperados com a implementação bem sucedida do processo.

Como exemplos de resultados esperados deste processo tem-se: documentação das estratégias de gestão; estabelecimento de um esquema de classificação de ativos; definição de critérios de aceitação, certificação e desativação; uso de mecanismo de armazenamento e de recuperação; registro de uso; e controle de alterações.

Para alcançar tais resultados esperados, a norma apresenta algumas atividades e tarefas que devem ser realizadas. A descrição dessas atividades é apresentada de maneira sucinta e sem detalhes na norma, pois o objetivo é apresentar "o que" deve ser feito e não

"como" deve ser feito, isto é, a forma como cada atividade e tarefa deve ser realizada deve ser estabelecida pela organização de acordo com o seu programa de reuso e recursos disponíveis.

O processo gestão de reuso de ativos também define o papel gerente de ativos e suas principais funções dentro da gestão de ativos, que são: definir um processo de gestão; implementar e manter um mecanismo de armazenamento e recuperação de ativos; desenvolver, documentar e manter um esquema de classificação de ativos; realizar revisões dos mecanismos de armazenamento e recuperação e do plano de gestão de acordo com o processo de revisão de software da mesma norma; e realizar o controle dos ativos.

No Quadro 4.1 é apresentado um resumo dos resultados esperados do processo gestão de reuso de ativos e as respectivas atividades e tarefas, que devem ser executadas para atender cada um deles, de acordo com a norma ABNT NBR ISO/IEC 12207 (2009).

Quadro 4.1 - Resumo do processo gestão de ativos reutilizáveis

Resultados esperados	Atividades	Tarefas
Uma estratégia de gestão de ativos é documentada	Implementação do processo	Criar plano de gestão
		Executar o plano de gestão
		Revisar o plano de gestão de acordo com o Processo de Revisão de Software
Um esquema de classificação de ativos é estabelecido	Definição do armazenamento e recuperação	Implementar e manter um mecanismo de armazenamento e de recuperação de ativo
		Desenvolver, documentar e manter um esquema de classificação
		Revisar o mecanismo de acordo com o Processo de Revisão de Software
Critérios para a aceitação, certificação e desativação de ativos são definidos	Gestão e controle de ativos	Cada ativo deve ser avaliado com base nos critérios de aceitação e certificação de ativos
		Cada ativo aceito deve ser disponibilizado para reuso por meio do mecanismo de armazenamento e recuperação
Um mecanismos de armazenamento e recuperação de ativos é operado		O ativo deve ser classificado de acordo com o esquema de classificação
		Cada ativo deve ser submetido a gerência de configuração, utilizando o Processo de Gerência de Configuração de Software
A utilização dos ativos é registrada		Cada reuso do ativo deve ser registrado e as informações sobre a reutilização real do ativo devem ser relatadas ao engenheiro do domínio
As alterações dos ativos são controladas		As solicitações de modificação e relatos de problemas, recebidos de usuários dos ativos, devem ser encaminhados ao engenheiro do domínio. Em seguida, cada solicitação é revisada e correções ou modificações nos ativos são planejadas pelo engenheiro de domínio
Os usuários dos ativos são notificados de problemas detectados, modificações realizadas, novas versões criadas e exclusão de ativos do mecanismo de armazenamento e recuperação		Aa solicitações a as ações tomadas são monitoradas e registradas
		Todos os usuários dos ativos e o engenheiro do domínio são notificados a respeito de problemas detectados, modificações realizadas, novas versões e exclusão de ativos

A gestão de ativos pode ser feita de forma manual, que é em geral impraticável devido a quantidade de ativos que deve ser gerenciada; e de maneira automatizada, com o apoio de repositórios de reuso. Devido a isso, repositórios de reuso são de interesse deste trabalho e são descritos na próxima seção.

4.3. Repositórios de reuso

De acordo com Bernstein e Dayal (1994), um repositório é um banco de dados compartilhado de informações sobre artefatos de engenharia, de diferentes tipos, produzidos ou utilizados pelas empresas de desenvolvimento de software. Almeida *et al.* (2007) comentam que para entender onde o repositório aparece no processo de reuso é fundamental que se tenha em mente o desenvolvimento para (produção) e com (consumo) reuso. Os autores utilizam o modelo de desenvolvimento baseado em componentes como um exemplo para mostrar o papel que o repositório assume durante o reuso.

De acordo com Apperly (2001), o desenvolvimento baseado em componentes possui três atividades bem definidas: produção-gerenciamento-consumo, conforme ilustradas na Figura 4.1.

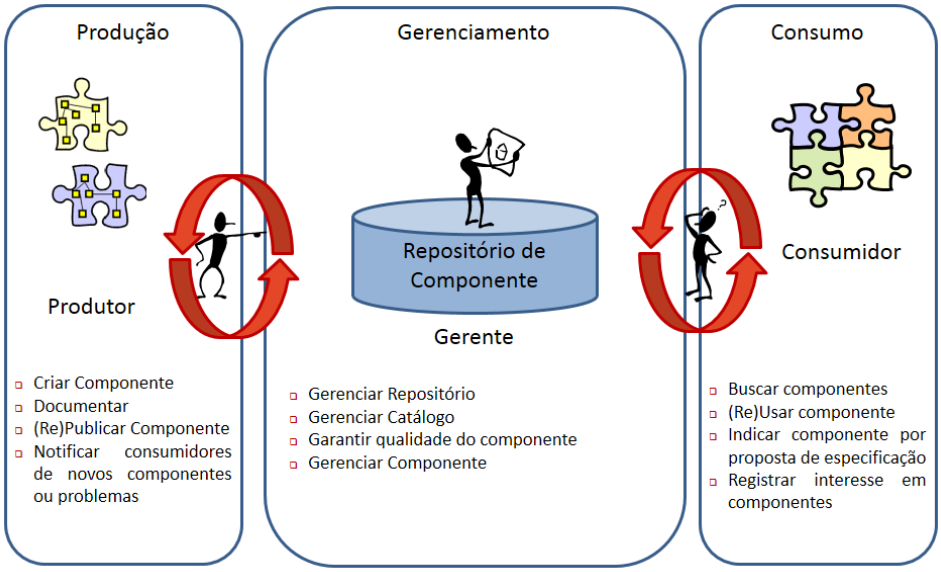


Figura 4.1 - Processo de produção-gerenciamento-consumo (adaptado de Apperly (2001))

A primeira atividade (Produção) é focada na produção e na publicação de componentes e, geralmente, é realizada por produtores de componentes. A segunda atividade (Gerenciamento) tem como objetivo a organização, o controle da disponibilidade e a qualidade dos componentes e, em geral, é realizada por um gerente de projeto. Por fim, a terceira atividade

(Consumo), preocupa-se em encontrar e utilizar os componentes desenvolvidos e é executada por um consumidor de componentes. Analogamente, um repositório de reuso deve prover subsídios para que essas três atividades possam ser realizadas no âmbito de qualquer tipo de ativo reutilizável

Dessa maneira, observa-se a necessidade de um repositório, não apenas para o armazenamento de ativos, mas que ofereça ferramentas que possam resolver os problemas e interesses dos três tipos de *stakeholders* supracitados.

Adicionalmente, de acordo com Apperly (2001), o repositório assume dois papéis: o de barramento de comunicação e o de assistente para o gerenciamento dos componentes armazenados. Sob a perspectiva de qualquer tipo de ativo reutilizável, o primeiro papel atua tanto na produção quanto no consumo de ativos, pois os desenvolvedores de software o utilizam como meio para realizar trocas de componentes produzidos e consumidos. Para desempenhar o segundo papel, o repositório fornece diversas ferramentas de análise e de gerenciamento de ativos para o gerente de projetos.

As definições de ativos reutilizáveis (Seção 3.2) não limitam os ativos a apenas um tipo específico, logo os elementos que um repositório de ativos reutilizáveis deve suportar não pode ficar limitado a um tipo específico de ativo também. Portanto, deve-se haver uma definição mais genérica possível para que quaisquer ativos de interesse possam ser armazenados em um repositório (ALMEIDA *et al.*, 2007).

Assim, é necessário que se utilize um modelo de especificação de ativo completo e efetivo (como é o caso da especificação RAS, apresentada na Seção 3.3) no desenvolvimento de repositórios de reuso para possibilitar o armazenamento, a recuperação e o gerenciamento de diferentes tipos de ativos durante a implantação incremental de um processo de reuso sistemático, principalmente em micro e pequenas empresas de desenvolvimento de software.

Apesar da importância de repositórios para apoiar programas de reuso, na literatura são encontrados poucos trabalhos sobre repositórios de ativos reutilizáveis, principalmente que criam ou definem uma especificação (metadados) dos ativos e que também levem em consideração a gestão do ciclo de vida dos ativos. Isso confirma a importância de um trabalho que explicite o conhecimento sobre especificação e gestão de ativos reutilizáveis para apoiar a construção de repositórios, independente do ativo que pode ser nele armazenado.

4.4. Requisitos de repositórios de reuso

Burégio (2006) e Almeida *et al.* (2007) estabeleceram um conjunto de requisitos que pode ser considerado no desenvolvimento de repositórios. A concepção desse conjunto de requisitos foi baseada em estudos acadêmicos e em experiências práticas dos autores na implementação e uso de repositórios em fábricas de software no Brasil. Os autores mencionam que esse conjunto de requisitos não é considerado completo ou final, que deve estar presente em todos os repositórios. Entretanto, pode servir como base para a construção de novos repositórios. A seguir estão listados os requisitos de repositórios estabelecidos pelos autores e que são considerados como fontes de informação da ontologia proposta:

- Inserir ativos: todo repositório deve possuir alguma maneira dos produtores inserir novos ativos reutilizáveis;
- Publicar especificação de ativos: o repositório deve possibilitar a publicação apenas dos metadados de ativos ainda não fornecidos pelo repositório, para que produtores possam se interessar em desenvolvê-los;
- Navegar pelos ativos: o repositório deve permitir uma maneira de navegar pelos ativos, podendo oferecer a possibilidade de agrupamento de ativos em categorias;
- Buscar ativos: essa talvez seja uma das tarefas mais importantes que um repositório deve realizar, principalmente para repositório com grande número de ativos, sendo que o ideal na busca dos ativos é que o usuário possa encontrar com facilidade os ativos que solucionam seus problemas. O repositório deve dar a possibilidade da busca ser realizada em texto puro, palavra-chave e classificação baseada em facetas[8];
- Gerar relatórios: o repositório deve oferecer ferramentas para visualizar informações de como está sendo utilizado, por exemplo, as buscas mais comumente realizadas pelos usuários, os ativos mais usados, etc;
- Notificar usuários: o repositório deve possibilitar que os usuários registrem seus interesses em determinados eventos vinculados aos ativos e recebam as notificações de quando tais eventos ocorrerem;

[8] Facetas são categorias de palavras e frases que pertencem ao domínio (Frakes *et al*., 1998).

- Oferecer serviços administrativos: o repositório deve oferecer serviços para a manutenção do repositório, por exemplo, cadastro dos usuários, cadastro de categorias de ativos, dentre outros;
- Oferecer versionamento de ativos: o repositório deve possibilitar o controle de versões dos ativos, bem como a recuperação de versões anteriores dos ativos;
- Gerenciar dependência entre ativos: o repositório deve permitir a inclusão de dependência entre os ativos, por exemplo, "usa" ou "é composto de";
- Oferecer integração do repositório com uma IDE (*Integrated Development Environment*): umas das funcionalidades mais comuns e úteis que um repositório pode fornecer é a capacidade de integração com IDEs. Isso facilita a busca, seleção e o reuso de ativos pelos desenvolvedores;
- Registrar *feedback*: o repositório deve permitir que os usuários forneçam um retorno sobre os ativos que estão reutilizando. Isso permite uma avaliação do ativo assim como uma análise qualitativa dos mesmos;
- Registrar anúncios (*Anunciador*): o repositório deve possibilitar a divulgação de informações sobre o reuso (os ativos mais reusados, boas práticas de reuso, os melhores produtores de ativos, etc) a fim de melhorar a cultura de reuso dentro de uma organização;
- Suportar ativos de diversas fontes: o repositório deve oferecer um suporte para propiciar o reuso de ativos de diversas fontes de ativos (ou seja, armazenados em outros repositórios);
- Oferecer apoio a processo de certificação de ativos: o repositório deve suportar os processos de certificação para assegurar a qualidade dos ativos que eles armazenam;
- Controlar acesso de ativos: o repositório deve possibilitar o controle de "quem" acessa "o que" e deve permitir a definição de diferentes visões de usuários; e
- Coletar Métricas: o repositório deve oferecer o suporte para a coleta de informações para a melhoria do repositório e do programa de reuso, por exemplo, custo/benefício, ROI (*Return Of Investiment*), satisfação dos usuários, qualidade do repositório, etc.

Com a intenção de identificar a especificação e a gestão de ativos comumente utilizadas por repositórios de reuso, foi realizada uma análise de domínio tomando como base três repositórios de projetos de software: Google *Code*, *Github* e *SourceForge*, utilizados nas comunidades de desenvolvimento de código aberto; e três repositórios de componentes de software: Eclipse *Marketplace*, Brécho e o *Component Source*.

O detalhamento da análise de domínio dos repositórios de projeto de software e dos repositórios de componentes é apresentado no Apêndice A. Salienta-se que as informações utilizadas na análise do domínio foram coletadas por meio da execução dos repositórios, como usuário comum, assim nem todas as funcionalidades, como instalação ou atividades de gerenciamento dos repositórios em si puderam ser analisadas em detalhes.

A partir da análise de domínio dos repositórios de projeto e de componentes de software, foi conduzida uma análise comparativa para averiguar a correspondência das funcionalidades comumente encontradas em repositórios de projeto e de componentes de software em relação aos requisitos de repositórios de reuso, indicados por Burégio (2006) e por Almeida *et al.* (2007), e em relação as atividades de gestão de ativos reutilizáveis da norma ABNT NBR ISO/IEC 12207 (2009), conforme ilustrado no Quadro 4.2.

Com base na análise comparativa, observou-se que os repositórios considerados na análise de domínio não contemplam todos itens dos trabalhos utilizados para a comparação, ou seja, os requisitos de repositórios levantados por Burégio (2006) e por Almeida *et al.* (2007), e atividades e tarefas do processo de gestão reuso de ativos da ABNT NBR ISO/IEC 12207 (2009). Porém, todas as funcionalidades providas pelos repositórios analisados estão mapeadas nos trabalhos considerados na análise. Com isso, observa-se a importância de usar esses trabalhos como fontes de informação para o desenvolvimento da ONTO-ResAsset.

Algumas funcionalidades observadas como "*Editar Códigos*" e "*Editar Wiki*" possuem comportamentos que podem ser mapeados para mais de um elemento dos trabalhos utilizados na comparação. O requisito Estratégia de Classificação é contemplado indiretamente por todos os repositórios analisados, mas a tarefa de como classificar está implícita e, portanto, não foram identificadas funcionalidades nos repositórios que possam ser associadas a esse requisito.

Quadro 4.2 - Comparação entre funcionalidades de repositórios de projetos e de componentes de software com requisitos de repositórios de reuso

Requisitos de repositórios de reuso (BURÉGIO, 2006; ALMEIDA *et al.*, 2007) e Atividades do processo de gestão de reuso ativos (ABNT NBR ISO/IEC 12207, 2005)	**Funcionalidades comuns a repositórios de projeto de software**	**Funcionalidades comuns a repositórios de componentes de software**
Critério de Certificação	-	-
Estratégia de Classificação	Possuem de maneira implícita	*Listar categorias*
Estratégia de Notificação	*Subscrever Notificações*	-
Mecanismo de Busca	*Buscar Projeto por Palavra* *Buscar Ticket*	*Buscar Componente*
Mecanismo de Dependência	-	-
Mecanismo de Feedback	-	-
Mecanismo de Gerência de Configuração	-	-
Mecanismo de Inclusão	*Atualizar/Fechar Ticket* *Criar Projeto* *Criar Página Wiki* *Criar Ticket* *Anexar Arquivos Ticket* *Editar Códigos* *Editar Wiki*	*Adicionar Componente*
Mecanismo de Métrica	*Estatística de Uso*	-
Mecanismo de Navegação	*Listar Projetos.* *Listar/Navegar Páginas Wiki* *Navegar/Listar Tickets* *Navegar/Ver Códigos* *Navegar/Ver Commits* *Obter Códigos*	*Ver Componente* *Recuperar Componente*
Mecanismo de Publicidade	-	-
Mecanismo de Versionamento	*Ver Histórico* *Diferença dos Códigos* *Diferenciar Commits*	-
Política de Controle de Acesso	*Administrar Projeto* *Compartilhar Projeto* *Sair do Projeto*	*Criar conta* *Editar Conta*
Suporte a Múltiplas Fontes	-	-

4.5. Ontologias no contexto de gestão de ativos reutilizáveis

Alguns trabalhos que tratam de ontologias e estão relacionados a repositórios de reuso foram encontrados na literatura, porém nenhum tem como objetivo representar o domínio de especificação e de gestão de ativos reutilizáveis por meio de ontologias, no entanto utilizam ontologia para resolver problemas pontuais relacionados a repositórios de reuso de software.

Ma *et al.* (2007) definiram uma ontologia como um mecanismo de integração de repositórios de componentes de software existentes. Os autores utilizam a técnica de sistemas em grades (*grids*) para a organização e integração dos repositórios.

Trabalhos como os de Yao e Etzkorn (2004), de Braga *et al.* (2001) e de Ma *et al.* (2007) usam ontologias para melhorar a precisão nas buscas e na recuperação de componentes

em repositórios. Sendo esse um dos tipos de problemas pontuais mais citados pelos trabalhos sobre repositórios de reuso. Dessa forma, em geral, os trabalhos encontrados usam ontologias como um mecanismo para aumentar a semântica dos ativos armazenados visando melhorar a eficiência das buscas e, consequentemente, do reuso. Porém, não foram encontrados trabalhos que utilizam ontologias para representar o domínio de especificação e gestão de ativos reutilizáveis, que é de extrema importância para os diferentes tipos de interessados em programas de reutilização de software.

4.6. Considerações finais

Neste capítulo foram apresentados o processo de gestão de ativos da norma ABNT NBR ISO/IEC 12207 (2009), bem como a definição e os conceitos básicos sobre repositórios de reuso, os quais são apoio computacional para o processo de gestão de ativos. Adicionalmente, são apresentados os principais requisitos presentes em repositórios de reuso, observados a partir de experiências na área e de uma análise de domínio conduzida.

O processo de gestão de reuso de ativos e os requisitos de repositórios de reuso, definidos por Burégio (2006) e Almeida *et al.* (2007), são utilizados como fontes de informação para a construção da ONTO-ResAsset apresentada no próximo capítulo.

Alguns trabalhos que usam ontologias no contexto de gestão de ativos, mais especificamente, repositórios são brevemente apresentados ressaltando a carência na literatura de uma ontologia de especificação e de gestão de ativos reutilizáveis a fim de apoiar o compartilhamento do conhecimento a respeito desse domínio.

No próximo capítulo é apresentado o desenvolvimento da ONTO-ResAsset em conformidade com o planejamento definido e com os objetivos deste trabalho, seguindo os passos especificação, conceitualização e implementação da *Methontology*.

Capítulo 5. Desenvolvimento da ontologia ONTO-ResAsset

5.1. Considerações iniciais

Neste capítulo são descritos os passos da *Methontology* que foram executados para o desenvolvimento da ontologia de ativos reutilizáveis (ONTO-ResAsset), bem como os resultados obtidos.

Na Seção 5.2 é apresentado o planejamento do desenvolvimento da ontologia, com a indicação e justificativa dos passos da *Methontology* selecionados para a criação da ontologia ONTO-ResAsset. É apresentada também o seu escopo e os seus potenciais usuários.

Na Seção 5.3 são indicadas as fontes de informação utilizadas para a aquisição do conhecimento, necessária para a especificação da ontologia. Adicionalmente, é apresentada a identificação dos conceitos da ontologia de maneira incremental, conforme previsto no planejamento.

Na Seção 5.4 são descritos textualmente, em linguagem estruturada, os relacionamentos, multiplicidades e restrições de relacionamentos entre os conceitos identificados na seção anterior, bem como a representação dos mesmos em um modelo conceitual, utilizando a notação do diagrama de classes da UML.

Por fim, na Seção 5.5 são apresentadas as considerações finais deste capítulo.

5.2. Planejamento da ontologia proposta

Como mostrado na Seção 2.3.1, a *Methontology* é definida por uma sequência de passos que permitem a evolução gradativa da ontologia em construção e a incorporação, a todo instante, de novos conceitos à medida que esses passos são realizados.

O passo inicial realizado foi planejar o desenvolvimento da ontologia. Nesse planejamento são definidos a finalidade, o objetivo e os recursos da ontologia.

A finalidade da ONTO-ResAsset é representar o domínio de ativos reutilizáveis no contexto do ciclo de desenvolvimento de software sob a perspectiva das atividades fundamentais, levando em consideração a especificação e a gestão de ativos reutilizáveis.

Busca-se, a partir da definição da ontologia ONTO-ResAsset, responder as seguintes questões de competência (QC) agrupadas por categoria:

Atividades fundamentais de desenvolvimento de software:

QC1. Quais atividades fazem parte da construção de um software?

QC2. Quem realiza as atividades de construção?

QC3. Com qual informação é possível identificar quais artefatos foram criados por uma determinada atividade de construção?

Especificação de ativos reutilizáveis:

QC4. Um ativo de software é composto por quais elementos?

QC5. Como é possível identificar qual problema um ativo propõe resolver?

QC6. Como é possível identificar quais são as estratégias de solução fornecidas por um ativo?

QC7. Os ativos podem prover quais tipos de soluções?

QC8. Uma solução resolve um problema e quem realiza essa solução é um ou mais artefatos. Com que informação é possível identificar quais são os contextos de uma solução do artefato?

QC9. Como é possível identificar quais são os contextos que uma atividade do artefato pode ser executada?

QC10. Como é possível identificar quais são os contextos em que uma atividade do ativo pode ser executada?

QC11. Como é possível identificar quais são as regras de uso que coordenam a execução das atividades de um ativo?

QC12. Como é possível identificar quais são as regras de uso que coordenam a execução das atividades de um artefato?

QC13. Como é possível identificar os contextos da classificação de um ativo?

QC14. Como é possível identificar quais são os descritores que fazem parte de uma determinada classificação de um ativo?

QC15. Como é possível identificar quais são os descritores que fazem parte de um determinado contexto?

QC16. Como é possível identificar quais são os pontos de variabilidade que podem ser personalizados em um artefato?

QC17. Como é possível identificar quais são as regras de customização que devem ser atendidas para personalizar um determinado ponto de variabilidade de um artefato?

QC18. Como é possível identificar a quantidade de problemas que um ativo propõe resolver (o qual determina a granularidade do ativo)?

QC19. Como é possível identificar o número de pontos de variabilidade que os artefatos possuem (o qual determina a variabilidade do ativo)?

QC20. Como é possível identificar a capacidade de articulação de um ativo? Quanto mais completo for o ativo para solucionar o problema, maior é sua articulação. Por exemplo, um ativo que provê os requisitos, casos de testes e a implementação da solução, é mais articulado do que se contivesse apenas os requisitos.

Gestão do ciclo de vida dos ativos reutilizáveis:

QC21. Com qual informação é possível determinar como é feita a gestão de reuso de ativos de software?

QC22. Quais são os mecanismos que podem ser usados para apoiar a realização da gestão de reuso de ativos de software?

QC23. Quais elementos, relacionados diretamente a um ativo de software, são necessários para dar suporte aos mecanismos de gestão de reuso de ativos?

QC24. Com qual informação é possível determinar as versões de um ativo?

QC25. Com qual informação é possível identificar os feedbacks de uma determinada versão de um ativo?

QC26. Como as dependências entre ativos é identificada?

QC27. Quem cria o plano de gestão de ativos de software?

QC28. A partir de qual informação é possível determinar quem pode utilizar cada mecanismo de um repositório?

Os demais passos da *Methontology* selecionados para a definição da ONTO-ResAsset são: especificar, capturar conhecimento, conceitualizar, implementar e avaliar a ontologia. Esses passos foram selecionados neste trabalho, pois conforme discutido na Seção 2.3, auxiliam o desenvolvimento incremental de uma ontologia semiformal, que é o interesse deste trabalho.

Para apoiar a definição da ONTO-ResAsset, os termos do domínio são identificados de maneira incremental nos passos **especificar** e **capturar conhecimento** da *Methontology*, conforme planejamento ilustrado na Figura 5.1 composto por quatro iterações.

A seguir é descrita cada uma das iterações planejadas, que fornecerão subsídios para responder as questões de competência supracitadas conforme indicadas no texto abaixo.

Na primeira iteração dos passos **especificar** e **capturar conhecimento**, cuja execução está apresentada na Seção 5.3.1, são identificadas e definidas as atividades fundamentais que fazem parte do desenvolvimento de software e que são levadas em consideração para a definição da ONTO-ResAsset (questões QC1 a QC3).

Em seguida, na segunda iteração apresentada na Seção 5.3.2, são mapeados os artefatos que podem ser reutilizados durante o desenvolvimento de software e que fazem parte do escopo da ONTO-ResAsset (questão QC3).

Na terceira iteração, cuja execução está apresentada na Seção 5.3.3, é mapeada a representação dos artefatos como ativos reutilizáveis, levando em consideração a especificação de ativos da OMG (2005) e o modelo de especificação de ativos reutilizáveis de Ezran *et al.* (2002) (questões QC3 a QC20).

Por fim, na quarta iteração, cuja execução está apresentada na Seção 5.3.4, são mapeados os processos necessários para a gestão de ativos reutilizáveis tomando como base o processo gestão de reuso de ativos, da norma ABNT NBR ISO/IEC 12207 (2009) (questões QC21 a QC28).

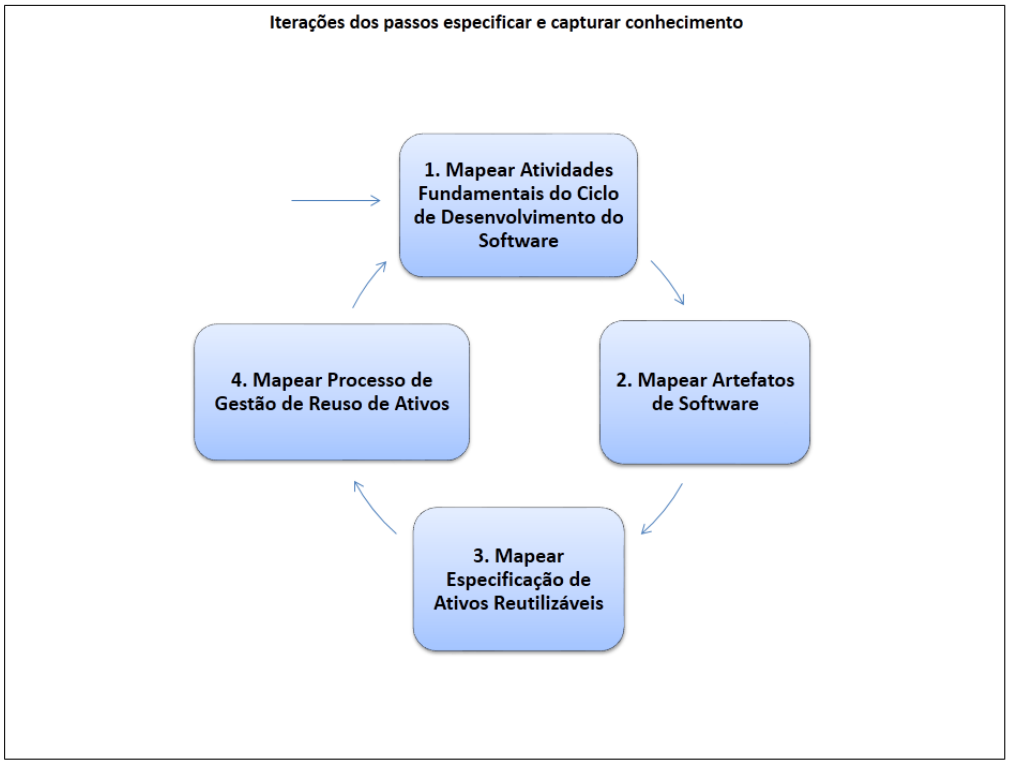


Figura 5.1 - Iterações para a definição dos termos do domínio de interesse

No passo seguinte, ou seja, **conceitualizar**, identificam-se os conceitos, correspondentes aos termos identificados na Seção 5.3, bem como os relacionamentos entre tais conceitos. No passo **implementar** é elaborado um modelo conceitual para representar esses elementos, utilizando a notação do diagrama de classes da UML. Além disso, cada elemento do modelo conceitual é descrito em linguagem natural, obtendo-se uma definição semiformal do domínio.

O passo **avaliar**, que representa a avaliação da ontologia ONTO-ResAsset é apresentado no Capítulo 6.

5.3. Passo: Especificar e capturar conhecimento

O passo capturar conhecimento é realizado de maneira transversal aos demais passos da *Methontology*. Como o objetivo dessa metodologia é a criação da ontologia de maneira evolutiva, o passo capturar conhecimento é realizado a todo instante e, por essa razão, não é descrito separadamente como os demais, sendo consequência da sua execução os elementos resultantes (termos, descrições, conceitos, relacionamentos e atributos) da execução dos passos especificar e conceitualizar.

Nas subseções a seguir é apresentada a identificação dos termos do domínio de interesse, de acordo com o planejamento das iterações explicitado na Seção 5.2. Todos os termos do domínio da ONTO-ResAsset estão apresentados no Apêndice B.

5.3.1. Mapeamento das atividades fundamentais do ciclo de desenvolvimento de software

Para mapear os ativos reutilizáveis durante o ciclo de desenvolvimento de software, inicialmente é necessário identificar e definir as atividades que compõem esse ciclo.

Falbo (1998) define uma ontologia de processo de desenvolvimento software e considera três subdomínios para representá-la: atividades, procedimentos e recursos. Os recursos são equivalentes aos artefatos de software no contexto da ONTO-ResAsset. As atividades são procedimentos realizados por pessoas que podem ter papéis diversos no decorrer do desenvolvimento, sendo cada atividade capaz de gerar ou consumir algum recurso.

Analisando a ontologia de Falbo (1998), observa-se que durante o desenvolvimento de software algumas atividades não estão diretamente associadas à construção do software em si, mas sim relacionadas com a infraestrutura e com as atividades de apoio, denominadas

atividades de gerência de processos e de avaliação de qualidade. Já, outras atividades estão diretamente associadas ao processo de desenvolvimento, que são as atividades fundamentais para a construção do software.

Neste trabalho, estão sendo consideradas como atividades fundamentais do ciclo de desenvolvimento de software aquelas atividades comumente executadas para a construção do software. Essas atividades foram identificadas neste trabalho a partir da análise das atividades de construção da ontologia de processos de software de Falbo (1998), do processo de desenvolvimento da ABNT NBR ISO/IEC 12207 (PAULK *et al.*, 1993, 1995) e do processo de implementação de software da versão mais atual dessa norma (ABNT NBR ISO/IEC 12207, 2009). Além disso, essas atividades podem ser consideradas como atividades básicas de desenvolvimento de software, pois também são comumente utilizadas em modelos tradicionais de processo de desenvolvimento de software, tais como o modelo sequencial e o modelo incremental (PRESSMAN, 2011) e em metodologias ágeis (BECK, 1999; SCHWABER e BEEDLE, 2001; PALMER e FELSING, 2002).

No Quadro 5.1 são apresentadas as atividades comumente utilizadas no desenvolvimento de software de acordo com os trabalhos supracitados.

Quadro 5.1 - Atividades para o desenvolvimento de software

Referência	**ABNT NBR ISO/IEC 12207 (PAULK et al., 1993, 1995)**	**Falbo (1998)**	**ABNT NBR ISO/IEC 12207 (2009)**
Item da referência analisado	Processo de Desenvolvimento das Atividades Fundamentais	Atividades de Construção	Processos de implementação de software
Atividades	Requisitos, Análise, Projeto, Implementação e Teste	Análise e especificação de requisitos, Projeto Arquitetural, Projeto Detalhado e Implementação	Análise de requisitos, projeto arquitetural, projeto detalhado, construção, integração, teste de qualificação.

No Quadro 5.2 é apresentado o glossário de termos relacionados às atividades fundamentais de desenvolvimento de software, bem como a indicação da fonte de informação e uma breve descrição de cada termo.

Na coluna denominada *fonte*, a letra **I** indica que a atividade é identificada e considerada no processo de software definido pela ABNT NBR ISO/IEC 12207 (2009), a letra **F** indica que a atividade é identificada e considerada pela ontologia de processo de desenvolvimento de Falbo (1998).

Quadro 5.2 - Glossário dos termos das atividades fundamentais de desenvolvimento de software

Termo	Fonte	Descrição Semântica (LARMAN, 2007; SOMMERVILLE, 2011)
AtividadeConstrucao	F	Representa as atividades principais de desenvolvimento de software, que são realizadas por pessoas que possuem papéis definidos dentro do processo de desenvolvimento. Como resultado de sua execução, cria ou atualiza um recursos de software <Artefato>.
AtividadeRequisitos	I, F	É uma atividade de construção em que os requisitos do software são levantados por meio da identificação das necessidades de um interessado (*stakeholder*). Em geral, segundo Sommerville (2011), os requisitos são apresentados em documentos de requisitos, em documentos de casos de uso ou em outros tipos de documentos. Essa atividade cria uma solução de requisitos.
AtividadeAnalise	I	É uma atividade de construção onde os requisitos são refinados e normalmente é criado um modelo de classes conceitual, que permite a representação do funcionamento interno do que está sendo desenvolvido. A análise pode ainda ser mais formal quando se cria os modelos de interação (diagramas de sequência ou de comunicação da UML) e o diagrama de estados, gerando como resultados modelos de análise. Essa atividade cria uma solução de análise.
AtividadeProjeto (*Design*)	I, F	É uma atividade de construção em que se define o projeto de software, de maneira a suprir o que até então já foi definido nas atividades anteriores, entretanto nessa atividade o modelo deve descrever características comportamental e estrutural do sistema levando em consideração fatores como a linguagem, sistema operacional, banco de dados, interface do usuário, etc. Nessa atividade devem ser especificados os elementos como arquitetura do software, interface do usuário, componentes, estrutura de dados e algoritmos. Sommerville (2011) indica os seguintes artefatos que podem ser elaborados: modelos de objetos (classes e dependência), de interação entre os objetos, de transição de estados, documentos sobre componentes e agregações e modelos de fluxo de dados. Essa atividade cria uma solução de projeto.
AtividadeImplementacao	I, F	É uma atividade de construção que é responsável pela conversão do projeto do sistema em um sistema executável por meio de uma linguagem de programação. O código-fonte, módulos e componentes são produzidos tomando como base os recursos criados na atividade de projeto, gerando uma solução de implementação.
AtividadeTeste	I	É uma atividade de construção responsável por avaliar, de maneira sistemática, o que foi construído nas outras fases, com o objetivo de averiguar a conformidade com os requisitos, e/ou busca de problemas ainda não descobertos que devem ser corrigidos. Nessa atividade são desenvolvidos o plano de testes e os casos de teste. Essa atividade cria uma solução de teste.

5.3.2. Mapeamento dos artefatos de software

Definidas as atividades fundamentais do ciclo de vida de desenvolvimento, buscam-se informações sobre os artefatos comumente encontrados, que podem ser produzidos ou consumidos em alguma dessas atividades.

As fontes de informação utilizadas para a identificação dos artefatos utilizados durante o desenvolvimento de software foram:

- Revisão sistemática de Konda e Mandava (2010) (indicada por **KM** no texto a seguir): objetiva identificar os ativos de software, citados na literatura, e que são reutilizados de alguma forma durante todo o projeto de software, exceto o código fonte, que foi desconsiderado propositalmente pelos autores;
- Análise de domínio de seis repositórios de reuso (indicada por **AD** no texto a seguir): realizada durante a execução deste trabalho com o objetivo de identificar as principais características de repositórios de reuso, tomando como base três repositórios de projetos de software Google *Code*, *Github* e *SourceForge*, utilizados nas comunidades de desenvolvimento de código aberto, e três repositórios de componentes de software o Eclipse *Marketplace*, Brécho e o *Component Source*. O detalhamento da análise de domínio dos repositórios de projeto de software e dos repositórios de componentes é apresentado no Apêndice A.
- Ontologia de repositórios de modelos de processos de negócios Ferreira e Oliveira (2013) (indicada por **OM** no texto a seguir): representa os conceitos sobre repositórios de modelos de processos de negócios por meio de um modelo conceitual, utilizando a notação do diagrama de classes da UML, a fim de oferecer aos interessados informações a respeito desse tipo de repositório;
- Artefatos utilizados durante o desenvolvimento de software de acordo com o Sommerville, (2011) (indicados por **S** no texto a seguir).

No Quadro 5.3 é apresentado o glossário de termos relacionados aos artefatos de software das atividades fundamentais de desenvolvimento de software, identificadas na Seção 5.3.1, bem como a indicação da fonte de informação de cada termo, uma breve descrição semântica de cada termo e a atividade em que o artefato é produzido ou atualizado.

Salienta-se que os termos dessas fontes de informação que estavam fora do escopo da ONTO-ResAsset, por não estarem relacionados às atividades básicas do desenvolvimento ou por estarem relacionados a paradigmas específicos (por exemplo, *Service Oriented-Architecture* (ERL, 2005)), foram desconsiderados, por exemplo, dados de projeto de software (K), *templates* de estimativas (K), planos (K), *wikis* (AD), contratos de serviços (AD).

A descrição semântica de cada artefato foi baseada nas definições apresentadas por Konda e Mandava (2010), Sommerville (2011) e Delamaro *et al*. (2007).

Quadro 5.3 - Glossário dos Artefatos de software identificados

Termo	Fonte	Descrição Semântica	Atividade básica de desenvolvimento de software
Algoritmo	KM	Artefato que representa a solução para algum problema, em geral utilizado em projetos (*design*) de software.	Projeto
Modelo de Arquitetura	KM, S	É um modelo que mostra a estrutura organizacional de um sistema, software ou componente de software, concebida durante a atividade de projeto.	Projeto
Modelo de projeto *(design)*	KM, S	Modelos que capturam o conhecimento e as decisões de projeto.	Projeto
Modelo de Interface Humana	KM	Artefato que contém informações sobre a interação de software ou hardware com o ser humano.	Projeto
Modelo	KM	É uma abstração de alguma coisa, cujo propósito é permitir que se conheça essa coisa antes de construí-la (BOOCH *et al*. 2006).	Análise, Projeto, Requisitos
Módulo	KM	São arquivos, em geral bibliotecas únicas ou um executável, que contém instruções que representam alguma parte do software.	Implementação
Documento de Requisito	KM	É a informação das capacidades (requisitos) que o sistema tem que atender com o objetivo de satisfazer as necessidades dos seus usuários.	Requisitos
Caso de Teste	KM	É formado por um dado de entrada e uma saída esperada e objetiva efetuar avaliação de software e/ou de componentes de software de acordo com os requisitos pré-determinados. Um caso de teste, ao ser executado, identifica um erro se a saída esperada for diferente da saída resultante da execução do software.	Teste
Código-fonte	AD	Representa a implementação do projeto do software em alguma linguagem de programação.	Implementação
Componente	AD	Representa um conjunto de recursos que encapsula algum requisito e está pronto para ser reusado em outros softwares.	Implementação
Modelo de Processo de Negócio	AD, OM	É a representação dos processos de negócio de alguma organização. Esse modelo pode ser utilizado para apoiar o mapeamento de alguma solução de software para a organização.	Requisitos

5.3.3. Mapeamento da especificação de ativos reutilizáveis

Após identificar os artefatos utilizados pelas atividades básicas de desenvolvimento de software, observou-se na literatura termos relacionados à especificação dos ativos reutilizáveis, os quais são compostos por um ou mais artefatos.

Para isso, foram utilizadas como fontes de informação as seguintes referências da literatura:

- Especificação de ativos reutilizáveis da OMG (2005) (indicada por **OMG** no texto a seguir): Neste trabalho foram considerados os dois primeiros níveis da hierarquia da documentação para o mapeamento dos termos da especificação de ativos reutilizáveis, pois fornecem o conhecimento necessário e adequado sobre ativos reutilizáveis, conforme discutido na Seção 3.3 do Capítulo 3; e
- Modelo de especificação de ativos reutilizáveis de Ezran *et al.* (2002): definem um ativo como uma auto composição, composta também de seus metadados, que contém informações sobre o ativo, como: identificação do ativo, autor, data de criação e atualização, descrição, dentre outras. Como as definições de Ezran *et al.* (2002) estão em conformidade com a especificação de ativos reutilizáveis da OMG (2005), optou-se por citar apenas a referência da OMG no Quadro 5.4 como fonte de informação.

No Quadro 5.4 é apresentado o glossário de termos relacionados à especificação de ativos reutilizáveis, criado a partir dos elementos da especificação de ativos da OMG considerados neste trabalho; bem como a indicação da fonte de informação com o respectivo termo original da especificação de ativos e uma breve descrição semântica de cada termo. Nesse quadro o termo original da especificação de ativos da OMG (2005) foi indicado para facilitar a correspondência com cada conceito da ontologia proposta, visto que essa especificação representa graficamente os termos e os relacionamentos entre eles por meio da notação do diagrama de classes da UML, que é a notação utilizada para representar a ontologia ONTO-ResAsset.

A descrição semântica de cada elemento da especificação de ativos reutilizáveis foi baseada nas definições apresentadas por OMG (2005), Ezran *et al.*, (2002) e Burégio (2006).

Quadro 5.4 - Glossário dos termos da especificação de ativos reutilizáveis

Termo	Fonte	Descrição Semântica (OMG, 2005; EZRAN *et al.*, 2002; BURÉGIO 2006)
Artefato	OMG (*artifact*)	Um artefato é qualquer produto que pode ser criado, armazenado e manipulado por produtores/consumidores ou por uma ferramenta, dentro de um processo de desenvolvimento de software.
Atividade	OMG (*activity*)	São as ações que um consumidor (por exemplo, um desenvolvedor ao utilizar o ativo) ou mesmo uma ferramenta (computacional) pode ou deve realizar quando utiliza o ativo. Uma atividade pode ser composta por outras atividades e cada atividade deve ser desempenhada por um papel. Ex: Um componente *C*, para ser compilado corretamente, necessita que o ponto de variabilidade *X* (que personaliza seu construtor) seja determinado, sendo a atividade *A* quem determina isso.

Termo	Fonte	Descrição Semântica (OMG, 2005; EZRAN *et al*., 2002; BURÉGIO 2006)
AtividadeArtefato	OMG (*artifact-activity*)	São as atividades que estão associadas a um Artefato específico, podem existir diversas atividades para um mesmo artefato, assim como tal atividade do artefato pode ser relevante em um ou mais contextos do artefato.
AtividadeAtivo	OMG (*asset-activity*)	São as atividades que estão relacionadas ao Ativo como um todo, podendo ser agrupada a outras atividades de ativo. Por exemplo, dado o Ativo *A*, ele possui a AtividadeAtivo *Aa* que determina que deve-se ajustar os parâmetros iniciais de construtor dos dois artefatos que compõem *A* (ou seja. os componentes *C1* e *C2*) para um mesmo valor.
Ativo	OMG (*asset*)	É o elemento principal da RAS; ele descreve e empacota os artefatos e provê a documentação sobre como e qual problema ele resolve.
AtivoRelacionado	OMG (*related-asset*)	Um ativo pode estar relacionado a outros ativos. Existem quatro tipos de relacionamentos: agregação, similaridade, dependência, generalização.
Classificacao	OMG (*classification*)	É o estabelecimento das características e comportamentos importantes dos ativos, por exemplo, domínio, autores, palavras-chave, etc. É composto de grupos de descritores.
Contexto	OMG (*context*)	O contexto esclarece onde um ativo é relevante. Por exemplo, um modelo de processo de negócios pertence ao contexto "Negócios" e também ao contexto "Documentação". Isso indica que esse ativo é relevante para o contexto específico de negócios e também colabora para a documentação do sistema.
GrupoDescritor	OMG (*descriptor-group*)	Representa um agrupamento de descritores que ajudam a definir a classificação do ativo e também pode estar relacionado aos contextos do ativo, com objetivo de melhorar a descrição semântica do ativo como um todo e dar mais relevância à informação que o agrupamento busca representar.
Descritor	OMG (*descriptor*)	Descreve os comportamentos e as características do ativo, como: "Autor: Luciano Édipo", "Domínio: Aluguel de Carros", etc.
PontoVariabilidade	OMG (*variability-point*)	Determinam os pontos (locais) nos artefatos onde pode ocorrer uma variabilidade, ou seja, onde pode ser feita uma customização para um contexto específico.
RegraUso	OMG (*usage*)	Representa a informação sobre como aplicar ou usar um ativo ou artefato. Essas instruções podem ser para todo o ativo ou para um artefato específico do ativo; são informações que restringem e definem também como as atividades dos artefatos e as atividades dos ativos serão executadas.
RegraCustomizacao	OMG (*variability-point-binding*)	Representa as regras, restrições e orientações para que o consumidor do ativo possa personalizar o ponto de variabilidade do artefato. Isso é feito durante a execução de atividades do artefato.
Solucao	OMG (*solution*)	Representa a solução fornecida pelo ativo, por meio da coleção de artefatos que o compõe. A solução é fornecida pelo(s) artefato(s) de acordo com o contexto do ativo que o artefato pertence. Existem alguns tipos de solução de acordo com a atividade que cria a solução: requisitos, análise, projeto, implementação e teste.
Descricao	OMG (*description*)	Descreve o problema tratado pelo ativo e as suas principais estratégias de solução.
ContextoArtefato	OMG (*artifact-context*)	Representa o contexto do ativo em que o artefato pode ser utilizado.

Termo	Fonte	Descrição Semântica (OMG, 2005; EZRAN *et al*., 2002; BURÉGIO 2006)
DependenciaArtefato	OMG (*artifact-dependency*)	Um artefato pode depender de outros artefatos. Como exemplos de dependência entre artefatos têm-se: dependência em tempo de compilação, dependência em tempo de execução.
AtividadeArtefatoContexto	OMG (*context-ref*)	Representa um agrupamento do contexto de relevância para um conjunto de Atividades do Artefato. Ou seja, agrupa quais AtividadeArtefato são relevantes em determinados contextos do artefato.
AtividadeAtivoContexto	OMG (*context-ref*)	Representa um agrupamento do contexto de relevância para um conjunto de Atividades do Ativo. Ou seja, agrupa quais AtividadeAtivo são relevantes em determinados contexto do ativo.

5.3.4. Mapeamento do processo de gestão de ativos reutilizáveis

O último grupo de termos do domínio da ontologia ONTO-ResAsset, identificados neste trabalho refere-se à gestão do ciclo de vida dos ativos reutilizáveis. Para isso foram utilizadas as seguintes referências como fontes de informação:

- Processo de gestão de reuso de ativos da norma ABNT NBR ISO/IEC 12207 (2009) (indicada por **ISO** no texto a seguir): descreve os resultados esperados, atividades e tarefas para efetuar o gerenciamento da vida de ativos reutilizáveis desde a concepção até sua desativação.
- Conjunto de funcionalidades de repositórios de reuso definido por Burégio (2006) (indicada por **BUR** no texto a seguir): contém funcionalidades comumente encontradas em repositórios de reuso de acordo com experiências práticas do uso desse tipo de repositório pelo autor. Essa referência foi utilizada como fonte de informação neste trabalho tendo em vista que um repositório de reuso é a ferramenta responsável pelo apoio computacional às atividades do processo de gestão de reuso de ativos (ISO) visando alcançar os objetivos desse processo.

No Quadro 5.5 é apresentado o glossário de termos relacionados à gestão de reuso de ativos obtidos a partir das fontes de informação supracitadas; bem como a indicação da fonte de informação e uma breve descrição semântica de cada termo, a qual foi baseada nas definições apresentadas em ABNT NBR ISO/IEC 12207 (2009) e Burégio (2006).

Quadro 5.5 - Glossário dos termos da gestão de reuso ativos

Termo	Fonte	Descrição (ABNT NBR ISO/IEC 12207 (2009) e Burégio (2006))
PlanoGestao	ISO	Criado pelo gerente de ativos e é onde se definem os recursos e os procedimentos que serão realizados para a gestão do ativo reutilizável. Esse plano deve ser executado pelo gerente de ativos e ser revisado levando em consideração o processo de revisão de software (ABNT NBR ISO/IEC 12207, 2009). No plano de gestão estão descritas todas as informações (critérios, políticas e procedimentos) para a realização da gestão de ativos.
CriterioAceitacao	ISO	São os critérios definidos no plano de gestão para que um ativo seja aceito ou não pelo gerente de ativos, o qual é responsável pela gestão dos ativos.
CriterioCertificacao	ISO, BUR	São os critérios definidos no plano de gestão para que o ativo seja certificado ou não pelo gerente de ativos, ou seja, garantem que o ativo esteja de acordo com a qualidade esperada.
CriterioDesativacao	ISO	São os critérios definidos no plano de gestão para que os ativos sejam descartados.
EstrategiaClassificacao	ISO, BUR	É um documento, criado pelo gerente de ativos, que compõe o plano de gestão e tem por objetivo definir a estratégia para a classificação dos ativos.
EstrategiaNotificacao	ISO, BUR	Definida no plano de gestão e determina quais notificações serão enviadas para os usuários do repositório sobre alterações que ocorrerem nos ativos.
MecanismoBusca	ISO, BUR	É o mecanismo que permite a busca e a recuperação dos ativos no repositório. Existem algumas formas para realizar as buscas, como por exemplo, por meio de palavras-chave, busca textual que pode ser realizada levando em consideração os elementos que compõem a especificação do ativo, ou mesmo por facetas que faz uso de uma combinação de pares (atributo, valor).
MecanismoDependencia	BUR, ISO	É o mecanismo que permite o registro de dependência entre ativos. Essa dependência pode ser: agregação, similaridade, dependência ou origem.
Mecanismo*Feedback*	BUR, ISO	É o mecanismo que permite o registro de impressões dos usuários a respeito da versão do ativo que eles utilizaram.
MecanismoGerenciaConfiguracao	ISO, BUR	É o recurso que permite a realização dos mecanismos de *feedback* e de versionamento com objetivo de executar a gerência de configuração do ativo.
MecanismoInclusao	ISO, BUR	É o mecanismo de armazenamento que permite a inclusão de ativos no repositório.
MecanismoMetrica	ISO, BUR	É o mecanismo do repositório que registra as métricas coletadas sobre a utilização do ativo.
MecanismoNavegacao	BUR	É o mecanismo que, a partir de algum grupo de descritores da classificação (Quadro 5.4), ou de algum método de agrupamento ou listagem simples, permite a navegação entre os ativos armazenados.
MecanismoPublicidade	BUR, ISO	É o mecanismo que oferece informações relativas ao reuso, iniciativas de reuso, ativos mais usados, etc.
MecanismoVersionamento	BUR, ISO	É o mecanismo que permite a criação e o armazenamento de múltiplas versões de um mesmo ativo.
PoliticaControleAcesso	BUR	Determina o acesso aos mecanismos oferecidos pelo repositório, de acordo com o papel que o usuário assume.
Repositorio	BUR	Representa o conceito de um armazém de ativos, que é responsável por prover um conjunto de mecanismos para a gestão e uso dos ativos.

Termo	Fonte	Descrição (ABNT NBR ISO/IEC 12207 (2009) e Burégio (2006))
SuporteMultiplasFontes	BUR	É a capacidade que um repositório deve ter de ser capaz de aceitar múltiplas fontes de origem de ativos, com objetivo de facilitar a integração entre equipes e entre repositórios diferentes.

Foi executada uma análise de domínio de repositórios de reuso (Apêndice A) levando em consideração as funcionalidades identificadas nos repositórios analisados. Observou-se que todos os termos do domínio, resultantes dessa análise, bem como os requisitos comumente encontrados em repositórios de reuso (BURÉGIO, 2006; ALMEIDA *et al.*, 2007) e as atividades do processo de gestão de reuso de ativos (ABNT NBR ISO/IEC 12207, 2009) estão presentes no Quadro 5.5, conforme análise de correspondência descrita no Quadro 4.2 do Capítulo 4. Isso indica que os termos relevantes para a gestão de ativos foram considerados na concepção da ONTO-ResAsset.

5.4. Passo: Conceitualizar e implementar

No passo **conceitualizar** são identificadas e apresentadas as associações que existe entre os conceitos, quem foram extraídas a partir das descrições textuais dos termos do domínio, os quais são mapeados como conceitos do domínio de ativos reutilizáveis. No passo **implementar**, esses elementos foram representados em um modelo conceitual, utilizando a notação do diagrama de classes da UML, o qual está apresentado na íntegra no Apêndice C.

Nesta seção, o modelo conceitual é apresentado por partes. Cada parte representa os resultados obtidos em cada iteração dos passos especificar e capturar conhecimento (Seções 5.3.1, 5.3.2, 5.3.3 e 5.3.4). Os relacionamentos entre os conceitos de cada parte do modelo conceitual e as respectivas restrições, caso existam, são descritos na ONTO-ResAsset.

O nome de cada conceito da ONTO-ResAsset está descrito no texto entre os caracteres "<" e ">". Alguns conceitos da ONTO-ResAsset são conceitos abstratos, por não possuírem instâncias. Como exemplo tem-se <*Artefato*>, <*Solucao*>, <*Modelo*>, <*DependenciaArtefato*>, <*AtividadeConstrucao*>, <*AtivoRelacionado*>, <*Atividade*> e <*Mecanismo*>. No modelo conceitual, conceitos desse tipo são representados com o nome em itálico e com o estereótipo <<conceito abstrato>>. Todos os relacionamentos que os conceitos abstratos participam são transitivos para os conceitos que herdam desse tipo de conceito. Por exemplo, <Componente> herda do conceito abstrato <*Artefato*>, o qual possui relacionamento com o conceito <ArtefatoSolucao>. Dada a instância *c* de <Componente> existe uma <ArtefatoSolucao>, já que *c* é um <*Artefato*> e herda as propriedades desse.

O nome de cada relacionamento entre conceitos, citados ao longo do texto desta dissertação, são representados em itálico, iniciando com letra minúscula e as demais palavras que compõem o nome do relacionamento, possuem a primeira letra maiúscula, por exemplo, o nome do relacionamento *éRelacionadoCom*.

Os nomes dos relacionamentos *éUmTipoDe* e *éCompostoDe*, presentes no Quadro 5.8, no Quadro 5.9, no Quadro 5.10 , no Quadro 5.11, no Quadro 5.12 e no Quadro 5.13, não estão explicitados no modelo conceitual pois a semântica dos relacionamentos de herança e de agregação (simples ou composta) da notação do diagrama de classes da UML já representa o significado dos nomes desses relacionamentos, respectivamente.

5.4.1. Alterações e inclusão de novos conceitos

Durante a análise da especificação RAS (OMG, 2005) para a concepção da ontologia ONTO-ResAsset, observou-se que alguns conceitos e, consequentemente, relacionamentos entre eles não estavam explícitos nos modelos presentes nessa especificação, mas sim apenas implícitos na descrição dos elementos. Isso também ocorreu durante a análise do processo de gestão de reuso de ativos da ABNT NBR ISO/IEC 12207 (2009) e do conjunto de funcionalidades de repositórios de reuso, definido por Burégio (2006), em que novos conceitos e relacionamentos foram abstraídos ou alterados a partir da definição dos termos.

Salienta-se que alguns dos conceitos abstraídos são resultantes do detalhamento de alguns relacionamentos, não estando presentes em nenhuma fonte de informação. Os conceitos abstraídos foram necessários para guiar a conceitualização tendo como objetivo responder as questões de competência levantadas no início deste capítulo. No Quadro 5.6 são apresentados os conceitos abstraídos, que estão agrupados por iteração do desenvolvimento da ontologia ONTO-ResAsset. Salienta-se que na iteração 2 não foram abstraídos novos conceitos.

Quadro 5.6 – Glossário dos novos conceitos definidos

Conceito	Descrição e Justificativa	Origem
Iteração 1: Atividades Fundamentais de Desenvolvimento		
Pessoa	Representa alguém que realiza alguma ação definida na ontologia, por meio de um papel definido.	Obtido da necessidade de expressar a realização das atividades fundamentais de desenvolvimento por um ser humano (Quadro 5.2).
Iteração 3: Especificação de Ativos Reutilizáveis		
ContextoAtivo	Representa um contexto em que o ativo pode ser utilizado.	Conceito observado a partir da abstração do domínio de ativos reutilizáveis, tomando como base as descrições a respeito do elemento "contexto" do RAS. Para ficar coerente com os refinamentos realizados no conceito

Conceito	Descrição e Justificativa	Origem
		<Atividade> (Figura 5.2), houve a necessidade de evidenciar o contexto também para o ativo.
ArtefatoSolucao	Representa a ligação entre Solução e o Artefato que a realiza. Delimita essa ligação de acordo com um contexto de relevância do artefato.	Conceitos abstraídos a partir da descrição semântica dos termos <Artefato> e <Solução> (Quadro 5.4)
Papel	Responsabilidades que uma pessoa assume durante o uso de ativos no desenvolvimento de um software.	Conceito abstraído a partir da descrição semântica do termo <Atividade> (Quadro 5.4)
Problema	Descreve o problema que o ativo soluciona.	Conceitos abstraídos a partir da descrição semântica do termo <Descrição> (Quadro 5.4)
EstrategiaSolucao	Descreve a estratégia para a sua solução do problema.	
SolucaoRequisito	É uma solução criada durante a execução de uma atividade de requisito.	Conceitos abstraídos a partir da descrição semântica do termo <Solução> (Quadro 5.4)
SolucaoAnalise	É uma solução criada durante a execução de uma atividade de análise.	
SolucaoProjeto	É uma solução criada durante a execução de uma atividade de projeto.	
SolucaoImplementacao	É uma solução criada durante a execução de uma atividade de implementação.	
SolucaoTeste	É uma solução criada durante a execução de uma atividade de teste.	
Agregacao	É um tipo de AtivoRelacionado cujo relacionamento denota a ideia de que um dado Ativo contenha outro.	Conceitos abstraídos a partir da descrição semântica do termo <AtivoRelacionado> (Quadro 5.4)
Similaridade	É um tipo de AtivoRelacionado cujo relacionamento denota a ideia de que um dado Ativo possui características similares a outro.	
Dependencia	É um tipo de AtivoRelacionado cujo relacionamento denota a ideia de que um dado Ativo conta com o uso de "serviços" ou artefatos de outro Ativo.	
Descendencia	É um tipo de AtivoRelacionado cujo relacionamento denota a ideia de que um dado Ativo descende ou deriva de um outro Ativo.	
TempoExecucao	É um tipo de DependênciaArtefato que determina que um Artefato depende em tempo de execução, de outro Artefato.	Conceitos abstraídos a partir da descrição semântica do termo <DependenciaArtefato> (Quadro 5.4)

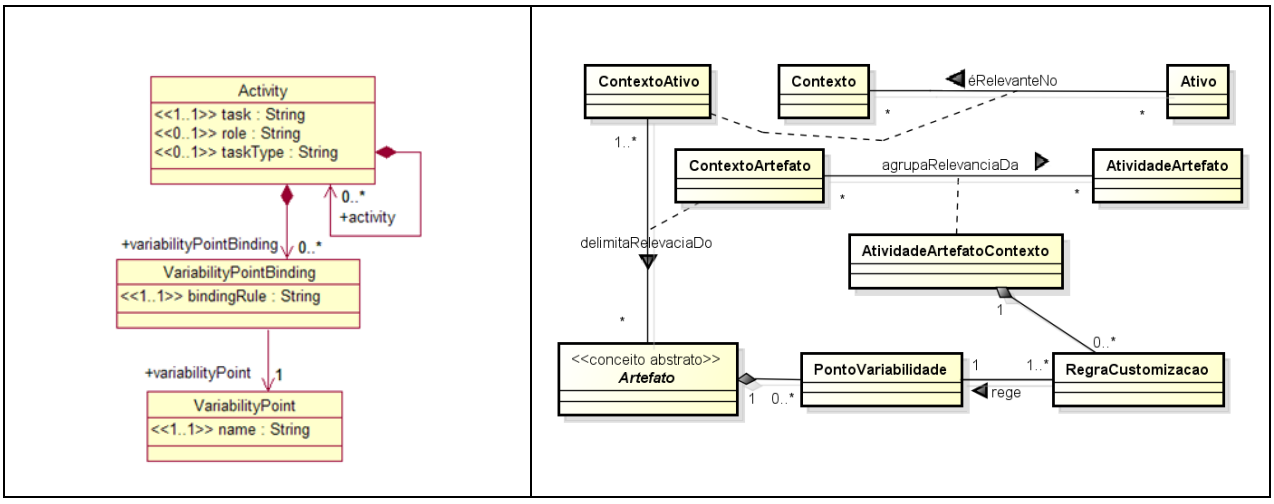
Conceito	Descrição e Justificativa	Origem
TempoCompilacao	É um tipo de DependênciaArtefato que determina que um Artefato depende em tempo de compilação, de outro Artefato.	
Iteração 4: Gestão de Ativos		
Versao	Cada uma das formas sucessivas que um artefato de software possui, à medida que é desenvolvido ou evoluído.	Conceito abstraído a partir da descrição semântica do <MecanismoVersionamento> (Quadro 5.5)
FeedBack	Corresponde a um posicionamento ou avaliação dado para uma versão de um Ativo.	Conceito abstraído a partir da descrição semântica do <MecanismoFeedback> (Quadro 5.5)
Mecanismo	É um conjunto de ações específicas para atingir um determinado objetivo do plano de gestão de ativos reutilizáveis.	Conceito identificado a partir da abstração e do agrupamento de diversas atividades relacionadas a Gestão de Ativos. Essas atividades possuem características em comum que são representadas pelo conceito <Mecanismo>.
MecanismoNotificacao	É um dos mecanismos que um repositório pode ter. Visa atender uma estratégia de notificação aos usuários sobre modificações ocorridas nos ativos, definida no plano de gestão.	Conceito extraído a partir da percepção de que um repositório deve oferecer um mecanismo para concretizar a EstrategiaNotificação do plano de gestão (Quadro 5.5).
MecanismoControleAcesso	É um mecanismo que um repositório pode ter que tem por objetivo, executar a política de controle de acesso definida no plano de gestão.	Conceito extraído a partir da percepção de que um repositório deve oferecer um mecanismo para concretizar a PolíticaControleAcesso do plano de gestão (Quadro 5.5).
EstrategiaPublicidade	É um documento que define as estratégias de publicidade, por exemplo, os ativos mais utilizados.	Faz parte do plano de gestão e foi definida para ser atendida pelo MecanismoPublicidade, identificado no Quadro 5.5.
EstrategiaMetrica	É um documento que define as métricas que devem ser coletadas, registradas e contabilizadas sobre os ativos.	Faz parte do plano de gestão e foi definida para ser atendida pelo MecanismoMétrica, identificado no Quadro 5.5.

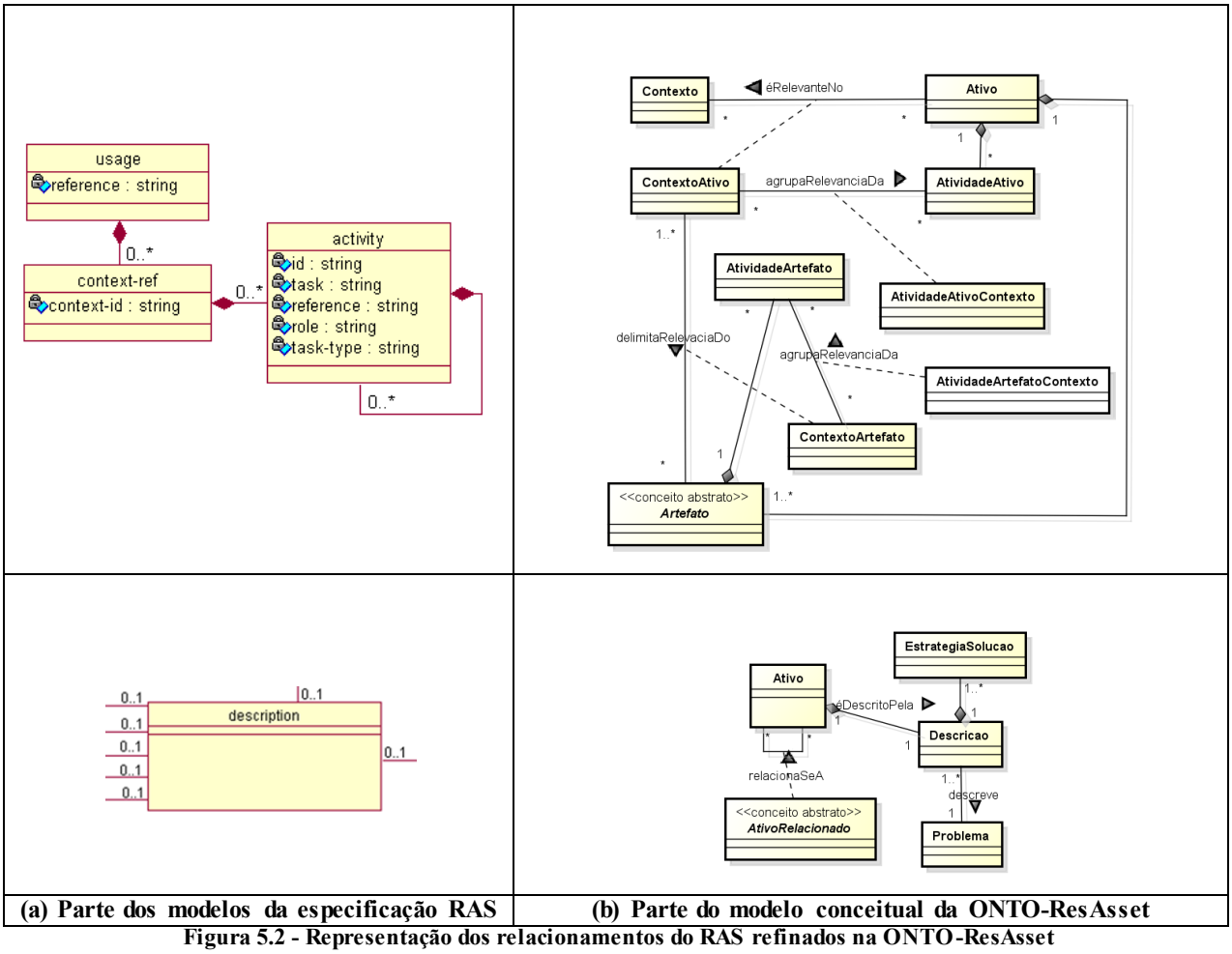
No Quadro 5.7 são apresentados os relacionamentos entre alguns elementos do RAS que foram refinados durante a concepção da ONTO-ResAsset, com a respectiva justificativa. Esses refinamentos foram necessários pois, assim como o RAS, este trabalho utiliza a notação do diagrama de classes da UML e, durante a representação dos conceitos da especificação de ativos da ONTO-ResAsset, houve a necessidade de explicitar alguns relacionamentos descritos na documentação do RAS mas que não estavam explicitados nos modelos dessa especificação.

Quadro 5.7- Alterações de relacionamentos entre elementos do RAS

Representação no RAS (Figura 5.2(a))	Representação na ontologia (Figura 5.2(b))	Justificativa
Relacionamento entre atividade (*Activity*) e regra de customização (*VariabilityPointBinding*)	Relacionamento entre <AtividadeArtefatoContexto> e <RegraCustomização>	A regra de customização personaliza um ou mais pontos de variabilidades de um artefato. Para essa customização é necessário executar uma ou mais atividades do artefato, as quais estão relacionadas a um contexto.
Relacionamento entre contexto *<context-ref>* e atividade *<activity>*	Relacionamento entre <AtividadeAtivo> com <ContextoAtivo> e entre <AtividadeArtefato> com <ContextoArtefato>	Como a separação das atividades dos ativos e dos artefatos foi explicitada, deixando mais claras as Atividades do Artefato (AtividadeArtefato) e as Atividades do Ativo (AtividadeAtivo) e, havendo a associação entre as atividades com um contexto, houve a necessidade de modificar os relacionamentos originais do RAS.
Todos os relacionamentos com o elemento *<description>*.	Relacionamento de composição entre <Descrição> com <Problema> e <EstratégiaSolução> Relacionamento de composição entre <Ativo> e <Descrição>.	De acordo com o RAS, o elemento descrição tem o objetivo de prover informações semânticas sobre os outros elementos dessa especificação. Assim, possui associação com praticamente todos os elementos do 2º nível do RAS. No contexto da ontologia ONTO-ResAsset, <Descrição> é composta por um <Problema> e uma <EstratégiaSolução> ou mais e, pertence a um <Ativo>. Decidiu-se criar relacionamento de <Descrição> apenas com <Ativo> por ser o conceito mais importante para a especificação de um ativo.

Na Figura 5.2 (a) são apresentadas as partes dos modelos da especificação RAS que possuem os relacionamentos que foram refinados. Na Figura 5.2 (b) são ilustradas as partes do modelo conceitual da ONTO-ResAsset que possuem os relacionamentos após o refinamento.





(a) Parte dos modelos da especificação RAS | **(b) Parte do modelo conceitual da ONTO-ResAsset**

Figura 5.2 - Representação dos relacionamentos do RAS refinados na ONTO-ResAsset

5.4.2. Conceitualização e implementação das atividades fundamentais de desenvolvimento de software

Nesta seção são apresentados os conceitos relacionados às atividades fundamentais de desenvolvimento de software, as quais são representadas pelo conceito <*AtividadeConstrucao*>, bem como a especificação dos relacionamentos entre eles e a multiplicidade de cada relacionamento.

O conceito <*AtividadeConstrucao*> possui origem na ontologia de atividade do trabalho de Falbo (1998). De acordo com esse autor, uma <Pessoa> *desempenha* um <Papel> ou mais e um <Papel> *realiza* uma <*AtividadeConstrucao*> ou mais em um processo de desenvolvimento de software, conforme ilustrado na Figura 5.3.

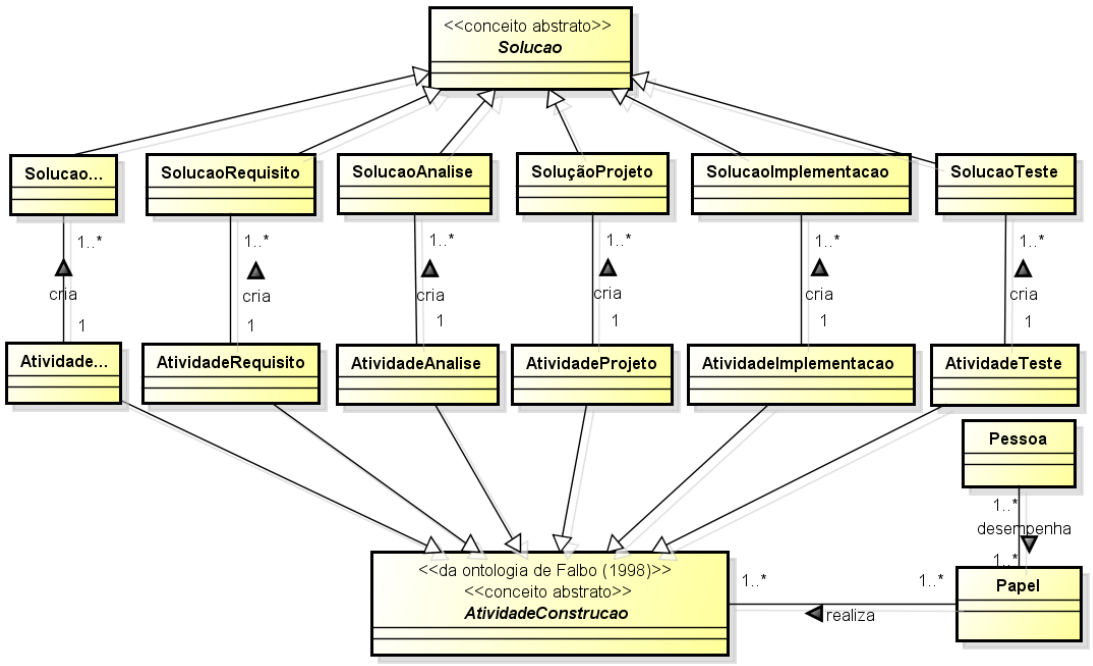


Figura 5.3 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <AtividadeConstrucao>

A <AtividadeRequisito>, a <AtividadeAnalise>, a <AtividadeProjeto>, a <AtividadeImplementacao> e a <AtividadeTeste> *éUmTipoDe* <*AtividadeConstrucao*>, que são as atividades fundamentais de desenvolvimento de software consideradas neste trabalho. A <Atividade...> indica que pode existir outro tipo de <*AtividadeConstrucao*>, além das explicitadas no modelo.

Cada <*AtividadeConstrucao*>, quando realizada, *cria* pelo menos uma <*Solucao*> correspondente, podendo ser <SolucaoRequisito>, <SolucaoAnalise>, <SolucaoProjeto>, <SolucaoImplementacao>, <SolucaoTeste> e <Solucao...>. Essa última indica que podem existir outras soluções, não apenas as explicitadas no modelo.

O conceito <*Solucao*> é abstrato, ou seja, não devem haver instâncias desse conceito, mas sim apenas dos conceitos <SolucaoRequisito>, <SolucaoAnalise>, <SolucaoProjeto>, <SolucaoImplementacao>, <SolucaoTeste> ou <Solucao...>.

No Quadro 5.8 é detalhado cada relacionamento existente entre os conceitos relacionados às atividades fundamentais de desenvolvimento representadas pelo conceito <*AtividadeConstrucao*>.

Quadro 5.8 - Relacionamento entre conceitos com foco em <AtividadeConstrucao>

Multiplicidade Origem	Origem	Nome do Relacionamento	Multiplicidade Destino	Destino
-	<AtividadeRequisito> <AtividadeAnalise> <AtividadeProjeto> <AtividadeImplementacao> <AtividadoTeste> <Atividade...>	*éUmTipoDe*	-	<AtividadeConstrucao>
1	<AtividadeRequisito>	*cria*	1..*	<SolucaoRequisito>
1	<AtividadeAnalise>	*cria*	1..*	<SolucaoAnalise>
1	<AtividadeProjeto>	*cria*	1..*	<SolucaoProjeto>
1	<AtividadeImplementacao>	*cria*	1..*	<SolucaoImplementacao>
1	<AtividadeTeste>	*cria*	1..*	<SolucaoTeste>
1	<Pessoa>	*desempenha*	1..*	<Papel>
1..*	<Papel>	*realiza*	1..*	<AtividadeConstrucao>

5.4.3. Conceitualização e implementação dos ativos reutilizáveis

Nesta seção são apresentados os conceitos relacionados à solução, artefato e ativos reutilizáveis, os quais são representados respectivamente pelos conceitos <*Solucao*>, <*Artefato*> e <Ativo> bem como a especificação dos relacionamentos entre eles e a multiplicidade de cada relacionamento. Para facilitar a escrita e o entendimento desta seção, as ilustrações e os quadros de relacionamentos estão apresentados separadamente, de acordo com cada conceito apresentado.

Uma <*Solucao*> *concretiza* uma <EstrategiaSolucao> para resolver o <Problema> de um <Ativo>, conforme ilustrado na Figura 5.4. Essa <*Solucao*> pode ser *fornecida* quando houver uso do <Ativo> em um dado <Contexto> chamado de <ContextoAtivo>.

A <*Solucao*> foi mapeada como um conceito abstrato, portanto deve existir de acordo com o tipo de atividade que a cria (conforme ilustrado na Figura 5.3), podendo ser <SolucaoRequisito>, <SolucaoAnalise>, <SolucaoProjeto>, <SolucaoImplementacao> e <SolucaoTeste>. Existe o tipo <Solucao...> que indica que podem existir ou surgir outros tipos de <*Solucao*>, além daquelas definidas no modelo.

Cada <*Solucao*> é realizada por ao menos um <*Artefato*> em um determinado contexto de utilização (<ContextoArtefato>) que *delimitaRelevanciaDo* <*Artefato*> que pertence a um ativo desse contexto (<ContextoAtivo>). Essa realização é representada por meio do relacionamento *éRealizadoPelo* do <ArtefatoSolucao>, oriundo do relacionamento existente entre <*Artefato*> e <*Solucao*>.

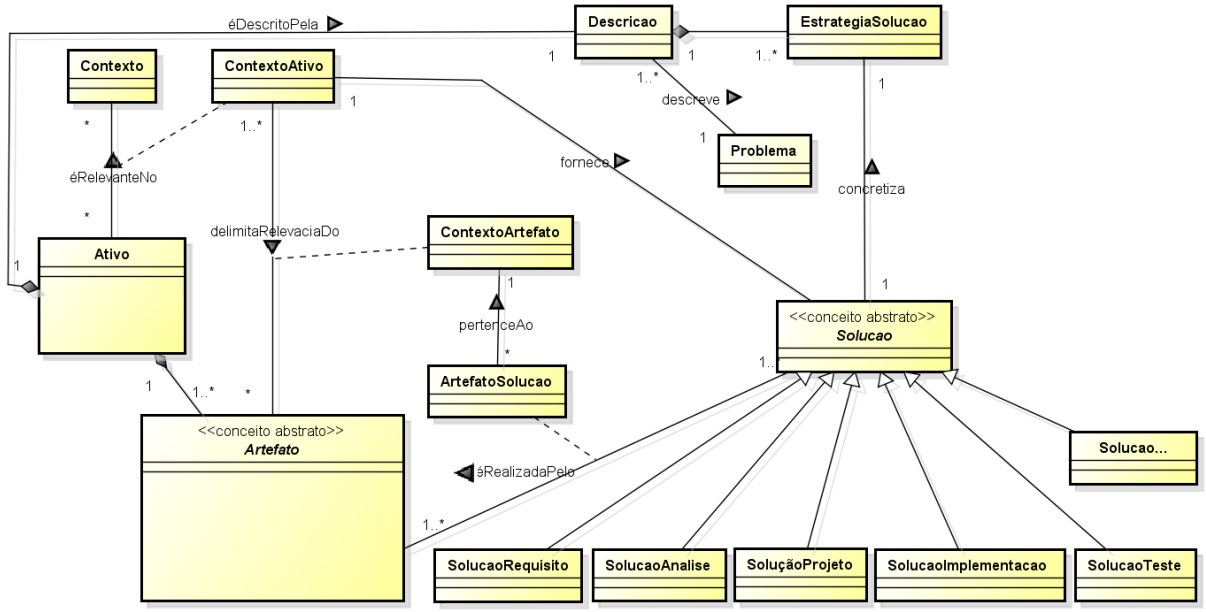


Figura 5.4 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <Solucao>

Por meio do conceito <ArtefatoSolucao> é possível determinar o tipo do <*Artefato*> participante da solução já que <Solucao> *éRealizadaPelo* <*Artefato*> gerando o <ArtefatoSolucao>, por exemplo, dado uma instância *c* de <Codigo-fonte>, que também é um <*Artefato*> e por conseguinte possui uma relação <ArtefatoSolucao> *éRealizadoPelo*(*c, s),* pode-se dizer que *c* é um artefato cuja solução foi criada na <AtividadeImplementação> se *s* for uma <SolucaoImplementacao>. Isso ocorre de maneira análoga para cada grupo, como detalhado no Quadro 5.10, que define cada relacionamento existente entre os conceitos relacionados à <*Solucao*> e os tipos de <*Artefato*>.

Quadro 5.9 - Relacionamento entre os conceitos com foco em <Solucao>

Multiplicidade Origem	Origem	Nome do Relacionamento	Multiplicidade Destino	Destino
1	<Solucao>	*concretiza*	1	<EstrategiaSolucao>
-	<SolucaoRequisito> <SolucaoAnalise> <SolucaoProjeto> <SolucaoImplementacao> <SolucaoTeste> <Solucao..>	*éUmTipoDe*	-	<Solucao>
1..*	<*Artefato*>	*éRealizadaPelo*	1..*	<Solucao>
*	<ArtefatoSolucao>	*pertenceAo*	1	<ContextoArtefato>
1	<ContextoAtivo>	*fornece*	1	<Solucao>

O <*Artefato*> é um recurso de software que é criado para prover uma <*Solucao*>, conforme ilustrado na Figura 5.5. Um <*Artefato*> é um conceito abstrato e *tem* <RegraUso>

que define e restringe sua utilização, assim como <RegraUso> também *coordenaExecução* de uma <*Atividade*> ou mais que foram definidas para a sua utilização.

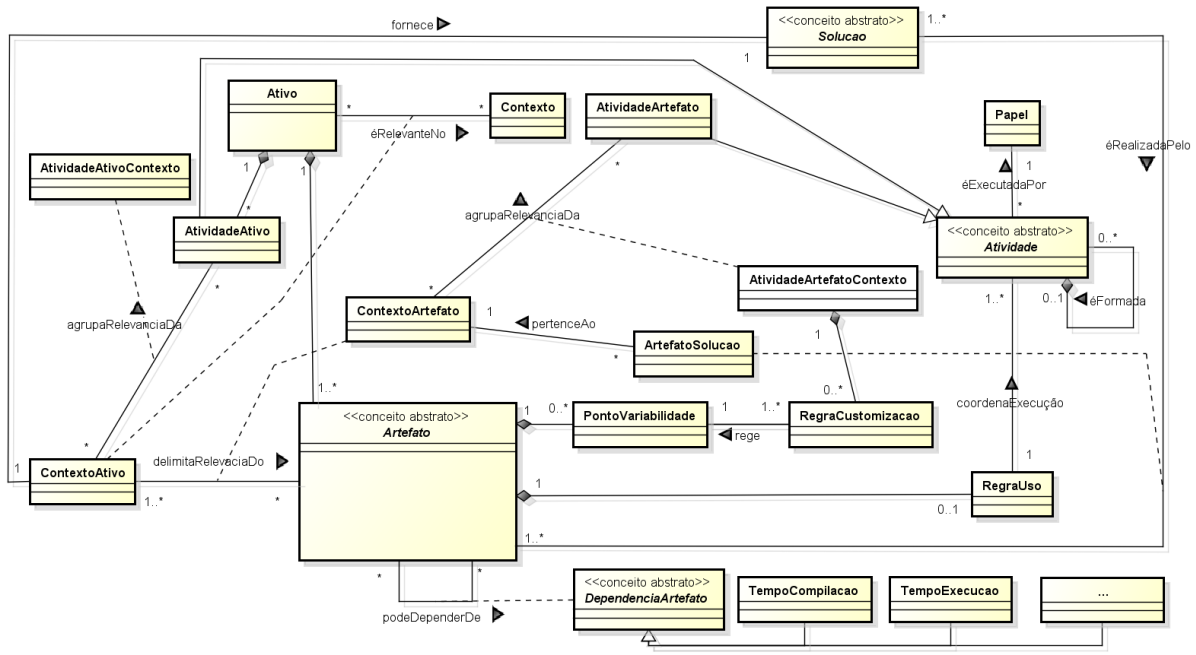


Figura 5.5 - Parte do modelo conceitual da ONTO-ResAsset com foco no conceito <Artefato>

Cada tipo de <*Artefato*>, mostrado na Figura 5.6, que foi identificado na fase de especificação na Seção 5.3.2, e cada tipo de <*Solucao*>, que foi identificado na fase de especificação na Seção 5.4.1, e que podem participar de um <ArtefatoSolucao>, ou seja, uma <Solucao> *éRealizadaPelo* <*Artefato*>. A ideia desse relacionamento é capturada pelo conceito <ArtefatoSolucao> como mostrado no Quadro 5.10. Por exemplo, <Código-fonte> é um <*Artefato*> que pode participar de um <ArtefatoSolucao> apenas se a solução for uma <SolucaoImplementacao>.

Quadro 5.10 – Conceitos que representam os tipos de <Artefato>

<Artefato> de <ArtefatoSolucao>	**<Solucao> de <ArtefatoSolucao>**
<Modelo de Processo de Negócio>	<SolucaoRequisito>
<Modelo Requisito>	<SolucaoRequisito>
<Documento de Requisito>	<SolucaoRequisito>
<Modelo Análise>	<SolucaoAnalise>
<Modelo Componente>	<SolucaoProjeto>
<Modelo de Arquitetura>	<SolucaoProjeto>
<Modelo de Interface Humana>	<SolucaoProjeto>
<Modelo de Projeto>	<SolucaoProjeto>
<Algoritmo>	<SolucaoProjeto>
<Código-fonte>	<SolucaoImplementacao>
<Componente>	<SolucaoImplementacao>
<Módulo>	<SolucaoImplementacao>
<Caso de Teste>	<SolucaoTeste>

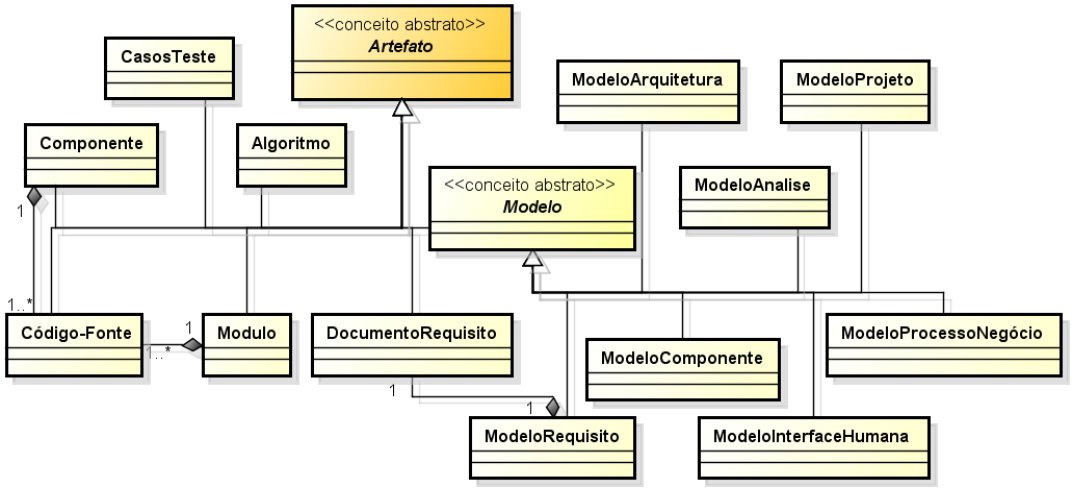


Figura 5.6 - Tipos de <Artefato>

Uma <*Atividade*> é uma tarefa ou ação que deve ou pode ser realizada quando o <Ativo> e o <*Artefato*> são utilizados, representadas respectivamente pelos conceitos <AtividadeAtivo> e <AtividadeArtefato> de acordo com a Figura 5.5. Cada <*Atividade*> pode ser desempenhada por pessoas ou por ferramentas computacionais. Uma <*Atividade*> é considerada um conceito abstrato no sentido de que não deve existir uma <*Atividade*> que não seja <AtividadeArtefato> ou <AtividadeAtivo>. Uma <*Atividade*> *éFormada* por outras, e qualquer <*Atividade*> *éExecutadaPor* um <Papel>.

Um <*Artefato*> é a unidade principal que compõe um <Ativo>, sendo um <Ativo> composto por um <*Artefato*> ou mais. Um <*Artefato*> *éCompostoDe* pelo menos um <PontoVariabilidade>, que indica as partes do artefato que podem ser personalizadas. Isso é definido por meio da <RegraCustomizacao> que *rege* o <PontoVariabilidade> de acordo com uma determinada <AtividadeArtefatoContexto> que *éCompostoDe* <RegraCustomizacao>.

Um <*Artefato*> *podeDependerDe* outros, essa <*DependenciaArtefato*> é um conceito abstrato e <TempoCompilacao>, <TempoExecucao> e <...> *éUmTipoDe* <*DependenciaArtefato*> que podem existir. O conceito <...> é utilizado para representar que podem existir outros tipos de dependência entre artefatos, além dos explicitados no modelo.

Dadas as instâncias a_1 *e* a_2 de <*Artefato*>, não pode haver <*DependenciaArtefato*> (a_1, a_1), ou seja, não pode haver dependência da instância a_1 com ela mesma. Assim, faz-se necessário tornar explícito que a <DependenciaArtefato>(a_1, a_2) implica que $a_1 \neq a_2$.

O <*Artefato*> pertence ao <Contexto> de utilização do <Ativo> do qual é composto, representado por <ContextoAtivo> que *delimitaRelevanciaDo* <*Artefato*> dado um <ContextoAtivo> ou mais. Consequentemente, um <*Artefato*> também deve ser utilizado

apenas em contextos do ativo o qual faz parte, conforme indicado pelo conceito <ContextoArtefato> na Figura 5.5.

Cada <AtividadeArtefato> é realizada de acordo com o <ContextoArtefato> e é indicada pelo conceito <AtividadeArtefatoContexto> ou seja, <AtividadeArtefatoContexto> *agrupaRelevanciaDa* <AtividadeArtefato> dado <ContextoArtefato>. Os relacionamentos envolvidos com os conceitos <ContextoArtefato> e <AtividadeContextoArtefato> possuem duas restrições.

A primeira restrição estabelece que não há contexto de uso do artefato (<ContextoArtefato>) que não seja um contexto também para o seu ativo (<ContextoAtivo>), sendo assim <ContextoArtefato> *delimitaRelevanciaDo* <*Artefato*> em relação ao <ContextoAtivo>.

A segunda restrição indica, de forma transitiva, que não há atividades definidas para o artefato (<AtividadeArtefato>) fora do <ContextoArtefato>.

No Quadro 5.11 é detalhado cada relacionamento existente entre os conceitos relacionados ao <*Artefato*>.

Quadro 5.11 - Relacionamento entre os conceitos com foco no <Artefato>

Multi-plicidade Origem	Origem	Nome do Relacionamento	Multi-plicidade Destino	Destino
-	<AtividadeArtefato>	*éUmTipoDe*	-	<Atividade>
1	<Artefato>	*éCompostoDe*	0..1	<RegraUso>
1	<Artefato>	*éCompostoDe*	0..*	<PontoVariabilidade>
1	<AtividadeArtefatoContexto>	*éCompostoDe*	0..*	<RegraCustomizacao>
1	<Artefato>	*éCompostoDe*	*	<AtividadeArtefato>
1	<RegraUso>	*coordenaExecução*	1..*	<Atividade>
*	<Atividade>	*éExecutadaPor*	1	<Papel>
0..1	<Atividade>	*éFormada*	0..*	<Atividade>
1	<RegraUso>	*coordenaExecução*	1..*	<Atividade>
1..*	<RegraCustomizacao>	*rege*	1	<PontoVariabilidade>
*	<Artefato>	*podeDependerDe*	*	<Artefato>
1..*	<ContextoAtivo>	*delimitaRelevanciaDo*	*	<Artefato>
*	<ContextoArtefato>	*agrupaRelevanciaDa*	*	<AtividadeArtefato>
-	<TempoCompilacao> <TempoExecucao> <...>	*éUmTipoDe*	-	<DependenciaArtefato>

Um <Ativo> *éDescritoPela* <Descricao>, que é a informação que *descreve* um <Problema> e *éCompostoDe* uma <EstrategiaSolucao> ou mais. O <Ativo> *éCompostoDe* outros elementos, sendo o principal deles o <*Artefato*>. Dessa forma, o <Ativo> é composto por pelo menos um <*Artefato*>. O <Ativo> *relacionaSeA* um <Ativo> ou mais, indicado na Figura 5.7 pelo conceito abstrato <*AtivoRelacionado*>.

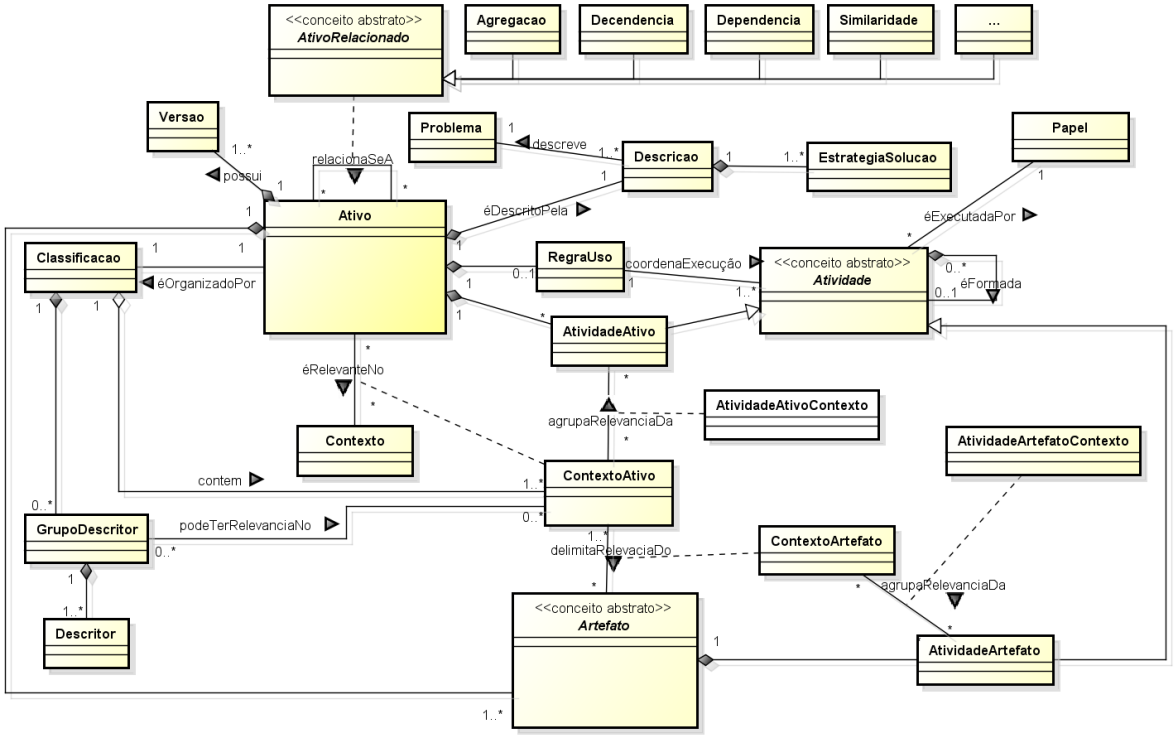


Figura 5.7 - Parte do modelo conceitual com foco no conceito <Ativo>

O tipo do relacionamento entre ativos pode ser determinado como sendo <Agregacao>, <Similaridade>, <Composicao>, <Decendencia> que *éUmTipoDe* <*AtivoRelacionado*>. O tipo de relacionamento <...> indica que podem existir outros tipos de relacionamentos entre <Ativo>, além dos determinados no modelo.

O <Ativo> também *éCompostoDe* <RegraUso>, <Versao> e <AtividadeAtivo>. Um <Ativo> *possui* obrigatoriamente pelo menos uma <Versao>. A <RegraUso>, assim como para o <*Artefato*>, define e restringe a utilização do <Ativo> e *coordenaExecucao* de cada <*Atividade*> e, por transitividade, *coordenaExecucao* de cada <AtividadeAtivo>.

Uma <AtividadeAtivo> de um <Ativo> é realizada de acordo e limitada a um <ContextoAtivo>. Essa limitação é expressa com o relacionamento *agrupaRelevanciaDa* existente entre <ContextoAtivo> e <AtividadeAtivo>, expressa por meio do conceito <AtividadeAtivoContexto>. Assim, define-se a regra de que não existe <AtividadeAtivo> sem que haja um <ContextoAtivo> para o qual a <*Atividade*> possa ser executada.

A <Classificacao> é o conceito que determina uma categorização para um <Ativo> ou seja, um <Ativo> *éOrganizadoPor* uma <Classificacao> que *éCompostoDe* de <GrupoDescritor> e sempre *contém* um <ContextoAtivo> ou mais.

Uma <Classificacao> *contém* um <ContextoAtivo> ou mais, por exemplo, dada uma instância *c* de <ContextoAtivo> que refere-se a um contexto de "implementação em J2EE na versão 1.3" de um <Ativo> *a*, obrigatoriamente a <Classificacao> *ac* desse ativo *a* deverá conter *c*, e assim para todo <ContextoAtivo> no qual o <Ativo> for relevante.

A <Classificacao> *éCompostoDe* <GrupoDescritor>, e esse por sua vez *éCompostoDe* um <Descritor> ou mais. Essa hierarquia de composição tem a função de prover maior detalhamento da <Classificação>, podendo prover informações semânticas adicionais ao ativo, principalmente para a realização do <MecanismoBusca> (Seção 5.4.4).

Para auxiliar esse detalhamento, o <GrupoDescritor> *podeTerRelevanciaNo* <ContextoAtivo> do ativo. Isso está representado na ontologia como forma de complementar as informações sobre um ativo em um contexto de uso, possibilitando uma completa e detalhada classificação para os ativos reutilizáveis.

No Quadro 5.12 é detalhado cada relacionamento existente entre os conceitos relacionados ao <Ativo>.

Quadro 5.12 - Relacionamento entre os conceitos com foco no <Ativo>

Multi-plicidade Origem	**Origem**	**Nome do Relacionamento**	**Multi-plicidade Destino**	**Destino**
1	<Ativo>	*éDescritoPela*	1	<Descricao>
1..*	<Descricao>	*Descreve*	1	<Problema>
1	<Descricao>	*éCompostoDe*	1..*	<EstrategiaSolucao>
1	<Ativo>	*Possui*	1..*	<Versao>
1	<Ativo>	*éCompostoDe*	0..1	<RegraUso>
1	<RegraUso>	*coordenaExecução*	1..*	<Atividade>
-	<AtividadeAtivo>	*éUmTipoDe*	-	<Atividade>
1	<Ativo>	*éCompostoDe*	*	<AtividadeAtivo>
*	<Ativo>	*éRelevanteNo*	*	<Contexto>
*	<ContextoAtivo>	*agrupaRelevanciaDa*		<AtividadeAtivo>
1	<Ativo>	*éOrganizadoPor*	1	<Classificacao>
1	<Classificacao>	*éCompostoDe*	0..*	<GrupoDescritor>
1	<Classificacao>	*Contém*	1..*	<ContextoAtivo>
1	<GrupoDescritor>	*éCompostoDe*	1..*	<Descritor>
0..*	<GrupoDescritor>	*podeTerRelevanciaNo*	0..*	<ContextoAtivo>
*	<Ativo>	*relacionaSeA*	*	<Ativo>
-	<Agregacao> <Similaridade> <Composicao> <Decendencia> <...>	*éUmTipoDe*	-	<AtivoRelacionado>

5.4.4. Conceitualização e implementação da gestão de ativos

Nesta seção são apresentados os conceitos relacionados à Gestão de Ativos, independentemente de apoio computacional envolvido. Apesar da concepção dos conceitos ser fortemente baseada no processo de gestão de reuso de ativos da norma ABNT NBR ISO/IEC 12207 (2009), o conceito <Repositorio> foi adicionado, representando o mecanismo de armazenamento e recuperação apresentado na norma citada.

O <PlanoGestao> é um documento que define os elementos necessários para a gestão de <Ativo>, sendo que o <Papel> "Gerente de Ativos" o *cria*, conforme ilustrado na Figura 5.8.

Um <PlanoGestao> *éCompostoDe* <CriterioDesativacao>, <CriterioAceitacao>, <CriterioCertificacao>, <EstrategiaClassificacao>, <EstrategiaNotificacao>, <EstrategiaPublicidade>, <EstrategiaMetrica> e <PoliticaControleAcesso>; os quais são documentos que determinam como será efetuada a gestão de ativos.

A <EstrategiaClassificacao> é um documento que *define* as políticas e métodos que devem ser realizados quando da criação de uma <Classificacao> ou mais pela qual o <Ativo> *éOrganizadoPor*, ou seja, determinam como serão criadas e agrupadas as informações que descrevem um ativo.

Um <Repositorio> *atende* o <PlanoGestao> e *temAcessoAOutro* <Repositorio>. Isso é capturado pelo conceito <SuporteMultiplasFontes>. O <Repositorio> *éCompostoDe* um conjunto de <*Mecanismo*>, que contempla (*atende*) o <PlanoGestao>.

O relacionamento *atende* entre <Repositorio> e <PlanoGestao> indica implicitamente que há relacionamento entre um <*Mecanismo*> ou mais e os elementos que compõem o <PlanoGestao>. Por exemplo, <MecanismoMetrica> *atende* por transição a <EstrategiaMetrica> definida para o <PlanoGestao>. Portanto, o relacionamento *atende* é um relacionamento transitivo.

O <*Mecanismo*> é um conceito abstrato que representa um conjunto de procedimentos que um <Repositório> pode fornecer. Um <*Mecanismo*> é utilizado pelo usuário do <Repositorio>, de acordo com a <PoliticaControleAcesso> definida para cada <Papel>. Em outras palavras, a <PoliticaControleAcesso> *delimitaUsoDo* <Mecanismo> dado um <Papel>.

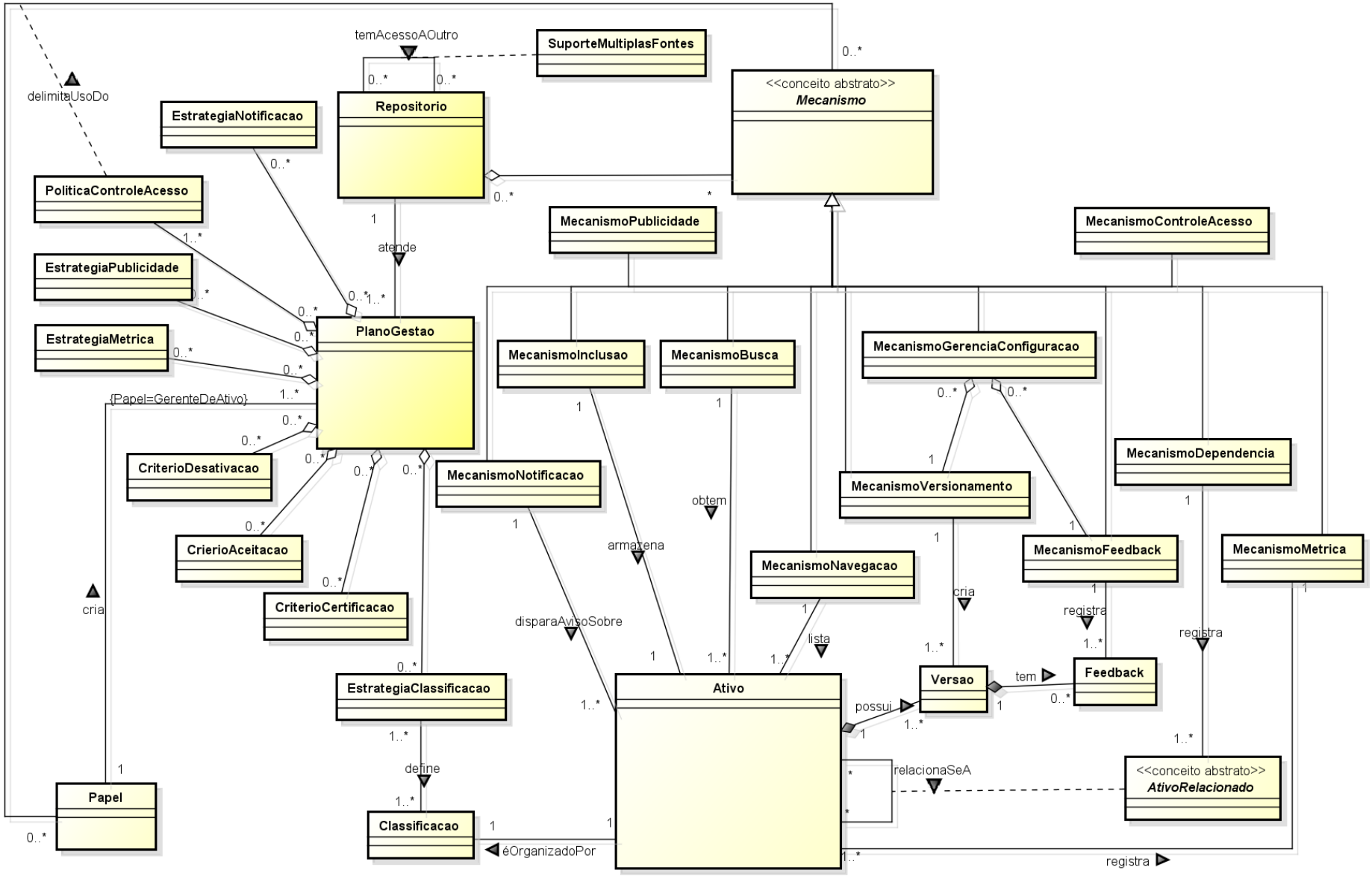


Figura 5.8 - Parte do modelo conceitual com foco no conceito <PlanoGestao>

Todos os tipos de mecanismos, como o <MecanismoNotificacao>, <MecanismoInclusao>, <MecanismoBusca>, e etc. *éUmTipoDe* <*Mecanismo*>. O <MecanismoGerenciaConfiguracao> *éCompostoDe* obrigatoriamente pelo menos um <MecanismoVersionamento> e um <MecanismoFeedback>; sendo o primeiro responsável pelo controle de versões e o segundo pelo controle de mudanças nos ativos.

Alguns mecanismos relacionam-se diretamente com <Ativo>: o <MecanismoInclusao> *armazena* o <Ativo>, o <MecanismoBusca> *obtem* um <Ativo> ou mais, o <MecanismoNavegacao> *lista* um <Ativo> ou mais, o <MecanismoNotificacao> *disparaAvisoSobre* um <Ativo> ou mais e o <MecanismoMetrica> *registra* as métricas em relação ao <Ativo>.

O <MecanismoVersionamento>, o <MecanismoFeedback> e o <MecanismoDependencia>, por sua vez, possuem relacionamentos com outros elementos do <Ativo>. O <MecanismoVersionamento> *cria* e registra uma <Versao> ou mais de um mesmo ativo. Cada <Versao> *éCompostoDe* algum <Feedback> que um consumidor do <Ativo> *registra* por meio do <MecanismoFeedback>.

Por último, o <MecanismoDependencia> *registra* <*AtivoRelacionado*>, que é a interdependência que pode existir entre dois ou mais <Ativo>, a qual pode ser de vários tipos conforme descrito no glossário de termos do Quadro 5.4

No Quadro 5.13 é detalhado cada relacionamento existente entre os conceitos relacionados a <PlanoGestao>.

Quadro 5.13 - Relacionamento entre os conceitos com foco no <PlanoGestao>

Multiplicidade Origem	**Origem**	**Nome do Relacionamento**	**Multiplicidade Destino**	**Destino**
1	<Papel>	*Cria*	1..*	<PlanoGestao>
0..*	<PlanoGestao>	*éCompostoDe*	0..*	<CriterioDesativacao> <CriterioAceitacao> <CriterioCertificacao> <EstrategiaClassificacao> <EstrategiaNotificacao> <EstrategiaMetrica> <EstrategiaPublicidade>
0..*	<PlanoGestao>	*éCompostoDe*	1..*	<PoliticaControleAcesso>
1	<Repositorio>	*Atende*	1..*	<PlanoGestao>
0..*	<Repositorio>	*éCompostoDe*	*	<Mecanismo>
0..*	<Repositorio>	*temAcessoAOutro*	0..*	<Repositorio>
0..*	<PoliticaControle Acesso>	*delimitaUsoDo*	0..*	<Mecanismo>

Multiplicidade Origem	Origem	Nome do Relacionamento	Multiplicidade Destino	Destino
-	<MecanismoInlusao> <MecanismoBusca> <MecanismoNavegacao> <MecanismoNotificacao> <MecanismoMetrica> <MecanismoPublicidade> <MecanismoVersionamento> <MecanismoDependencia> <MecansimoFeedback> <MecanismoControleAcesso> <MecansimoGerenciaConfiguracao>	*éUmTipoDe*	-	<Mecanismo>
0..*	<MecansimoGerenciaConfiguracao>	*éCompostoDe*	1..*	<MecanismoVersionamento> <MecansimoFeedback>
0..*	<Repositorio>	*temAcessoAOutro*	0..*	<Repositorio>
1	<MecanismoNotificacao>	*disparaAvisoSobre*	1..*	<Ativo>
1	<MecanismoInlusao>	*Armazena*	1	<Ativo>
1	<MecanismoBusca>	*Obtem*	1..*	<Ativo>
1	<MecanismoNavegacao>	*Lista*	1..*	<Ativo>
1	<EstrategiaClassificacao>	*define*	1..*	<Classificacao>
1	<MecanismoVersionamento>	*cria*	1..*	<Versao>
1	<MecanismoFeedback>	*registra*	1..*	<Feedback>
1	<MecanismoDependencia>	*registra*	1..*	<AtivoRelacionado>
1	<MecanismoMetrica>	*registra*	1..*	<Ativo>

5.5. Considerações finais

Neste capítulo foi apresentada a construção da ontologia proposta, tomando como base passos da metodologia *Methontology* e utilizando glossários de termos para a identificação e definição dos conceitos, de acordo com o escopo definido.

Foram utilizados também o diagrama de classes da UML, para a representação gráfica da ONTO-ResAsset, e linguagem natural semiestruturada, para a descrição dos relacionamentos e restrições, bem como tabelas de relacionamentos para documentar os relacionamentos entre os conceitos e suas respectivas multiplicidades.

No próximo capítulo é apresentada a avaliação da ontologia ONTO-ResAsset, contemplando o passo avaliar da *Methontology,* sob a perspectiva de especialistas e não especialistas no domínio de reuso de software.

Capítulo 6. Avaliação da ONTO-ResAsset

6.1. Considerações iniciais

Neste capítulo é apresentada a avaliação da ONTO-ResAsset, realizada em duas etapas sob diferentes perspectivas. Na primeira etapa, a ontologia foi avaliada por especialistas no domínio de reuso de software e, na segunda etapa, por dois grupos, representando os não especialistas no domínio, interessados em potencial, na reutilização de ativos de software. O grupo 1 dos não especialistas é representado por voluntários de uma equipe de desenvolvimento de software de uma empresa real e o grupo 2 por alunos da disciplina Desenvolvimento de Software do Programa de Pós-Graduação da Facom, ofertada no segundo semestre de 2013. Em ambas etapas, a avaliação da ONTO-ResAsset foi conduzida por meio de questionários.

Na Seção 6.2 é apresentado o planejamento da avaliação. Na Seção 6.3 e na Seção 6.4 são apresentados a execução e os resultados da primeira e da segunda etapas da avaliação, respectivamente, e na Seção 6.5 são apresentadas as considerações finais do capítulo.

6.2. Planejamento da avaliação

O **objeto da avaliação** é a ontologia ONTO-ResAsset e o **objetivo** é avaliar essa ontologia por especialistas e não especialistas em reutilização de software. A avaliação tem como **foco qualitativo** avaliar a capacidade da ONTO-ResAsset em prover conhecimento sobre o domínio de especificação e gestão de ativos reutilizáveis e tem como **perspectiva** apoiar a disseminação desse domínio. Como **contexto**, a avaliação, realizada em agosto de 2013, foi conduzida por um aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação da Facom, tendo a colaboração de um docente como consultor.

A avaliação foi dividida em duas etapas. Na primeira etapa, a avaliação foi feita por especialistas no domínio de reutilização de software. A partir das respostas dos questionários dessa etapa, foi realizada uma análise para identificar as oportunidades de melhorias na ONTO-ResAsset e, consequentemente, refiná-la. Posteriormente, a segunda etapa da avaliação foi feita por dois grupos de não especialistas no domínio, com perfis distintos.

A **seleção dos participantes** da primeira etapa foi realizada levando em consideração pesquisadores da área de Engenharia de Software que têm interesse em linhas de pesquisa em reutilização de software. Os participantes da segunda etapa da avaliação foram selecionados por conveniência, tomando como base a equipe de desenvolvimento de uma empresa real, a

qual o autor deste trabalho pertence ao seu quadro funcional; e os alunos de pós-graduação matriculados na disciplina de Desenvolvimento de Software, ofertada no segundo semestre de 2013 e ministrada pela orientadora deste trabalho.

Antes do início da segunda etapa, foi ministrado um **treinamento** para o grupo 1 de desenvolvedores de software, visando facilitar o entendimento pelos participantes dos assuntos envolvidos durante a avaliação. Para isso, foi dado um treinamento de duas horas sobre reuso de software, ontologias, repositórios de reuso e sobre a notação do diagrama de classes da UML, que é a notação utilizada para representar o modelo conceitual da ontologia.

No Quadro 6.1 são apresentados os documentos fornecidos aos participantes em cada etapa da avaliação.

Quadro 6.1 - Lista de documentos utilizados durante a avaliação (etapas 1 e 2)

Documentos utilizados durante a avaliação	**Etapas da avaliação**
Glossário de conceitos da ontologia	Etapa 1
Modelo conceitual da ontologia	Etapa 1
Descrição do desenvolvimento da ontologia (capítulo 5)	Etapa 1
Questionário para avaliação da ONTO-ResAsset por especialista no domínio	Etapa 1
Glossário de conceitos da ontologia refinado após primeira etapa	Etapa 2
Modelo conceitual da ontologia refinado após primeira etapa	Etapa 2
Questionário de perfil dos não especialistas no domínio	Etapa 2
Questionário para avaliação da ONTO-ResAsset por não especialistas no domínio	Etapa 2

6.3. Primeira etapa da avaliação: Especialistas

Nesta etapa, a ONTO-ResAsset foi avaliada por dois especialistas na área de reuso de software por meio do questionário disponível no Apêndice D. O principal objetivo desse questionário é obter um parecer de especialistas no domínio quanto a coerência e correta cobertura da ontologia, de acordo com o seu escopo, e quanto à incoerências ou possíveis erros em relação à definição e representação dos conceitos e dos relacionamentos entre eles.

A partir dos questionários respondidos pelos avaliadores, foi realizada uma análise nas respostas para averiguar as sugestões de melhorias para a ONTO-ResAsset. A maioria das sugestões de melhoria indicada pelos especialistas foi aceita e está descrita na Seção 6.3.1. As sugestões de melhorias não atendidas são apresentadas na Seção 6.3.2, com a devida justificativa.

6.3.1. Sugestões dos especialistas atendidas

A seguir é apresentada cada sugestão de melhoria, indicada pelos avaliadores especialistas, e o respectivo refinamento realizado na ONTO-ResAsset:

- Melhorar descrição textual dos conceitos AtividadeAtivo, AtividadeArtefato e AtividadeArtefatoContexto no glossário de conceitos: A descrição textual desses conceitos foi revisada e aprimorada, adicionando exemplos para torná-la mais clara. Além disso, o trecho da descrição do conceito Ativo "e qual solução ele resolve" foi refinado para "e qual problema ele resolve", conforme sugestão do avaliador.
- Adicionar conceitos para representar os tipos do conceito DependenciaArtefato: foram criados conceitos que representam os tipos de dependência entre ativos. O conceito existente DependenciaArtefato tornou-se um conceito abstrato e foram criados os conceitos TempoExecução e TempoCompilação, os quais herdam de DependenciaArtefato. De forma análoga, o conceito AtivoRelacionado também sofreu o mesmo tipo de refinamento, se tornando um conceito abstrato. A partir da descrição semântica desse conceito, foram criados os conceitos Agregação, Similaridade, Dependência e Descendência, os quais herdam de AtivoRelacionado.
- Permitir maior flexibilização nas especializações de alguns conceitos: para evoluir a ONTO-ResAsset, novos conceitos e relacionamentos podem ser adicionados a ela. Por exemplo, outras atividades que apoiam o desenvolvimento de software, como é o caso das atividades de apoio podem ser adicionadas na ontologia, aumentando consequentemente o seu escopo. Com isso, outros tipos de artefatos também devem ser adicionados. O conceito "..." foi adicionado apenas no modelo conceitual, herdando de AtividadeContrucao, Solução, AtivoRelacionado, Artefato e DependenciaArtefato para explicitar essa flexibilidade da ONTO-ResAsset.
- Retirar a representação do conceito *RecursoSoftware*: optou-se por retirar esse conceito pois o número de associações que podem surgir com os coutros conceitos fogem do escopo definido para a ONTO-ResAsset.
- Retirar o relacionamento de composição entre os conceitos Ativo e Classificacao: a Classificacao deixou de formar uma composição com Ativo e se tornou participante de uma associação simples com Ativo, denominada

"*éOrganizadoPor*". A multiplicidade também foi alterada de 0..1 para 1, denotando que um Ativo é sempre organizado por uma Classificação.

- Retirar composição entre Descricao e Problema e ajustar sua multiplicidade: O conceito Descricao não compõe mais o conceito Problema, se tornando participante de uma associação simples, chamado de *"descreve";* tendo a multiplicidade alterada de 1 para 1..*, permitindo representar que um problema pode ser descrito por várias descrições.
- Retirar agregação entre o conceito Repositório com ele mesmo, representado na classe de associação SuporteMultiplasFontes: O auto-relacionamento de agregação entre o conceito Repositório, tornou-se uma associação simples chamada de *"temAcessoAOutro"*. Esse refinamento ocorreu pois o relacionamento de agregação permitia o entendimento de que um Repositório era composto por outros, não sendo esse o objetivo da representação e sim a ideia de que um Repositório pode ter acesso a outros, ou seja, usar como fonte de dados outros repositórios.
- Corrigir multiplicidade da associação entre Papel e Pessoa: a multiplicidade da associação entre Papel e Pessoa foi alterada de 1..* - 1 para 1..* - 1..*, permitindo a ideia de que um papel é desempenhado por uma ou várias pessoas e uma pessoa desempenha um ou mais papéis.
- Melhorar o nome dos relacionamentos das classes associativas: todos os nomes de relacionamento das classes associativas foram alterados de forma a representar de maneira mais clara a associação propriamente dita entre os dois conceitos. Antes da avaliação pelos especialistas, os nomes de tais relacionamentos, por restrição da ferramenta *Astah* utilizada na modelagem, tinham o mesmo nome de sua classe associativa. Essa restrição foi contornada retirando no modelo a visibilidade dos nomes de relacionamentos que eram iguais às classes associativas e usando o elemento de caixa de texto da ferramenta para nomear tais relacionamentos

6.3.2. Sugestões dos especialistas não atendidas

A seguir é apresentada cada sugestão de melhoria dos avaliadores especialistas, porém não atendida, com a devida justificativa.

- Retirar associação circular entre Contexto, Classificacao e GrupoDescritor: a associação circular entre Contexto, Classificação e GrupoDescritor não foi removida, pois é explicitada no RAS e mantida na ONTO-ResAsset, visando permitir tal redundância de informações a fim de facilitar a busca do ativo.
- Retirar herança entre MecanismoFeedback e Mecanismo: a herança entre MecanismoFeedback e Mecanismo foi mantida pois, apesar da associação de agregação com MecanismoGerenciaConfiguracao, que também herda de Mecanismo, pode existir um MecanismoFeedback independente de MecanismoGerenciaConfiguracao, havendo assim a necessidade de manter independência entre a definição dos dois mecanismos.
- Associar cada mecanismo à sua estratégia correspondente: A sugestão de associação de cada mecanismo separadamente com suas respectivas estratégias não foi realizada, pois o relacionamento *atende* entre os conceitos PlanoGestao e Repositorio foi determinado e descrito de forma a suprir tal necessidade.

6.4. Segunda etapa da avaliação: não especialistas

Conforme mencionando anteriormente, na segunda etapa da avaliação, a ONTO-ResAsset foi avaliada por dois grupos:

- **Grupo 1**: composto por onze avaliadores não especialistas no domínio. Esses avaliadores fazem parte da equipe de desenvolvimento de uma empresa que foi indicada por conveniência e se candidataram como voluntários para participar da avaliação.
- **Grupo 2**: composto por nove alunos da disciplina de Desenvolvimento de Software do Programa de Pós-Graduação da Facom e que não possuem como especialidade o domínio de reuso de software.

A avaliação foi feita por meio da aplicação de um questionário, disponível no Apêndice E. O principal objetivo desse questionário é averiguar se a ONTO-ResAsset foi capaz de divulgar e transmitir o conhecimento sobre a especificação e gestão de ativos reutilizáveis. Para isso, as questões desse questionário foram formuladas tomando como base as questões de competência definidas no planejamento da ONTO-ResAsset (Seção 5.2), as quais estão relacionadas à especificação e gestão de ativos reutilizáveis durante as atividades de construção de software. Esse questionário foi submetido em uma avaliação piloto para observar

inconsistências, ambiguidades e redundâncias nas questões. Após a avaliação piloto, o questionário foi refinado.

Antes do treinamento e da avaliação da ONTO-ResAsset, cada avaliador respondeu um questionário de perfil, apresentado no Apêndice E, para obter informações quanto a formação, experiência e tempo de experiência no desenvolvimento de software, conhecimento da notação UML e da prática do reuso de software.

6.4.1. Resultados do grupo 1 – desenvolvedores de software

Na Figura 6.1 são apresentados os resultados das questões de perfil dos avaliadores do grupo 1, após a tabulação das respostas dos questionários respondidos. Apenas os cargos foram ocultados pois não houve qualquer correlação com as respostas do questionário de avaliação.

Observa-se que 90,9% dos avaliadores realizam algum tipo de reuso durante o desenvolvimento de software. Em relação ao nível de formação, 36,4% dos avaliadores possuem nível médio, 27,3% possuem nível superior, sendo a mesma porcentagem daqueles que possuem pós-graduação, e 9,1% possuem apenas nível técnico.

Quanto ao nível de conhecimento na notação UML, não houve nenhuma resposta para os níveis de conhecimento *nenhum* ou *alto*, sendo que 36,4% dos avaliadores possuem conhecimento razoável em UML e 63,6% tem pouco conhecimento dessa notação.

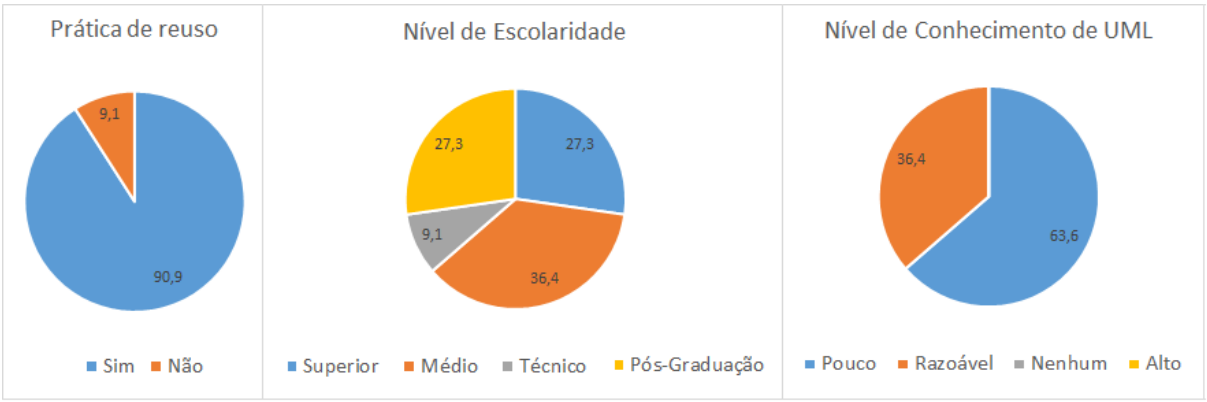


Figura 6.1 - Perfil dos avaliadores do grupo 1 (desenvolvedores)

Quanto ao tempo de experiência dos voluntários do grupo 1 no desenvolvimento de software, a média em meses foi de 79 meses, ou seja, aproximadamente seis anos e meio. O menor tempo de experiência é de 10 meses e o maior de 180 meses. Na Figura 6.2 é ilustrado

o tempo de experiência nas de fases requisitos, análise, projeto, implementação e teste, sendo a fase de implementação a que os voluntários possuem mais tempo de experiência.

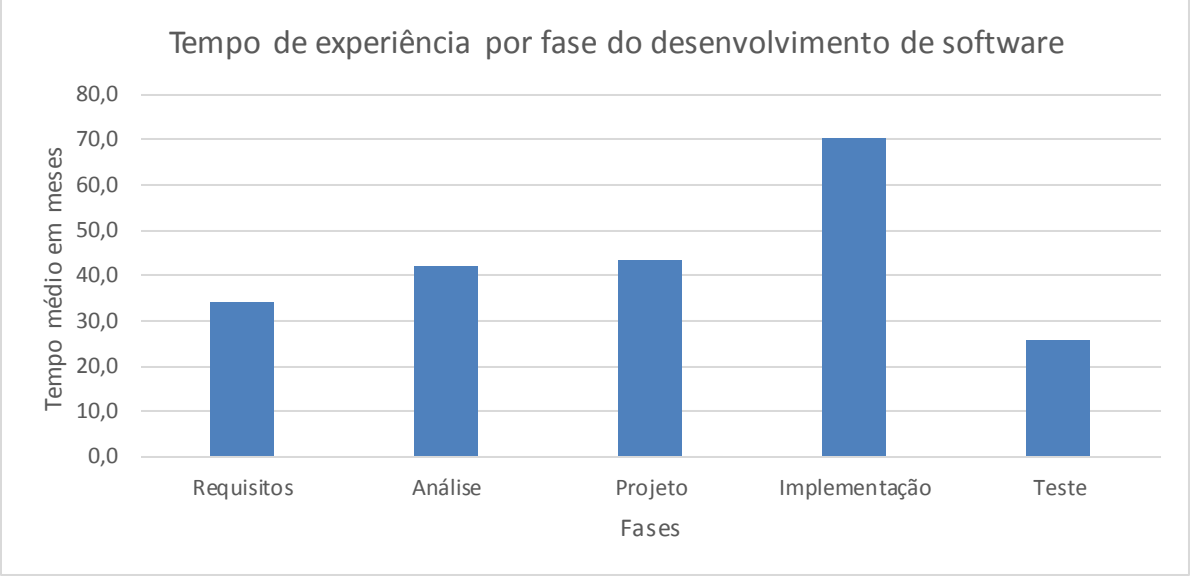


Figura 6.2 - Tempo de experiência em relação às fases do desenvolvimento de software do grupo 1 (desenvolvedores)

Cada questionário respondido pelos avaliadores não especialistas do grupo 1 foi avaliado com o objetivo de verificar a corretude das respostas das questões e, consequentemente, obter a quantificação do entendimento do domínio da ontologia pelos avaliadores. Cada resposta foi classificada como: correta, parcialmente correta, errada ou sem resposta.

Na Figura 6.3 são apresentados os resultados tabulados das respostas dos avaliadores não especialistas, conforme classificação supracitada. São mostradas a taxa de acerto de forma geral e as taxas para cada agrupamento das questões. Esse agrupamento foi apresentado na Seção 5.2.

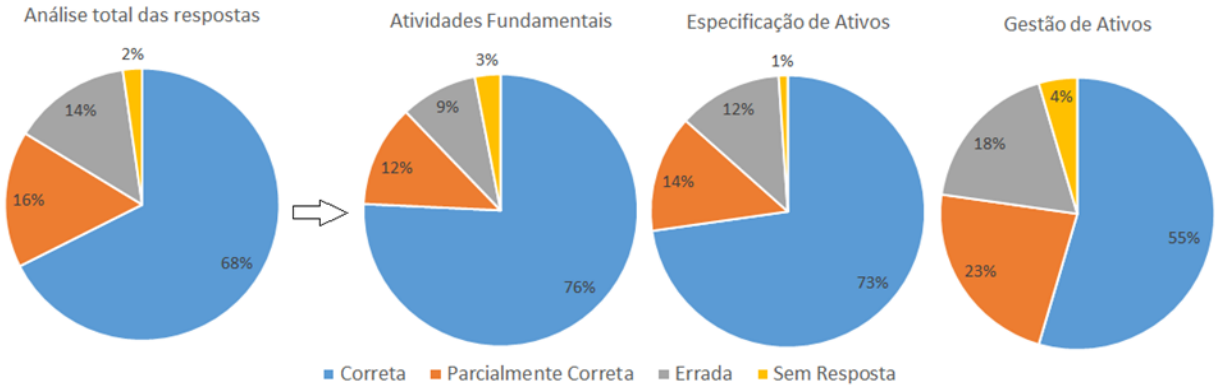


Figura 6.3 - Apresentação das taxas de acerto das questões da avaliação - grupo 1 (desenvolvedores)

A porcentagem total de acerto das questões foi de 68% e de acerto parcial foi de 16%. A resposta foi classificada como parcialmente correta quando apresentou partes dos conceitos esperados ou quando não apresentou todos os itens esperados como resposta correta. Além disso, observou-se que 14% das respostas estavam completamente incorretas e 2% das questões não foram respondidas.

A partir da análise total das respostas é possível inferir que os avaliadores conseguiram entender e responder com uma taxa de acerto satisfatória as questões de competência que nortearam a construção da ONTO-ResAsset.

Na Figura 6.4 é apresentada a quantidade de respostas corretas, parcialmente corretas, erradas e as questões que não foram respondidas pelos avaliadores em relação a cada questão.

Pode ser observado que as três questões com menor número de acerto, são as referentes a parte de gestão de ativos (indicadas por QC 23, 27 e 28), conforme comprovado pela Figura 6.3 que mostra que essa foi a parte que teve a menor taxa de acerto médio, ou seja, 55% de acerto, sendo menor que a das outras duas partes da ONTO-ResAsset. Isso pode indicar que os conceitos sobre especificação e gestão de ativos e os respectivos relacionamentos necessitam de melhor descrição e aprofundamento, ou que os avaliadores não responderam com atenção as questões sobre gestão de ativos por essas estarem localizadas na última parte do questionário, considerando o elevado número total de questões do mesmo.

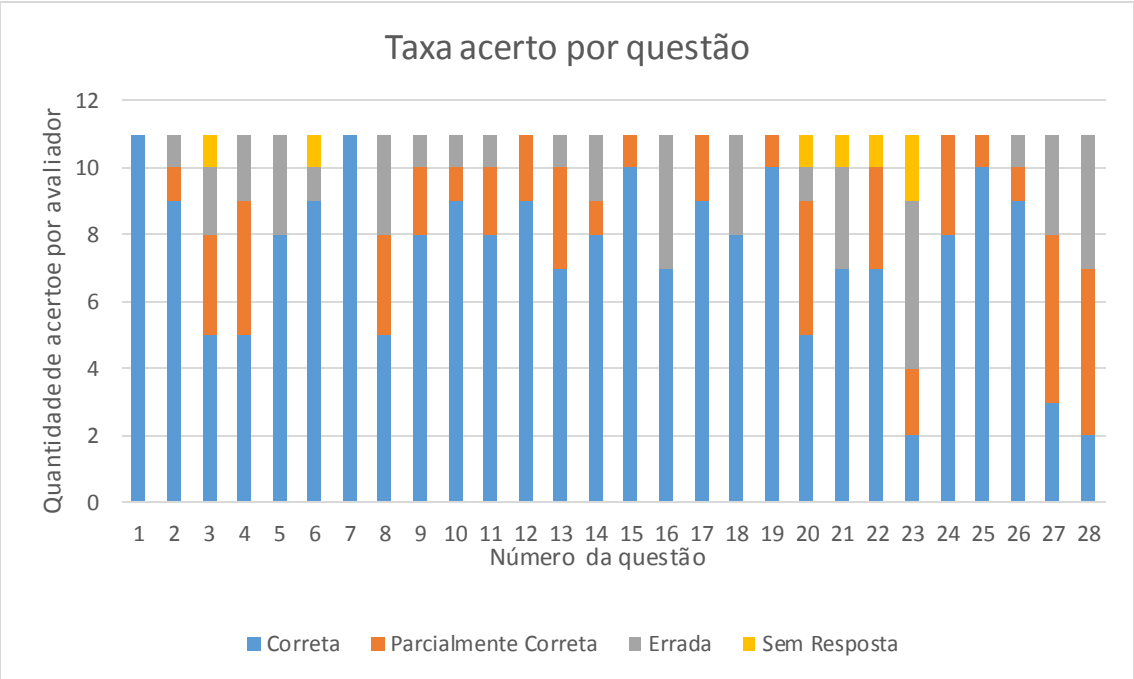


Figura 6.4 - Taxa de acerto por questão grupo 1 (desenvolvedores)

As questões indicadas por QC 3, 4, 8, e 20 também apresentaram taxa menor de acerto. Em relação as questões QC3 e QC4, essa taxa pode ser justificada pois a resposta correta possui diversos itens. Porém, considerando a taxa de acerto e a taxa de acerto parcial, em média 77% dos avaliadores observaram satisfatoriamente os conceitos da ONTO-ResAsset relacionados às questões QC3 e QC4.

Em relação as questões QC8 e QC20 observou-se que a taxa menor de acerto pode ser justificada devido ao enunciado das questões, que contém palavras que não representam conceitos da ONTO-ResAsset, podendo ter confundido o avaliador.

A partir dos resultados das demais questões observou-se que os avaliadores, foram capazes de obter uma compreensão melhor sobre a especificação e gestão de ativos de software e da sua gestão por meio da ONTO-ResAsset.

Além de questões específicas sobre a ontologia, o questionário possui três questões relacionadas às impressões dos avaliadores (questões 29, 30 e 31). A questão 29 questiona sobre o entendimento obtido quanto a especificação e gestão de ativos a partir da ONTO-ResAsset. A questão 30 objetiva averiguar se o conhecimento obtido por meio da ontologia auxiliará os avaliadores na definição de critérios para a seleção de repositórios e a questão 31 questiona se a ONTO-ResAsset pode ajudar no desenvolvimento de repositórios, em especial, no apoio a identificação dos requisitos e na definição da estrutura do repositório.

As respostas dos avaliadores do grupo 1 para essas questões são apresentadas na Figura 6.5. Salienta-se que dos onze avaliadores, oito responderam essas perguntas representando 72% dos avaliadores participantes do grupo 1.

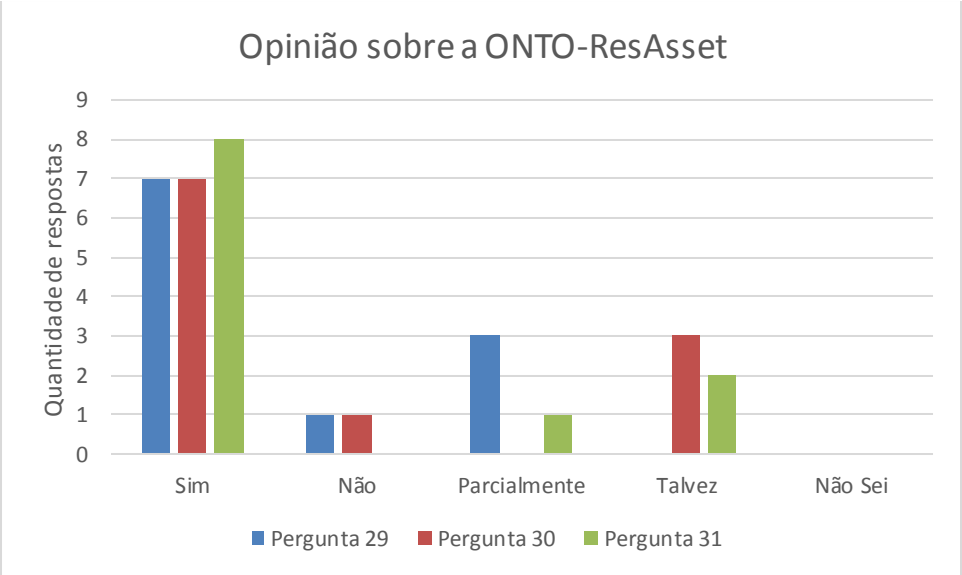


Figura 6.5 - Opinião sobre a ONTO-ResAsset - grupo 1 (desenvolvedores)

A partir das respostas obtidas observa-se que é possível afirmar que a ONTO-ResAsset proporcionou conhecimento para a maioria dos avaliadores sobre a especificação e gestão de ativo, podendo auxiliar na escolha ou na construção de repositórios de ativos reutilizáveis. Nenhum avaliador teve dúvida nas respostas.

6.4.2. Resultados do grupo 2 – alunos de pós-graduação

Na Figura 6.6 são apresentados os resultados das questões de perfil dos avaliadores do grupo 2, após a tabulação das respostas dos questionários respondidos. Observa-se que 37% dos avaliadores realizam algum tipo de reuso durante o desenvolvimento de software e 63% dizem não realizar qualquer prática do reuso de software.

Em relação ao nível de formação, lembrando que o grupo 2 de avaliadores é formado por alunos de pós-graduação de mestrado e doutorado, 50% dos avaliadores possuem o superior completo e 50% pós-graduado. Quanto ao nível de conhecimento na notação UML, não houve nenhuma resposta para o nível de conhecimento *nenhum*, sendo que 62% dos avaliadores possuem conhecimento *razoável*, 25% tem *pouco* conhecimento e 13% tem *alto* conhecimento.

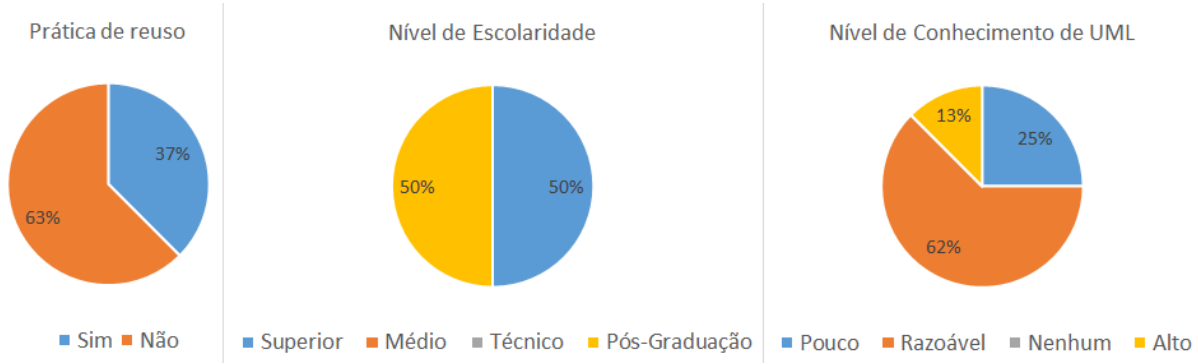


Figura 6.6 - Perfis dos avaliadores do grupo 2 (Alunos)

Quanto ao tempo de experiência dos voluntários do grupo 2 no desenvolvimento de software, a média em meses foi de 30 meses, ou dois anos e meio. O menor tempo de experiência é de 2 meses e o maior de 240 meses. Na Figura 6.7 é ilustrado o tempo de experiência dos voluntários nas de fases requisitos, análise, projeto, implementação e teste, sendo a fase de implementação a que os voluntários possuem mais tempo de experiência.

Com base nos resultados da avaliação do grupo 1, algumas questões do questionário de avaliação foram refinadas conforme apresentado no Apêndice F.

Figura 6.7 - Tempo de experiência em relação às fases do desenvolvimento de software do grupo 2 (alunos)

Cada questionário respondido pelos avaliadores não especialistas do grupo 2 foi avaliado com o objetivo de verificar a corretude nas respostas das questões e, consequentemente, obter a quantificação do entendimento do domínio da ontologia pelos avaliadores. Como conduzido na análise dos questionários do grupo 1, cada resposta dos questionários respondidos pelos avaliadores do grupo 2 foi classificada como: correta, parcialmente correta, errada ou sem resposta.

Na Figura 6.8 são apresentados os resultados tabulados das respostas dos avaliadores do grupo 2, conforme classificação supracitada. São mostradas a taxa de acerto de forma geral total e as taxas para cada agrupamento das questões, como apresentado na Seção 5.2.

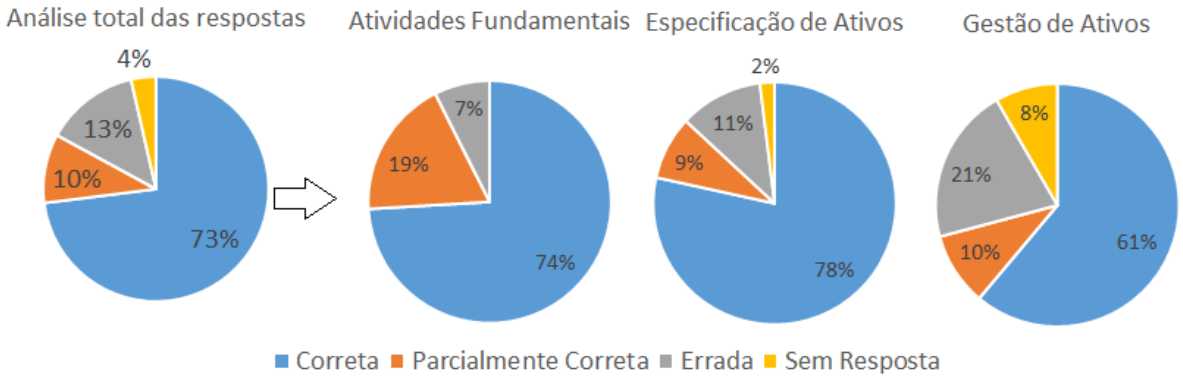


Figura 6.8 - Apresentação das taxas de acerto das questões da avaliação grupo 2 (alunos)

A porcentagem total de acerto das questões foi de 73% e de acerto parcial foi de 10%. Uma resposta é classificada como parcialmente correta quando apresenta partes dos conceitos esperados ou quando não apresentou todos os itens esperados como resposta correta. Além

disso, observou-se que 13% das respostas estavam completamente incorretas e 4% das questões não foram respondidas.

Na Figura 6.9 é apresentada a quantidade de respostas corretas, parcialmente corretas, erradas e que não foram respondidas pelos avaliadores do grupo 2, em relação a cada questão.

Pode-se notar que as questões com menor número de acerto, são as referentes a parte de gestão de ativos (indicadas por QC 20, 22 e 23), conforme comprovado pela Figura 6.8. Permitindo observar que, assim como no grupo 1, essa foi a parte que teve a menor taxa de acerto médio (61% de acerto), valor menor que o obtido nas outras duas partes. Isso pode indicar que os conceitos relacionados à especificação de ativos e aos mecanismos de gestão de ativos, bem como os respectivos relacionamentos, necessitam de melhor descrição e aprofundamento. Esses resultados podem também ser justificados considerando o elevado número total de questões do questionário e as questões sobre gestão de ativos estarem localizadas na última parte do questionário, possivelmente fazendo com que os avaliadores não respondessem com atenção essas questões.

A partir desses dados observou-se que os avaliadores do grupo 2, apesar de terem conhecimento na área de desenvolvimento de software e tempo de experiência menor que o grupo 1, foram capazes de obter uma compreensão melhor sobre a especificação e gestão de ativos de software e da sua gestão por meio da ONTO-ResAsset. Isso pode ser comprovado pela taxa de acerto relativamente superior em todas as partes analisadas.

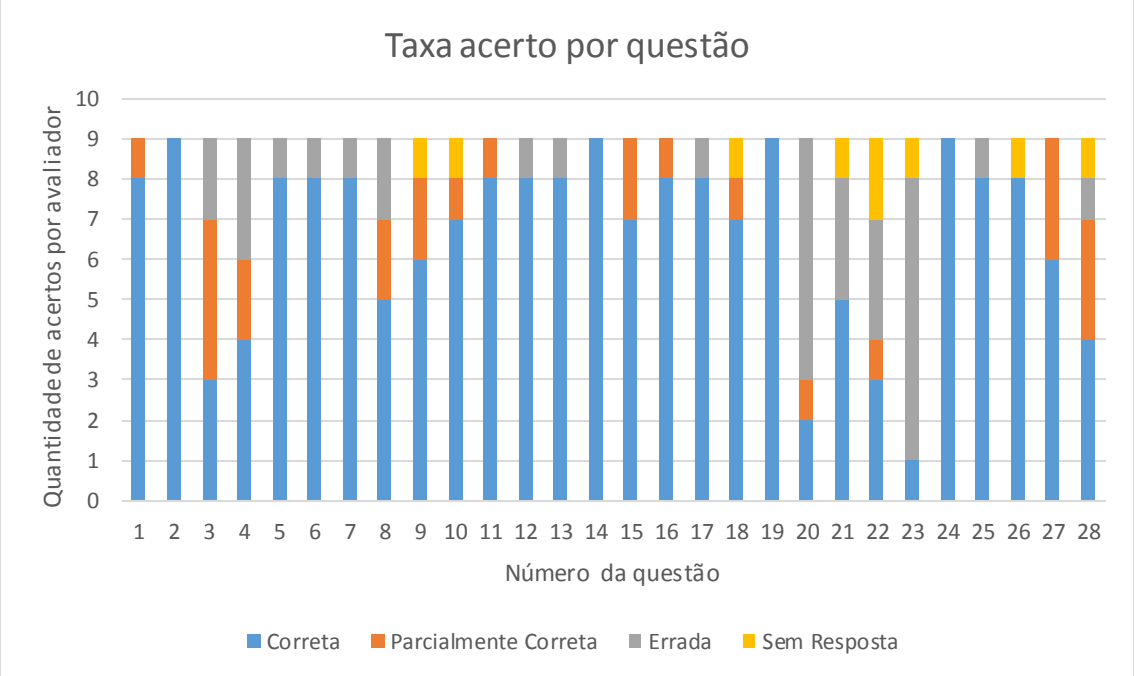


Figura 6.9 - Taxa de acerto por questão grupo 2 (alunos)

Além de questões específicas sobre a ontologia, o questionário possui três questões relacionadas às impressões dos avaliadores (questões 29, 30 e 31). A questão 29 questiona sobre o entendimento obtido quanto a especificação e gestão de ativos a partir da ONTO-ResAsset. A questão 30 objetiva averiguar se o conhecimento obtido por meio da ontologia auxiliará os avaliadores na definição de critérios para a seleção de repositórios e a questão 31 questiona se a ONTO-ResAsset pode ajudar no desenvolvimento de repositórios, em especial, no apoio a identificação dos requisitos e na definição da estrutura do repositório.

As respostas dos avaliadores do grupo 2 para essas questões são apresentadas na Figura 6.10, mostrando que a maioria das respostas nas três questões foi afirmativa.

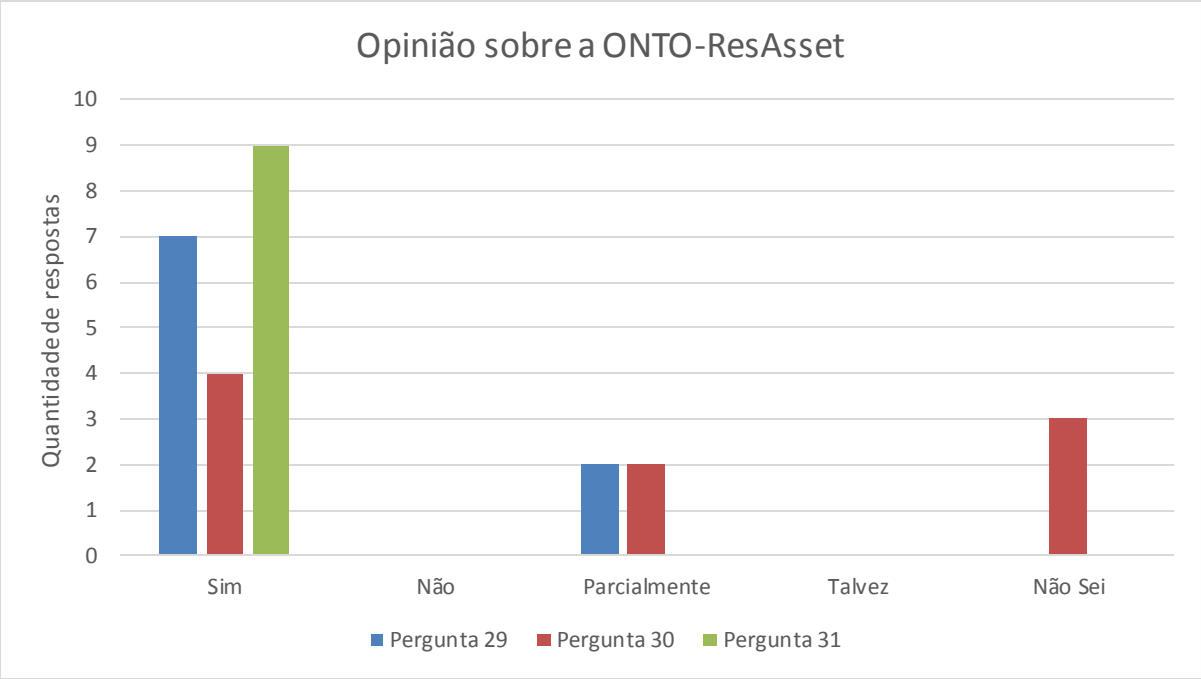


Figura 6.10 - Opinião sobre a ONTO-ResAsset para grupo 2 (alunos)

A partir das respostas obtidas observa-se que é possível afirmar que a ONTO-ResAsset proporcionou, assim como para o grupo 1, conhecimento sobre o domínio da ontologia, podendo auxiliar na escolha ou na construção de repositórios de ativos reutilizáveis.

6.5. Considerações finais

Neste capítulo foram apresentadas as duas etapas de avaliação da ONTO-ResAsset. Por meio da primeira etapa de avaliação, a ONTO-ResAsset pôde ser refinada tomando como base as recomendações dos especialistas no domínio, além de ser validada por pessoas externas ao projeto.

Os resultados obtidos na segunda etapa, em ambos grupos, apontam que a ontologia proposta é capaz de representar o conhecimento sobre a especificação e gestão de ativos reutilizáveis e pode apoiar na definição de critérios para a seleção de repositórios, bem como na definição de requisitos e da estrutura dos mesmos.

Salienta-se que devido ao número reduzido de avaliadores não especialistas não foi possível obter resultados com significância estatística.

No próximo capítulo é apresentada a conclusão do trabalho, discutindo as suas contribuições, limitações, dificuldades encontradas durante o seu desenvolvimento, bem como indicação de trabalhos futuros decorrentes do trabalho desenvolvido.

Capítulo 7. Conclusão

7.1. Considerações iniciais

Neste capítulo é apresentada a conclusão do trabalho realizado. Na Seção 7.2 são apresentadas as principais contribuições do trabalho. Na Seção 7.3 são elencadas as limitações observadas e as principais dificuldades encontradas durante o desenvolvimento deste trabalho, assim como as possíveis ações tomadas para contornar algumas dessas dificuldades. Por fim, na Seção 7.4, são indicados trabalhos futuros, decorrentes do trabalho realizado.

7.2. Contribuições do trabalho

A principal contribuição do trabalho é a ontologia semiformal ONTO-ResAsset, que explicita e documenta em um único local e em um único formato, conhecimento sobre a especificação e a gestão de ativos que podem ser reutilizados durante as atividades de construção do desenvolvimento de software, ou seja, requisitos, análise, projeto, implementação e teste.

Para a construção da ONTO-ResAsset, foram utilizados trabalhos relevantes como fontes de informação: ontologia de processo de desenvolvimento de software (FALBO, 1998), repositórios de reuso de software (EZRAN *et al.*, 2002), especificação de ativos reutilizáveis (OMG, 2005), processo de gestão de reuso de ativos (ABNT NBR ISO/IEC 12207, 2009), mapeamento sistemático de artefatos de software (KONDA e MANDAVA, 2010; SOMMERVILLE, 2011), e ontologia de repositórios de modelos de processos de negócios (FERREIRA e OLIVEIRA, 2013).

Como público-alvo que pode se beneficiar da ontologia desenvolvida neste trabalho podem ser citados os gerentes de ativos, desenvolvedores de software, analista de sistemas e qualquer outro interessado na área de reuso de software, em especial, na especificação e gestão de ativos reutilizáveis.

Adicionalmente, o desenvolvimento da ONTO-ResAsset permitiu observar a importância do uso de uma metodologia sistemática para possibilitar a construção de uma ontologia completa e correta, em conformidade com o seu propósito. Isso pode ser notado neste trabalho devido as poucas recomendações de melhorias, indicadas pelos avaliadores especialistas no domínio, se comparado com a quantidade de conceitos, relacionamentos e restrições representados na ONTO-ResAsset.

A partir da avaliação da ontologia proposta (Capítulo 6) observou-se também que ela apoia efetivamente na disseminação do conhecimento sobre a especificação e gestão de ativos. Esse conhecimento também pode ser utilizado para apoiar as empresas de desenvolvimento de software no estabelecimento de critérios para a seleção de repositórios de ativos reutilizáveis, bem como pode apoiar desenvolvedores de repositórios na definição dos requisitos e da estrutura desse tipo de software. Além disso, a partir da avaliação pode-se inferir que os conceitos relacionados a gestão de ativos necessitam de uma descrição mais detalhada e que a representação desses conceitos seja expandida.

Além disso, devido ao escopo bem delimitado da ONTO-ResAssset, é possível integrá-la a outras ontologias da área de conhecimento de domínio de reutilização de software.

Assim, observa-se que a ontologia ONTO-ResAsset contribui com as pesquisas e estudos na área de reutilização, pois reuni o conhecimento existente sobre especificação e gestão de ativos reutilizáveis durante as atividades de construção de software, disponível em diversas fontes de informação relevantes da literatura.

7.3. Dificuldades e limitações

Há na literatura diversas definições para a especificação de ativos, sendo que todas elas são em geral obtidas a partir de fontes acadêmicas e não possuem correspondência com conceitos relacionados à gestão de ativos e às atividades dos processos fundamentais de desenvolvimento de software. Para contornar isso, foram utilizadas como fontes de informação para a construção da ontologia ONTO-ResAsset o RAS da OMG (2005), que é a especificação padrão de ativos reutilizáveis, a documentação do processo gestão de reuso de ativos da ABNT NBR ISO/IEC 12207 (2009) e as atividades de construção da ontologia de processo de desenvolvimento de Falbo (1998).

Apesar da ONTO-ResAsset não ter sido documentada em uma ferramenta específica de construção de ontologias, como é o caso da ferramenta *Protégé* (NOY *et al.,* 2000), utilizou-se neste trabalho o modelo conceitual, de acordo com a notação do diagrama de classes da UML. Além da notação desse diagrama ser considerada uma alternativa para a representação de ontologias, também é a notação utilizada nos modelos do RAS. A descrição dos conceitos, relacionamentos entre eles e restrições é baseada em linguagem natural estruturada, sendo suficiente para documentar uma ontologia semiformal em conjunto com o modelo conceitual.

Na ONTO-ResAsset podem ser encontrados grupos taxonômicos que não foram explorados, como por exemplo, os tipos dos artefatos, os tipos de atividades de construção, os

tipos de soluções e os tipos de mecanismos. Tais grupos podem ser detalhados em novas ontologias capazes de representá-los em um nível maior de detalhamento e de profundidade.

Apesar de definida a classe associativa ArtefatoSolucao, que proporciona a identificação de quais atividades de construção de software criam ou utilizam os artefatos, não estão explícitos na ONTO-ResAsset os relacionamentos (propriedades) do conceito <Artefato> que possam por si só diferenciar mais claramente informações sobre a atividade que criou ou as atividades que reutilizaram cada artefato.

Do ponto de vista da avaliação, não foi possível obter resultados com significância estatística na segunda etapa da avaliação devido ao número reduzido de avaliadores não especialistas envolvidos. Além disso, a segunda etapa da avaliação não contemplou uma avaliação específica e direcionada para averiguar como e quanto a ONTO-ResAsset pode apoiar a definição de critérios para a seleção de repositórios de ativos reutilizáveis, bem como para apoiar na definição dos requisitos e da estrutura de repositórios de ativos reutilizáveis. Como ameaça à validade dessa segunda etapa da avaliação pode ser destacada a seleção da empresa de desenvolvimento, na qual o autor desse trabalho faz parte do quadro de funcionários, bem como a seleção da disciplina Desenvolvimento de Software, na qual a orientadora deste trabalho foi a professora responsável.

7.4. Trabalhos futuros

Como trabalhos futuros decorrentes do trabalho desenvolvido podem ser destacados os elencados a seguir:

- Implementar a ONTO-ResAsset em uma ferramenta específica de construção de ontologia, como é o caso da ferramenta *Protégé* que suporta a linguagem ontológica OWL;
- Expandir a ontologia ONTO-ResAsset com a criação de subontologias específicas para representar com maior profundidade de detalhes os grupos taxonômicos, indicados pelos conceitos AtividadeConstrucao, Artefato, Solucao, Mecanismo, Modelo, AtivoRelacionado e DependenciaArtefato;
- Melhorar a distinção entre os tipos de artefatos buscando representá-los de forma mais clara e com propriedades distintas para diferenciá-los por si só, sem necessidade de um relacionamento com conceitos mais distante;
- Avaliar a ONTO-ResAsset com um número maior de avaliadores não especialistas para obter resultados com significância estatística;

- Definir uma avaliação específica para averiguar o apoio da ONTO-ResAsset na definição de critérios para a seleção de repositórios de ativos reutilizáveis;
- Definir uma arquitetura de referência de repositórios de ativos reutilizáveis tomando como base a ONTO-ResAsset;
- Desenvolver um repositório de ativos reutilizáveis tomando como base a arquitetura de referência proposta no item anterior; e
- Utilizar a ONTO-ResAsset como fonte de informação para o desenvolvimento de novos trabalhos que visam compartilhar e especificar conhecimento na área de reutilização de software.

Referências

ABNT NBR ISO/IEC 12207. (2009), Engenharia de sistemas e software - processos de ciclo de vida de software, ABNT.

Almeida, E. d., Alvaro, A., Garcia, V. C., Mascena, J. C., Burégio, V. A., Nascimento, L. M., (2007). *C.R.U.I.S.E. Component Reuse In Software Engineering*. Cesar e-books.

Antoniou, G., & Harmelen, F. v. (2009). Web Ontology Language: OWL. In: S. Staab, & R. Studer, *Handbook on Ontologies* (pp. 91-110). Springer Berlin Heidelberg.

Apperly, H. (2001). Configuration Management and Component Libraries. In: G. T. HEINEMAN, & W. T. COUNCILL, *Component based software engineering: putting the pieces together*.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. (2001). OilEd: a Reason-able Ontology Editor for the Semantic Web. In: *Joint German/Austrian conference on Artificial Intelligence* (pp. 396-408). Vienna: Springer-Verlag LNAI.

Beck, K. (1999). *eXtreme Programming eXplained: Embrace Change*. Reading: Addison-Wesley.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.

Bernstein, P. A., & Dayal, U. (1994). An overview of repository technology. In: *Proceedings of VLDB 1994*, (pp. 707-713). Santiago de Chile, Chile.

Booch, G.; Rumbaugh, J. Jacobson, I. UML: Guia do Usuário, Editora Campus, 2006.

Braga, R. M., Mattoso, M., & Werner, C. M. (2001). The use of mediation and ontology technologies for software component information retrieval. *SIGSOFT Softw. Eng. Notes*, (pp. 19-28). Toronto, Ontario, Canada.

Burégio, V. A. (2006). *Specification, Design and Implementation of a Reuse Repository*. Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife (PE).

Caldiera, G., & Basili, V. R. (1991). Identifying and qualifying reusable software components. *IEEE Computer, 24*, 61-70.

Chacon, S. (2009). *Pro Git*. Disponível em http://git-scm.com/book, Acesso em 20 de agosto de 2012.

Corcho, O., Fernandez-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data & Knowledge Engineering* (46), 41-64.

Councill, B., & Heineman, G. T. (2001). Definition of a software component and its elements. In: *Component-based software engineering* (pp. 5-19). Addison-Wesley Longman Publishing Co., Inc.

Cranefield, S., & Purvis, M. (1999). UML as an ontology modelling language. University of Otago, Dunedin, Nova Zelândia.

Daltio, J. (2007). *Aondê: Um Serviço Web de Ontologias para Interoperabilidade em Sistemas de Biodiversidade* (Aondê: An Ontology Web Service for Interoperability across Biodiversity Information Systems. Dissertação de Mestrado, Unicamp, Instituto de Computação, Campinas (SP).

Delamaro, M. E., Maldonado, J. C., & Jino, M. (2007). *Introdução ao teste de software*. Elsevier.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology & Design (1ª Ed.)*. Prentice Hall.

Ezran, M., Morisio, M., & Tully, C. (2002). *Practical Software Reuse*. Springer.

Falbo, R. A. (1998). *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese de Doutorado, 205p. Doutorado em Ciências em Engenharia de Sistemas e Computação. Programa de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro (RJ).

Fayad, M., & Schmidt, D. C. (1997). Object-oriented application frameworks. *Commun. ACM, 40*(10), 32-38.

Fensel, D. (2003). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce (2ª Ed.). Springer-Verlag.

Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., & Klein, M. (2000). OIL in a Nutshell. In: R. Dieng, & O. Corby (Eds.), *Knowledge Engineering and Knowledge Management Methods, Models, and Tools* (Vol. 1937, pp. 137-154). Springer Berlin / Heidelberg.

Ferreira, A. H. A. S., Oliveira, A. M. (2013). *Ontologia de Repositórios de Modelos de Processos de Negócio*. Trabalho de Conclusão de Curso, Universidade Federal do Mato Grosso do Sul, Campo Grande (MS).

Férnandez-López, M., Gómez Pérez, A., & Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series* (pp. 33-40). American Asociation for Artificial Intelligence.

Frakes, W. B., & Fox, C. J. (1995). Sixteen Questions about Software Reuse. *Communications of the ACM, 38*(6), 75-87.

Frakes, W. B., & Kang, K. (2005). Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering, 31*(7), 529-536.

Frakes, W., Prieto-Diaz, R., & Fox, C. (1998). DARE: Domain analysis and reuse environment. *In: 5th Annals of Software Engineering*, (pp. 125–141).

Gruber, T. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies, 43*(5-6), 907-928.

Gruber, T. R. (1992). *Ontolingua: A mechanism to support portable ontologies.* Stanford University, California, Estados Unidos.

Grunninger, M., & Fox, M. S. (1994). *The Role of Competency Questions in Enterprise Engineering*. Department of Industrial Engineering, University of Toronto, Toronto.

Guizzardi, G., Wagner, G., Herre, (2004). On the foundations of uml as an ontology representation language. In: Engineering Knowledge in the Age of the Semantic Web. Springer Berlin Heidelberg, 2004. p. 47-62.

Harmelen, F. v., Patel-Schneider, P. F., & Horrocks, I. (2001). *Reference description of the DAML+OIL (March 2001) ontology markup language*. Disponível em http://www.daml.org/2001/03/reference.html, Acesso em 25 de 07 de 2012.

Hendler, J., & McGuinness, D. L. (2000). *The DARPA agent markup language*. Disponível em ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-00-10.html, Acesso em 20 de 08 de 2012.

IBM. (2006). *alphaWorks technology: BPEL Repository*. Disponível em http://www.ibm.com/developerworks/webservices/library/ws-awbpelrepos/index.html, Acesso em 20 de 08 de 2012

IBM. (2008). *IBM Rational RequisitePro*. Disponível em http://public.dhe.ibm.com/common/ssi/ecm/en/rad10955usen/RAD10955USEN.PDF, Acesso em 20 de 08 de 2012.

IEEE. (1997). *1074-1995 - IEEE Standard for Developing Software Life Cycle Processes.* Institute of Electrical and Electronics Engineers, New York, Estados Unidos.

Jamwal, D. (2010). Software Reuse: A Systematic review. In: *4th National Conference INDIACom-2010.* New Delhi, India.

Jorge, E. (2012). *Requirements and Tests Management Repository*. Disponível em http://rtmr.net/en/index.html, Acesso em 20 de 08 de 2012.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., et al. (2002). UML for ontology development. *The Knowledge Engineering Review, 17*(1), 61-64.

Konda, B. M., & Mandava, K. K. (2010). *A systematic mapping study on software reuse*. Department of System and Software Engineering, Blekinge Institute of Technology, Sweden.

Krueger, C. W. (1992). Software Reuse. *ACM Computer Survey, 24*, 131-183.

Lucrédio, D., Brito, K. d., Alvaro, A., Garcia, V. C., Fortes, R. P., & Meira, S. L. (2008). Software reuse: The Brazilian industry scenario. *Journal of Systems and Software, 81*(6), 996-1013.

Luke, S., & Heflin, J. (2000). *SHOE 1.01*. University of Maryland, Institute for Advanced Computer Studies, Baltimore, Estados Unidos.

Ma, Y., He, K., Liu, W., & Tian, J. (2007). A Grid-Oriented Platform for Software Component Repository Based on Domain Ontology. *IEEE International Conference on Services Computing (SCC 2007)*, pp. 628-635.

Martínez, A., Ristuccia, C., Pisarello, R., Stubbs, E., Caminotti, L., Balparda, J., et al. (2004). Las categorías o facetas fundamentales: una metodología para el diseño de taxonomías corporativas de sitios Web argentinos. *Ciência da Informação, 33*(2), 106-111.

McILROY, M. D. (1968). Mass-Produced Software Components. *1st International Conference on Software Engineering*, (pp. 88-98). Garmisch-Partenkirchen, Alemanha.

Mizoguchi, R. (2004). Tutorial on ontological engineering Part 2: Ontology development, tools and languages. *New Generation Computing, 22*(1), 61-96.

Mizoguchi, R., Vanwelkenhuysen, J., & Ikeda, M. (1995). Task Ontology for Reuse of Problem Solving Knowledge. In: N. J. Mars, *Towards Very Large Knowledge Bases* (pp. 46-59). ISO Pres. Amsterdam, Holanda.

Moore, M. M. (2001). Software Reuse: Silver Bullet? *IEEE Software*, 86.

Morisio, M., Ezran, M., & Tully, C. (2002). Success and failure factors in software reuse. *IEEE Transactions on Software Engineering, 28*(4), 340-357.

Noy, N., Fergerson, R., Musen, M., Dieng, R., & Corby, O. (2000). The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In: *Knowledge Engineering and Knowledge Management Methods, Models, and Tools* (Vol. 1937, pp. 69-82). Springer Berlin / Heidelberg.

OMG. (2005). *Reusable Asset Specification*. Object Management Group.

Palmer, S. R.; Felsing, J. M. (2002). *A Practical Guide to Feature-Driven Development*. Englewood Cliffs: Prentice Hall.

Prieto-Diaz, R., & Arango, G. (1991). Domain Analysis and Software System Modeling. *IEEE Computer Society Press Tutorial*.

Schwaber, K.; Beedle, M. (2001). *Agile Software Development with Scrum*. Englewood Cliffs: Prentice Hall.

Singh, S., & Singh, G. (2010). Reusability of the Software. *International Journal of Computer Applications, 7*(14), 38-41.

SOA, S. I. (2012). *Repository Manager - SDA Mapping and Discovery*. Disponível em http://www.soa.com/products/repository_manager/, Acesso em 20 de 08 de 2012.

Sommerville, I. (2011). *Engenharia de Software* (9ª Edição ed.). Pearson Education.

SRI, I. (1995). *Open Knowledge Base Connectivity Home Page*. Disponível em http://www.ai.sri.com/~okbc/, Acesso em 21 de 08 de 2012.

Staab, S., Schnurr, H., Studer, R., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems, I*(16), 26-34.

Studer, R., Benjamins, R., & D., F. (1998). Knowledge Engineering: Principles and Methods. *IEEE Trans. on Data and Knowledge, 25*(1-2), 161-197.

Su, X., & Ilebrekke, L. (2006). A Comparative Study of Ontology Languages and Tools. In: A. Pidduck, M. Ozsu, J. Mylopoulos, & C. Woo (Eds.), *Advanced Information Systems Engineering* (Vol. 2348, pp. 761-765). Springer Berlin / Heidelberg.

Uschold, M. (1996). Building Ontologies: Towards A Unified Methodology. In 16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems.

Uschold, M., & King, M. (1995). Towards a Methodology for Building Ontologies. *Workshop on Basic Ontological Issues in Knowledge Sharing.*

van Ommering, R. (2005). Software reuse in product populations. *IEEE Transactions on Software Engineering, 31*(7), 537-550.

W3C. (2004a). *OWL Web Ontology Language Reference*. Disponível em http://www.w3.org/TR/owl-ref/, Acesso em 25 de 07 de 2012.

W3C. (2004b). *RDF/XML Syntax Specification (Revised)*. Disponível em http://www.w3.org/TR/REC-rdf-syntax/, Acesso em 25 de 07 de 2010.

W3C. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Disponível em http://www.w3.org/TR/REC-xml/, Acesso em 25 de 07 de 2012.

W3C. (2009). *OWL 2 Web Ontology Language New Features and Rationale*. Disponível em http://www.w3.org/TR/2009/REC-owl2-new-features-20091027/, Acesso em 25 de 07 de 2012.

Werner, C., Murta, L., Lopes, M., Dantas, A., Lopes, L., Fernandes, P., et al. (2007). Brechó: Catálogo de Componentes e Serviços de Software. In: *XXI Simpósio Brasileiro de Engenharia de Software*, (pp. 24-30). João Pessoa (PB).

Yao, H., & Etzkorn, L. (2004). Towards a semantic-based approach for software reusable component classification and retrieval. In: *Proceedings of the 42nd annual Southeast regional conference*, (pp. 110-115). Huntsville, Alabama, Estados Unidos.

Apêndice A Análise de Domínio de Repositórios

Neste apêndice são apresentadas a análise de domínio de Repositórios de Projeto de Software (Seção A.1) e a análise de domínio de Repositórios de Componentes de Software (Seção A.2).

Conforme definido pelo RAS da OMG (2005), um ativo é composto por diversos artefatos. Somente os artefatos comuns aos três repositórios selecionados em cada análise de domínio conduzida serão considerados. Neste apêndice, tanto artefato quanto ativo serão chamados de "elemento". São considerados apenas os metadados dos elementos comuns aos repositórios analisados.

As funcionalidades em comum, identificadas nos repositórios analisados, são categorizadas em três tipos de acordo com as atividades de produção, consumo e gerenciamento. São consideradas como atividade de **produção** todas as funcionalidades que criam e/ou modificam/atualizam o conteúdo ou metadados de algum elemento do repositório; as atividades de **consumo** são aquelas responsáveis pela busca, navegação e recuperação do conteúdo do elemento e/ou suas informações; e as atividades de **gerenciamento** que são aquelas que monitoram, alteram e/ou controlam fluxos e operações do repositório em si.

Todas as informações resultantes da análise de domínio foram obtidas por meio da utilização dos repositórios como um usuário comum. Algumas funcionalidades são realizadas de maneira implícita durante a execução de outras. Tais funcionalidades, se presentes nos outros repositórios analisados, também são considerados no resultado final da análise.

A.1 Análise de Domínio de Repositórios de Projeto de Software

Nesta análise de domínio foram analisados os seguintes repositórios de projeto de software por serem os mais utilizados pela indústria, e conhecidos pela comunidade de desenvolvimento, principalmente de software livre:

- ***Google Code***: disponível em http://code.google.com, acesso em Agosto de 2012
- ***SourceForge***: disponível em http://sourceforge.net, acesso em Agosto de 2012
- ***GitHub***: disponível em https://github.com, acesso em Agosto de 2012

O ativo principal desse tipo de repositório são os projetos de software.

Os elementos comuns aos repositórios de projetos de software analisados são: projeto, código fonte, tarefas (*tickets* ou *issues*) e *wiki*.

No Quadro A.1 são apresentadas as funcionalidades comuns a repositórios de projetos de software, identificadas a partir da análise de domínio e agrupadas de acordo com o tipo de funcionalidades: de consumo, de produção e de gerenciamento.

Quadro A.1 – Funcionalidades comuns aos repositórios de projeto de software

Nome	Tipo de Função	Descrição
Buscar Projeto por Palavra	Consumo	Realiza a busca dos projetos de acordo com termos que são informados para a busca
Listar Projetos		Permite a navegação entre os projetos no repositório
Atualizar/Fechar *Ticket*		Atualiza o status do *ticket* criado, como aceito, fechado ou reabri-lo por exemplo
Listar/Navegar Páginas *Wiki*		Lista e permite a navegação entras as páginas *wikis* do projeto
Buscar *Ticket*		Realiza a busca dos *tickets* criados
Navegar/Listar *Tickets*		Navega/Lista os *tickets* abertos
Navegar/Ver Códigos		Navega/Visualiza códigos do projeto
Histórico/*diff* nos Códigos		Disponibiliza o histórico de revisões dos códigos e compara as diferenças
Subscrever Notificações		Subscreve o usuário para o recebimento de alterações ou notificações do projeto ou de algum artefato específico
Navegar/Ver *Commits*		Navega ou lista pelos *commits* dos códigos no repositório
Diferenciar *Commits*		Mostra as diferenças entre as revisões selecionadas
Obter Códigos		Permite obter os códigos do repositório e transferi-los para o computador local do usuário
Administrar Projeto	Gerenciamento	Permite o gerenciamento das configurações e metadados do projeto; altera Artefato
Compartilhar Projeto		Compartilha o projeto com outros usuários do repositório
Sair do Projeto		Permite a retirada do usuário do projeto no repositório, por ele mesmo
Estatística de Uso		Permite a visualização de algumas estatísticas de uso do repositório
Criar Projeto	Produção	Realiza a instanciação do artefato projeto no repositório de projetos; cria artefato
Criar Página *Wiki*		Cria uma nova página *wiki* pertencente ao projeto
Previsualizar Página *Wiki*		Cria uma pré-visualização da página *wiki* que está sendo criada ou editada
Editar *Wiki*	Produção	Permite alteração de uma *wiki* já existente no repositório
Criar *Ticket*		Cria um novo *ticket* no repositório

Nome	Tipo de Função	Descrição
Editar Códigos		Permite edição de códigos dos projetos diretamente no repositório
Anexar Arquivos *Ticket*		Anexa os arquivos aos *tickets* criados, cada repositório define regras para os tipos de arquivos e quantidade máxima em cada tipo de *ticket*

No Quadro A.2 são apresentados os artefatos comuns aos repositórios de projetos de software, identificados a partir da análise de domínio, assim como os metadados correspondentes a cada um desses artefatos.

Quadro A.2 - Artefatos e metadados comuns aos repositórios de projeto de software

Elemento	Nome	Descrição	Vinculado ao Artefato
Artefato	Projeto		
Metadado	Nome Projeto	Nome do projeto criado	Projeto
Metadado	Descrição	Descreve brevemente o projeto	
Metadado	Licença	Tipo de licença que o projeto possui, como, Apache, GNU GPL, MIT, *Creative Commons*, etc.	
Metadado	Palavras-Chave	Palavras que determinam características chave do projeto	
Artefato	*Issue (Tickets)*		
Metadado	Título do *ticket*	Nome do *ticket* que é criado	Issue
Metadado	Descrição	Descrição textual do *ticket*	
Metadado	Responsável	Pessoa que será atribuída a responsabilidade de fornecer a solução ou leitura do *ticket*	
Metadado	Palavras-Chave	Palavras que determinam características chave sobre o *ticket* que está sendo criado	
Metadado	Status	Situação sobre o andamento da solução do *ticket*	
Metadado	Número	Número atribuído para acompanhamento do *ticket*, em geral, esse número é definido em ordem cronológica de criação	
Metadado	Anexo	Arquivo(s) anexado(s) ao *ticket*	
Artefato	Código		
Metadado	Nome Código	Nome do arquivo, por exemplo: “main.cpp”	Código
Metadado	Data Alteração	Data da última alteração do código no repositório	
Metadado	Autor	Nome(s) da(s) pessoa(s) que criou o código	
Metadado	*Log Commit*	Mensagens de informações sobre cada atualização do código	
Artefato	*Wiki*		
Metadado	Nome Página	Nome da página *wiki*	*Wiki*
Metadado	Descrição Alterações	Mensagens de informações sobre cada alteração realizada na página *wiki*	*Wiki*

A.2 Repositório de Componentes

Nesta análise de domínio foram analisados os seguintes repositórios de componentes devido a serem bem aceitos pela comunidade de desenvolvimento, por serem gratuitos e no caso do *Brécho* por ser de origem acadêmica e brasileira:

- ***Eclipse Maketplace***: disponível em http://marketplace.eclipse.org/, acesso em Agosto de 2012
- **Brécho**: disponível em http://reuse.cos.ufrj.br/brecho, acesso em Agosto de 2012
- ***Component Source***: disponível em http://www.componentsource.com, acesso em Agosto de 2012

O ativo principal desse tipo de repositório são os componentes de software.

A análise dos repositórios de componentes foi realizada levando em consideração uma conta de usuário comum gratuita. Devido a isso, algumas funcionalidades podem não ser observada. Por exemplo, no repositório *Component Source*, o envio de um novo componente não pode ser observado e analisado, pois necessitava da ativação de programa de filiação.

No Quadro A.3 são apresentadas as funcionalidades comuns aos repositórios de componentes de software, identificadas a partir da análise de domínio e agrupadas de acordo com o tipo de funcionalidade: de produção, de consumo e de gerenciamento.

Quadro A.3- Funcionalidades comuns aos repositórios de componentes de software

Nome	Tipo de Função	Descrição
Criar conta	Gerenciamento	Cria uma nova conta de usuário no repositório.
Editar Conta		Altera as informações sobre uma determinada conta de usuário.
Adicionar Componente	Produção	Adiciona um novo componente ao repositório.
Buscar Componente	Consumo	Realiza a busca de componentes por algum termo, que é baseado em algum metadado do componente.
Listar categorias		Permite a listagem de componentes de acordo com algum tipo de categorização.
Ver Componente		Mostra as informações sobre o componente
Recuperar Componente		Retorna o componente do repositório de acordo com o seu formato e empacotamento ou permite a importação direta por alguma outra ferramenta.

No Quadro A.4 são apresentados os metadados comuns aos componentes que foram possíveis obter pela análise de domínio.

Quadro A.4 - Metadados dos Componentes de Software

Elemento	Nome	Descrição	Vinculado ao Artefato
Artefato	Componente		
Metadado	Nome do Componente	Nome do componente	Componente
Metadado	Descrição	Descrição textual sobre o componente	
Metadado	Categoria	Categorias que o componente faz parte	
Metadado	Data de Adição	Data de inclusão do componente no repositório	
Metadado	Publicador	Usuário que inseriu o componente no repositório	
Metadado	Versão	Versão do componente inserido	

Apêndice B Glossário Geral de Termos da ONTO-ResAsset

Este apêndice apresenta no Quadro B.5 todos os termos presentes na ONTO-ResAsset, ordenados alfabeticamente e suas respectivas descrições.

Quadro B.5 - Lista de conceitos da ONTO-ResAsset em ordem alfabética

Termo	Descrição Semântica
Agregacao	É um tipo de AtivoRelacionado; denota a ideia de que um ativo contém outro ativo.
Algoritmo	Artefato que representa a solução para algum problema, em geral, utilizado em projetos (design) de software.
Artefato	Um artefato é qualquer produto que pode ser criado, armazenado e manipulado por produtores/consumidores ou por uma ferramenta, dentro de um processo de desenvolvimento de software.
ArtefatoSolucao	Representa a relação entre a Solução e o Artefato que a realiza. Delimita essa relação de acordo com um contexto de relevância do artefato.
Atividade	São as ações que um consumidor (por ex. um desenvolvedor ao utilizar o ativo) ou mesmo uma ferramenta (computacional) pode ou deve realizar quando utiliza o ativo. Uma atividade pode ser composta por outras atividades e cada atividade deve ser desempenhada por um papel. Ex: Um componente C, para ser compilado corretamente, necessita que o ponto de variabilidade X (que personaliza seu construtor) seja determinado, e a Atividade Implementação é quem determina isso.
AtividadeAnalise	É uma atividade de construção onde os requisitos são refinados e normalmente é criado um modelo de classes conceitual, que permite a representação dos conceitos do sistema que está sendo desenvolvido. A análise pode ainda ser mais formal quando se cria os modelos de interação (diagramas de sequência ou de comunicação da UML) e o diagrama de estados, gerando como resultados modelos de análise. Essa atividade cria uma solução de análise.
AtividadeArtefato	São as atividades que estão associadas a um artefato específico, podem existir diversas atividades para um mesmo artefato, assim como tal atividade do artefato pode ser relevante em um ou mais contextos do artefato.
AtividadeArtefatoContexto	Representa um agrupamento do contexto de relevância para um conjunto de Atividades do Artefato. Ou seja, agrupa quais AtividadeArtefato são relevantes em determinados contexto do artefato.
AtividadeAtivo	São as atividades que estão relacionadas ao ativo como um todo, podendo ser agrupadas a outras atividades de ativo. Por exemplo, tendo o Ativo A, ele possui a AtividadeAtivo Aa que determina que deve-se ajustar os parâmetros do método construtor dos dois artefatos que compõem A, os Componentes C1 e C2.
AtividadeAtivoContexto	Representa um agrupamento do contexto de relevância para um conjunto de atividades do ativo, ou seja, agrupa quais AtividadeAtivo são relevantes em determinados contexto do ativo.
AtividadeConstrucao	São as atividades principais de desenvolvimento de software, que são realizadas por pessoas que possuem papéis definidos dentro do processo de desenvolvimento e cria ou atualiza como resultado de sua execução um artefato.

Termo	Descrição Semântica
AtividadeImplementacao	É uma atividade de construção que é responsável pela conversão do projeto do sistema em um sistema executável por meio de uma linguagem de programação. O código-fonte, módulos e componentes são produzidos tomando como base os recursos criados na atividade de projeto, gerando uma solução de implementação.
AtividadeProjeto (Design)	É uma atividade de construção em que se define o projeto de software, de maneira a suprir o que até então já foi definido nas atividades anteriores, entretanto, nessa atividade o modelo deve descrever características comportamental e estrutural do sistema levando em consideração fatores como a linguagem, sistema operacional, banco de dados, interface do usuário, etc. Nessa atividade devem ser especificados os elementos como arquitetura do software, interface do usuário, componentes, estrutura de dados e algoritmos. Sommerville (2011) indica os seguintes artefatos que podem ser elaborados: modelos de objetos (classes e dependência), de interação entre os objetos, de transição de estados, documentos sobre componentes e agregações e modelos de fluxo de dados. Essa atividade cria uma solução de projeto.
AtividadeRequisitos	É uma atividade de construção em que os requisitos do software são levantados por meio da identificação das necessidades de um interessado (stakeholder). Em geral, segundo Sommerville (2011), os requisitos são apresentados em documentos de requisitos, em documentos de casos de uso ou em outros tipos de documentos. Essa atividade cria uma solução de requisitos.
AtividadeTeste	É uma atividade de construção responsável por avaliar, de maneira sistemática, o que foi construído nas outras fases, com o objetivo de averiguar a conformidade com os requisitos, e/ou busca de problemas ainda não descobertos que devem ser corrigidos. Nessa atividade são desenvolvidos o plano de testes e os casos de teste. Essa atividade cria uma solução de teste.
Ativo	É o elemento principal da especificação de um ativo reutilizável. Descreve e empacota os artefatos e provê a documentação sobre como e qual problema eles resolvem.
AtivoRelacionado	Um ativo pode estar relacionado a outros ativos. Existem quatro tipos de relacionamentos entre ativos: agregação, similaridade, dependência e generalização.
Caso de Teste	É formado por um dado de entrada e uma saída esperada e objetiva efetuar avaliação de software e/ou de componentes de software de acordo com os requisitos pré-determinados. Um caso de teste, ao ser executado, identifica um erro se a saída esperada for diferente da saída resultante da execução do software.
Classificacao	Classificam as características e comportamentos importantes dos ativos, por exemplo, domínio, autores, palavras-chave, etc. Contém grupos de descritores, que podem pertencer a contextos.
Codigo-fonte	Representa a implementação do projeto do software em alguma linguagem de programação.
Componente	Representa um conjunto de recursos que encapsula algum requisito e está pronto para ser reusado em outros softwares.
Contexto	O contexto esclarece onde um ativo é relevante. Por exemplo, um modelo de processo de negócios pertence ao contexto “Negócios” e também ao contexto “Documentação”. Isso indica que esse ativo é relevante para o contexto específico de negócios e também colabora para a documentação do sistema.
ContextoArtefato	Representa o contexto do ativo em que o artefato pode ser utilizado.
ContextoAtivo	Representa um contexto em que o ativo pode ser utilizado.
CriterioAceitacao	São os critérios definidos no plano de gestão para que um ativo seja aceito ou não pelo gerente de ativos, o qual é responsável pela gestão dos ativos.

Termo	Descrição Semântica
CriterioCertificacao	São os critérios definidos no plano de gestão para que o ativo seja certificado ou não pelo gerente de ativos, ou seja, garantem que o ativo esteja de acordo com a qualidade esperada.
CriterioDesativacao	São os critérios definidos no plano de gestão para que os ativos sejam descartados.
Dependencia	É um tipo de AtivoRelacionado; denota a ideia de que um ativo depende de outro ativo.
DependenciaArtefato	Um artefato pode depender de outros artefatos. Como exemplos de dependência entre artefatos têm-se: dependência em tempo de compilação, dependência em tempo de execução.
Descendencia	É um tipo de AtivoRelacionado; denota a ideia de que um dado ativo descende ou deriva de um outro ativo.
Descricao	Descreve o problema tratado pelo ativo e as suas principais estratégias de solução.
Descritor	Descreve os comportamentos e as características do ativo.
DocumentodeRequisito	É a informação das capacidades (requisitos) que o sistema tem que atender com o objetivo de satisfazer as necessidades dos seus usuários.
EstrategiaClassificacao	É um documento, criado pelo gerente de ativos, que compõe o plano de gestão e tem por objetivo definir a estratégia para a classificação dos ativos.
EstrategiaMetrica	É um documento que define as métricas que devem ser coletadas, registradas e contabilizadas sobre os ativos.
EstrategiaNotificacao	Definida no plano de gestão e determina quais notificações serão enviadas para os usuários do repositório sobre alterações que ocorrerem nos ativos.
EstrategiaPublicidade	É um documento que define as estratégias de publicidade, por exemplo, os ativos mais utilizados.
EstrategiaSolucao	Descreve a estratégia para a solução do problema resolvido pelos artefatos de um ativo.
FeedBack	Corresponde a um posicionamento ou avaliação dado para uma versão de um Ativo.
GrupoDescritor	Representa um agrupamento de descritores para ajudar a definir o contexto do ativo.
Mecanismo	É um conjunto de ações específicas para atingir um determinado objetivo do plano de gestão de ativos reutilizáveis.
MecanismoBusca	É o mecanismo que permite a busca e a recuperação dos ativos no repositório. Existem algumas formas para realizar as buscas, como por exemplo, por meio de palavras-chave, busca textual que pode ser realizada levando em consideração os elementos que compõem a especificação do ativo, ou mesmo por facetas que faz uso de uma combinação de pares (atributo, valor).
MecanismoControleAcesso	É um mecanismo que um repositório pode ter e que tem por objetivo executar a política de controle de acesso definida no plano de gestão.
MecanismoDependencia	É o mecanismo que permite o registro de dependência entre ativos. Essa dependência pode ser: agregação, similaridade, dependência ou origem.
MecanismoFeedback	É o mecanismo que permite o registro de impressões dos usuários a respeito da versão do ativo que eles utilizaram.
MecanismoGerenciaConfiguracao	É o recurso que permite a realização dos mecanismos de feedback e de versionamento com objetivo de executar a gerência de configuração do ativo.
MecanismoInclusao	É o mecanismo de armazenamento que permite a inclusão de ativos no repositório.
MecanismoMetrica	É o mecanismo do repositório que registra as métricas coletadas sobre a utilização do ativo. Atende a EstrategiaMetrica do plano de gestão de ativos reutilizáveis.

Termo	Descrição Semântica
MecanismoNavegacao	É o mecanismo que, a partir de algum grupo de descritores da classificação ou de algum método de agrupamento ou listagem simples, permite a navegação entre os ativos armazenados.
MecanismoNotificacao	É um dos mecanismos que um repositório pode ter. Visa atender uma estratégia de notificação (EstrategiaNotificacao) aos usuários sobre modificações ocorridas nos ativos, definida no plano de gestão. Atende a estratégia de notificação do plano de gestão de ativos reutilizáveis.
MecanismoPublicidade	É o mecanismo que oferece informações relativas ao reuso, iniciativas de reuso, ativos mais usados, etc. Atende a estratégia de publicidade (EstrategiaPublicidade) do plano de gestão de ativos reutilizáveis.
MecanismoVersionamento	É o mecanismo que permite a criação e o armazenamento de múltiplas versões de um mesmo ativo.
Modelo	É um artefato e refere-se a "uma abstração de alguma coisa, cujo propósito é permitir que se conheça essa coisa antes de construí-la" (BOOCH et al. 2006).
ModeloArquitetura	É um modelo que mostra a estrutura organizacional de um sistema, software ou componente de software, concebida durante a atividade de projeto.
ModeloInterfaceHumana	É um modelo que contém informações sobre a interação de software ou hardware com o ser humano.
ModeloProcessoNegocio	É um modelo que representa os processos de negócio de alguma organização. Esse modelo pode ser utilizado para apoiar o mapeamento de alguma solução de software para a organização.
ModeloProjeto (design)	É um modelo que captura o conhecimento e as decisões de projeto.
Modulo	É um artefato que consiste de um conjunto de arquivos, em geral bibliotecas únicas ou um executável, que contém instruções que representam alguma parte do software.
Papel	Responsabilidade que uma pessoa assume durante o uso de ativos no desenvolvimento de um software; sendo que uma pessoa pode assumir mais do que um papel.
Pessoa	Representa alguém que realiza alguma ação definida, por meio de um papel definido.
PlanoGestao	Criado pelo gerente de ativos e é onde se definem os recursos e os procedimentos que serão realizados para a gestão do ativo reutilizável. Esse plano deve ser executado pelo gerente de ativos e ser revisado levando em consideração o processo de revisão de software (ISO/IEC 12207, 2009). No plano de gestão estão descritas todas as informações (critérios, políticas e procedimentos) para a realização da gestão de ativos.
PoliticaControleAcesso	Determina o acesso aos mecanismos oferecidos pelo repositório, de acordo com o papel que o usuário assume.
PontoVariabilidade	Determinam os pontos (locais) nos artefatos onde pode ocorrer uma variabilidade, ou seja, onde pode ser feita uma customização para um contexto específico.
Problema	Descreve o problema que o ativo soluciona.
RegraCustomizacao	Representa as regras, restrições e orientações para que o consumidor do ativo possa personalizar o ponto de variabilidade do artefato. Isso é feito durante a execução de atividades do artefato.
RegraUso	Representa a informação sobre como aplicar ou usar um ativo ou artefato. Essas instruções podem ser para todo o ativo ou para um artefato específico do ativo; são informações que restringem e definem também como as atividades dos artefatos e dos ativos serão executadas.
Repositorio	Representa o conceito de um armazém de ativos, que é responsável por prover um conjunto de mecanismos para a gestão e uso dos ativos.

Termo	Descrição Semântica
Similaridade	É um tipo de AtivoRelacionado; denota a ideia de que um ativo possui características similar a outro ativo.
Solucao	Representa a solução fornecida pelo ativo, por meio da coleção de artefatos que o compõe. A solução é fornecida pelo(s) artefato(s) de acordo com o contexto do ativo que o artefato pertence. Existem alguns tipos de solução de acordo com a atividade que cria a solução: requisitos, análise, projeto, implementação e teste.
SolucaoAnalise	É uma solução criada durante uma execução de uma atividade de análise no desenvolvimento de software.
SolucaoImplementacao	É uma solução criada durante uma execução de uma atividade de implementação no desenvolvimento de software.
SolucaoProjeto	É uma solução criada durante uma execução de uma atividade de projeto no desenvolvimento de software.
SolucaoRequisito	É uma solução criada durante uma execução de uma atividade de requisito no desenvolvimento de software.
SolucaoTeste	É uma solução criada durante uma execução de uma atividade de teste no desenvolvimento de software.
SuporteMultiplasFontes	É a capacidade que um repositório pode ter para aceitar múltiplas fontes de origem de ativos, com objetivo de facilitar a integração entre equipes e entre repositórios diferentes.
TempoCompilacao	É um tipo de DependenciaArtefato que determina que um artefato depende de outro artefato em tempo de compilação.
TempoExecucao	É um tipo de DependenciaArtefato que determina que um artefato depende de outro artefato em tempo de execução.
Versao	Cada uma das formas sucessivas que um artefato de software possui, à medida que é desenvolvido ou evoluído.

Apêndice C Modelo Conceitual da ONTO-ResAsset

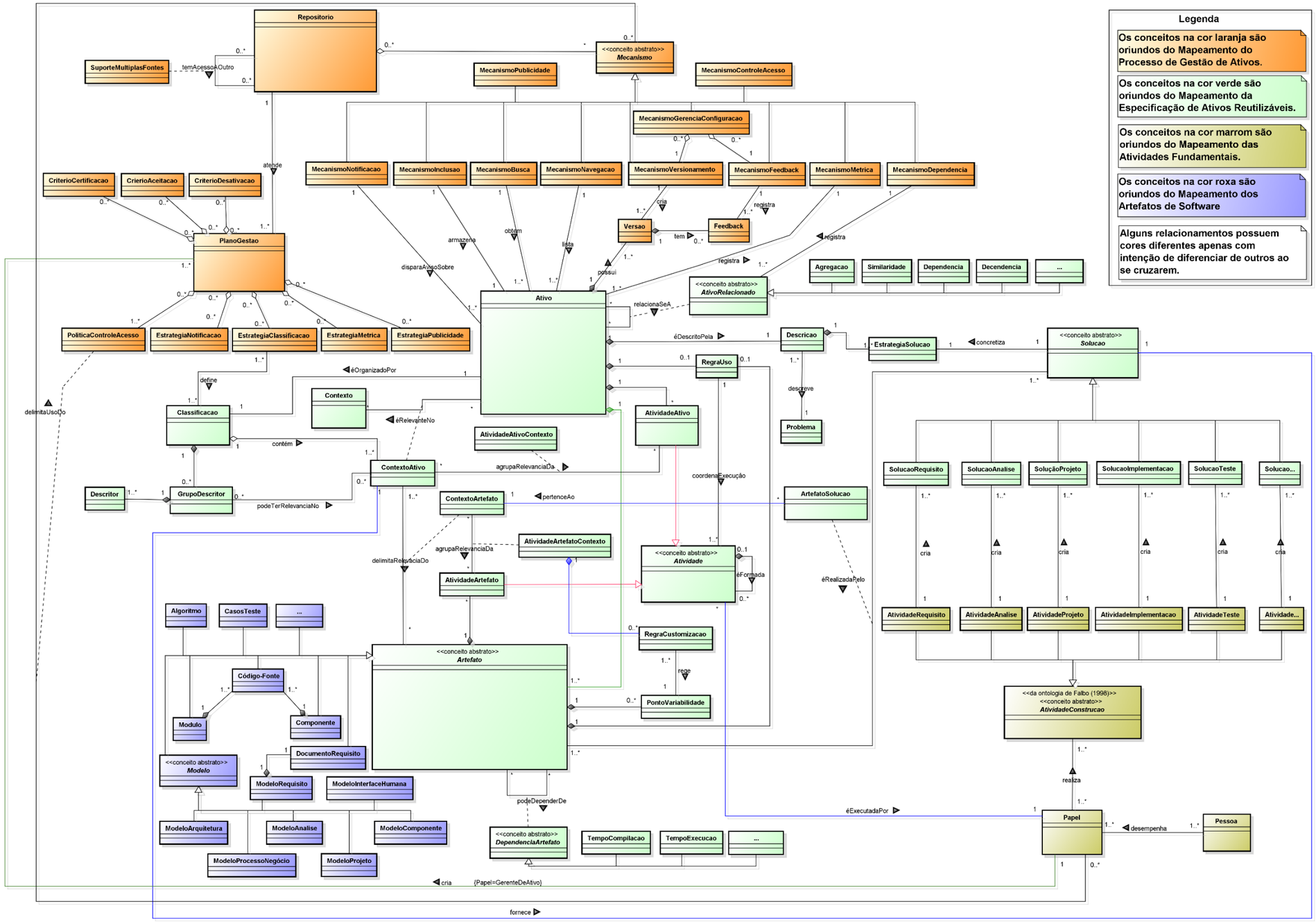


Figura C.1 - Modelo Conceitual da ONTO-ResAsset

Apêndice D Questionário para Avaliação da ONTO-ResAsset por Especialista no Domínio

Este apêndice apresenta o questionário para avaliação da ONTO-ResAsset por especialistas no domínio de reuso de software.

A ontologia ONTO-ResAsset é uma ontologia de ativos reutilizáveis durante as atividades fundamentais de desenvolvimento de software, ou seja, requisitos, análise, projeto (*design*), implementação e teste; tendo como escopo os artefatos que compõem os ativos, bem como a especificação e a gestão dos ativos.

A partir da leitura da lista de termos de conceitos, do entendimento da representação da ontologia no modelo conceitual e da seção de conceitualização responda as questões a seguir.

1) Existe algum conceito descrito incorretamente?

( ) Sim ( ) Não

Caso afirmativo, indique qual(is) e justifique:

2) Algum conceito importante do domínio não está representado na ontologia?

( ) Sim ( ) Não

Caso afirmativo, indique qual(is) conceito(s) deveria(m) estar representado(s) e justifique:

3) Existe algum conceito na ontologia que não faz parte do domínio?

( ) Sim ( ) Não

Caso afirmativo, indique qual(is) conceito(s) não faz(em) parte do domínio e justifique:

4) Existem relacionamentos ou restrições entre os conceitos que foram mapeados de maneira equivocada?

( ) Sim ( ) Não

Caso afirmativo, indique qual(is) relacionamentos ou restrições que foram mapeados de maneira equivocada e justifique:

5) Falta algum relacionamento ou restrição entre conceitos?
   ( ) Sim ( ) Não

Caso positivo, indique qual(is) relacionamento(s) ou restrição(ções) está(ão) faltando e justifique:

6) Por favor, forneça quaisquer outras observações que poderiam auxiliar na melhoria da ontologia de ativos reutilizáveis:

Apêndice E Questionário para Avaliação da ONTO-ResAsset por Não Especialista no Domínio

Este apêndice apresentam os questionários para avaliação da ONTO-ResAsset por não especialistas no domínio de reuso de software, porém que tenham formação na área de Computação ou experiência no desenvolvimento de software na Seção E.1 e no Seção E.2 uma cópia do questionário de perfil aplicado aos voluntários, realizado por meio de formulários do *Google Docs*[1].

E.1 Questionário avaliação da ONTO-ResAsset

A ONTO-ResAsset é uma ontologia de ativos reutilizáveis durante as atividades fundamentais de desenvolvimento de software, ou seja, requisitos, análise, projeto (*design*), implementação e teste; tendo como escopo os artefatos que compõem os ativos, bem como a especificação e a gestão dos ativos.

A partir da leitura dos glossários dos termos e conceitos e do entendimento da representação da ontologia no modelo conceitual, responda as questões a seguir indicando os conceitos e/ou relacionamentos da ontologia ONTO-ResAsset envolvidos.

1) Quais atividades fazem parte da construção de um software?
2) Quem realiza as atividades de construção?
3) Com qual informação é possível identificar quais artefatos foram criados por uma determinada atividade de construção?
4) Um ativo de software é composto por quais elementos?
5) Como é possível identificar qual problema um ativo propõe resolver?
6) Como é possível identificar quais são as estratégias de solução fornecidas por um ativo?
7) Quais os tipos de soluções os ativos podem prover?
8) Uma Solução resolve um Problema, quem realiza essa solução é um Artefato. Com que informação é possível identificar quais são os contextos de uma solução do artefato?

[1] Serviço disponível em: http://docs.google.com

9) Como é possível identificar quais são os contextos que uma Atividade do Artefato pode ser executada?
10) Como é possível identificar quais são os contextos em que uma atividade do Ativo possa ser executada?
11) Como é possível identificar quais são as regras de uso que coordenam a execução das Atividades de Ativo?
12) Como é possível identificar quais são as regras de uso que coordenam a execução das Atividades de Artefato?
13) Como é possível identificar os contextos do qual uma classificação organiza um ativo?
14) Como é possível identificar quais são os descritores que fazem parte de uma determinada Classificação de um Ativo?
15) Como é possível identificar quais são os descritores fazem parte de um determinado contexto?
16) Como é possível identificar quais são os pontos de variabilidade que podem ser personalizados em um artefato?
17) Como é possível identificar quais são as regras de customização que devem ser atendidas para personalizar um determinado ponto de variabilidade de um artefato?
18) Com qual informação a granularidade do ativo, que é determinada pelo número de artefatos que compõe um ativo, pode ser identificada?
19) Com qual informação a variabilidade do ativo, que é determinada pelo número de pontos de variabilidade que os artefatos possuem, pode ser identificada?
20) A articulação do ativo, é determinada pela quantidade de artefatos que o compõe que estejam mais próximos de uma solução de implementação, a partir de qual informação isso pode ser identificado?
21) Com qual informação é possível determinar como é feita a gestão de reuso de ativos de software?
22) Quais são todos os mecanismos que podem ser usados para apoiar a realização da gestão de reuso de ativos de software?
23) Quais elementos, relacionado diretamente a um Ativo de software, são necessários para dar suporte à algum mecanismo da gestão de reuso de ativos?
24) Com qual informação é possível determinar as versões de um ativo?
25) Com qual informação é possível identificar os *feedbacks* de uma determinada versão de um ativo?
26) Como as dependência entre ativos é identificada?

27) Quem cria o plano de gestão de ativos de software?

28) A partir de qual informação é possível determinar quem pode utilizar cada mecanismo de um repositório?

29) A ontologia ONTO-ResAsset possibilitou a você um melhor entendimento sobre a especificação e gestão de ativos?

( ) Sim ( ) Não ( ) Parcialmente

( ) Talvez ( ) Não Sei

30) A partir do conhecimento sobre especificação e gestão de ativos, obtido por meio da ONTO-ResAsset, você acredita que será mais fácil estabelecer critérios para selecionar repositórios de reuso para apoiar o desenvolvimento de software na empresa?

( ) Sim ( ) Não ( ) Parcialmente

( ) Talvez ( ) Não Sei

31) Se Se você fosse desenvolver um repositório de reúso, você acredita que a ontologia ONTO-ResAsset pode lhe ajudar na identificação dos requisitos e na definição da estrutura do repositório?

( ) Sim ( ) Não ( ) Parcialmente

( ) Talvez ( ) Não Sei

Utilize o espaço abaixo para fazer comentários adicionais sobre a ONTO-ResAsset.

E.2 Questionário de Perfil

Questionário de Perfil

Levantamento do perfil dos avaliadores da ontologia ONTO-ResAsset

*Obrigatório

1. **Nome ou Iniciais do Nome Completo:** *
   Caso queira utilize apenas suas iniciais.

   _____

2. **01. Qual seu nível de formação mais alto?** *
   Marcar apenas uma oval.

   - ◯ Médio
   - ◯ Técnico
   - ◯ Superior
   - ◯ Pós Graduado
   - ◯ Outro: _____

3. **02. Qual seu cargo atual na empresa?** *
   Ex: Analista, Design, Programador, Testador, Estagiário, etc.

   _____

4. **03. Quais papéis realiza atualmente no desenvolvimento de software?** *
   Marque todas que se aplicam.

   - ☐ Analista de Requisitos
   - ☐ Analista de Processos de Negócios
   - ☐ Analista de Sistemas
   - ☐ Desenvolvedor
   - ☐ Testador
   - ☐ Design
   - ☐ Outro: _____

5. **04. Quanto tempo você tem de experiência com desenvolvimento de software?** *
   Número de meses.

   _____

6. **05. Quanto tempo você tem de experiência na fase de Requisitos?** *
   Número de meses. Caso não tenha experiência, informe 0 (zero).

7. **06. Quanto tempo você tem de experiência na fase de Análise?** *
   Número de meses. Caso não tenha experiência, informe 0 (zero).

8. **07. Quanto tempo você tem de experiência na fase de Projeto?** *
   Número de meses. Caso não tenha experiência, informe 0 (zero).

9. **08. Quanto tempo você tem de experiência na fase de Implementação (codificação)?** *
   Número de meses. Caso não tenha experiência, informe 0 (zero).

10. **09. Quanto tempo você tem de experiência na fase de Teste?** *
    Número de meses. Caso não tenha experiência, informe 0 (zero).

11. **10. Qual é o seu nível de conhecimento na notação do diagrama de classes da UML?** *
    Marcar apenas uma oval.

    - ◯ Nenhum
    - ◯ Pouco
    - ◯ Razoável
    - ◯ Alto

12. **11. Qual é sua experiência prática no uso da notação do diagrama de classes da UML?** *
    Marcar apenas uma oval.

    - ◯ Nenhum
    - ◯ Estudado em aula ou a partir de um livro
    - ◯ Praticado em um projeto de classe
    - ◯ Usado em um projeto ou na indústria
    - ◯ Usado em vários projetos na indústria

13. **12. Realiza alguma prática de reuso?** *

Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 14.*
- ◯ Não

Caso tenha Respondido SIM na questão 12, referente à prática de reuso, responda as questões a seguir

Ir para a pergunta 14.

14. **13. Há quanto tempo realiza reuso?**

Tempo total em meses em que utilizou do reúso. Por exemplo, você trabalhou em um projeto X, que utilizou Frameworks como técnica de reuso e esse projeto teve duração de 12 meses. Ao final some o total de meses em que trabalhou em projetos que reusaram ativos.

_____

15. **14. Quais artefatos já reusou?**

Marque todas que se aplicam.

- ☐ Bibliotecas
- ☐ Casos de Teste
- ☐ Componentes
- ☐ Frameworks
- ☐ Padrões de Projeto
- ☐ Requisitos
- ☐ Padrões de Análise
- ☐ Trechos de Código
- ☐ Outro: _____

16. **15. De onde (local) foram obtidos os artefatos que você reutilizou?**

Marque todas que se aplicam.

- ☐ Google Code
- ☐ GitHub
- ☐ SouceForge
- ☐ Repositório Local (da empresa)
- ☐ Livro
- ☐ StackOverFlow
- ☐ Tutoriais
- ☐ Internet em geral (excluindo-se os itens acima)
- ☐ Outro: _____

Apêndice F Questionário para Avaliação da ONTO-ResAsset por Não Especialista no Domínio (Refinado grupo 2)

A ONTO-ResAsset é uma ontologia de especificação e gestão de ativos reutilizáveis durante as atividades fundamentais de desenvolvimento de software, ou seja, requisitos, análise, projeto (*design*), implementação e teste.

A partir do glossário de conceitos e do entendimento da representação da ontologia no modelo conceitual, responda as questões a seguir indicando o(s) **conceito(s)**, **relacionamento(s)** e **multiplicidade(s)** da ontologia ONTO-ResAsset envolvido(s) para responder cada questão.

Nome ou iniciais do nome completo:

_____

Questões:

7) Quais atividades fazem parte da construção de um software?

8) Quem realiza as atividades de construção?

9) Com qual informação é possível identificar quais artefatos foram criados por uma determinada atividade de construção?

10) Um ativo de software é composto por quais elementos?

11) Como é possível identificar qual problema um ativo propõe resolver?

12) Como é possível identificar quais são as estratégias de solução fornecidas por um ativo?

13) Os ativos podem prover quais tipos de soluções?

14) Uma solução resolve um problema e quem realiza essa solução é um ou mais artefatos. Com que informação é possível identificar quais são os contextos de uma solução do artefato?

15) Como é possível identificar quais são os contextos que uma atividade do artefato pode ser executada?

16) Como é possível identificar quais são os contextos em que uma atividade do ativo pode ser executada?

17) Como é possível identificar quais são as regras de uso que coordenam a execução das atividades de um ativo?

18) Como é possível identificar quais são as regras de uso que coordenam a execução das atividades de um artefato?

19) Como é possível identificar os contextos da classificação de um ativo?

20) Como é possível identificar quais são os descritores que fazem parte de uma determinada classificação de um ativo?

21) Como é possível identificar quais são os descritores que fazem parte de um determinado contexto?

22) Como é possível identificar quais são os pontos de variabilidade que podem ser personalizados em um artefato?

23) Como é possível identificar quais são as regras de customização que devem ser atendidas para personalizar um determinado ponto de variabilidade de um artefato?

24) Como é possível identificar a quantidade de problemas que um ativo propõe resolver (o qual determina a granularidade do ativo)?

25) Como é possível identificar o número de pontos de variabilidade que os artefatos possuem (o qual determina a variabilidade do ativo)?

26) Como é possível identificar a capacidade de articulação de um ativo? Quanto mais completo for o ativo para solucionar o problema, maior é sua articulação. Por exemplo, um ativo que provê os requisitos, casos de testes e a implementação da solução, é mais articulado do que se contivesse apenas os requisitos.

27) Com qual informação é possível determinar como é feita a gestão de reuso de ativos de software?

28) Quais são os mecanismos que podem ser usados para apoiar a realização da gestão de reuso de ativos de software?

29) Quais elementos, relacionados diretamente a um ativo de software, são necessários para dar suporte aos mecanismos de gestão de reuso de ativos?

30) Com qual informação é possível determinar as versões de um ativo?

31) Com qual informação é possível identificar os *feedbacks* de uma determinada versão de um ativo?

32) Como as dependências entre ativos é identificada?

33) Quem cria o plano de gestão de ativos de software?

34) A partir de qual informação é possível determinar quem pode utilizar cada mecanismo de um repositório?

35) A ontologia ONTO-ResAsset possibilitou a você um melhor entendimento sobre a especificação e gestão de ativos?

( ) Sim
( ) Não
( ) Parcialmente
( ) Talvez
( ) Não sei

36) A partir do conhecimento sobre especificação e gestão de ativos, obtido por meio da ONTO-ResAsset, você acredita que será mais fácil estabelecer critérios para selecionar repositórios de reuso para apoiar o desenvolvimento de software em uma empresa?

( ) Sim
( ) Não
( ) Parcialmente
( ) Talvez
( ) Não sei

37) Se você fosse desenvolver um repositório de reúso, você acredita que a ontologia ONTO-ResAsset pode lhe ajudar na identificação dos requisitos e na definição da estrutura do repositório?

( ) Sim
( ) Não
( ) Parcialmente
( ) Talvez
( ) Não sei

Utilize o espaço abaixo para fazer comentários adicionais sobre a ONTO-ResAsset.