Uma Ontologia de Requisitos de Software

Julio Cesar Nardi, Ricardo de Almeida Falbo

Mestrado em Informática, Universidade Federal do Espírito Santo, Vitória - ES - Brasil
julionardi@yahoo.com.br, falbo@inf.ufes.br

Abstract. Tool integration in Software Development Environments (SDEs) is a complex problem. A way to improve tool integration is building them sharing a common conceptualization, given by ontologies. This paper presents a software requirements ontology developed to serve as basis for the development of tools to support the requirements engineering process in the SDE ODE (Ontology-based software Development Environment). Since in the context of ODE's Project several software engineering domain ontologies were developed, the software requirements ontology was built integrated to them, reusing conceptualizations previously established.

1 Introdução

Obter os requisitos corretos em um projeto de software é a parte mais importante e difícil do processo de software. A deficiência no tratamento de requisitos tem sido apontada como a principal causa de fracassos de projetos de software [1]. Neste contexto, o processo de Engenharia de Requisitos (ER) ganha importância, assim como o apoio automatizado a ele.

Segundo Kotonya e Sommerville [2], o processo da ER é sistemático e abrange as atividades de elicitação, análise e negociação, documentação, validação e gerência de requisitos. Para os engenheiros de requisitos, que buscam especificar requisitos dentro de prazos e orçamentos definidos, a aplicação de métodos e ferramentas é considerada crucial para se obter sucesso. Entretanto, muitas vezes, ferramentas interferem na realização das atividades da ER ao invés de apoiá-las [1]. Acreditamos que, para serem efetivas, tais ferramentas têm de ser construídas apoiadas em um entendimento de consenso acerca dos elementos envolvidos no processo de ER.

Considerando que os requisitos são o foco do processo da ER, é interessante que se tenha uma compreensão clara do que vem a ser um requisito e de como ele se relaciona com outros elementos do processo de software. Como fruto dessa compreensão, é possível construir ferramentas de apoio ao processo de ER que sejam mais amplamente utilizáveis, já que essas estão baseadas em definições compartilhadas do domínio de requisitos.

Como forma de estabelecer uma conceituação básica acerca de requisitos, desenvolvemos uma ontologia de requisitos de software a ser usada como base para a construção de ferramentas de apoio ao processo de ER no ambiente de desenvolvimento de software (ADS) ODE (*Ontology-based software Development Environment*) [3]. Como o nome indica, ODE tem sua fundamentação baseada em ontologias, partindo do pressuposto que, se as ferramentas de um ADS são

construídas baseadas em ontologias, a integração das mesmas é facilitada, pois os conceitos envolvidos são bem definidos. Ontologias são usadas em ODE, dentre outros, para estruturar o ambiente e sua infra-estrutura de gerência de conhecimento e para estabelecer uma forma padrão de comunicação entre agentes (humanos e de software) no ambiente [3]. Ontologias servem a esse propósito, na medida em que elas definem um vocabulário de representação, que captura os conceitos e as relações de um domínio, e um conjunto de axiomas, que restringem a sua interpretação [4].

Este trabalho apresenta a Ontologia de Requisitos de Software desenvolvida visando à formalização do conhecimento acerca do domínio de requisitos, sobretudo, para apoiar o desenvolvimento de ferramentas para o ambiente ODE. Essa ontologia foi definida buscando seguir o critério de compromissos ontológicos mínimos, de modo que seja possível reutilizar esse conhecimento de várias formas no ambiente, tais como na comunicação entre pessoas e o ambiente, entre agentes de software atuando no ambiente, na construção e integração de ferramentas, no aprendizado de Engenharia de Requisitos dentro do ambiente etc. Além disso, uma vez que diversas ontologias já foram construídas no contexto do projeto ODE, procurou-se sempre reutilizar conceituações previamente estabelecidas, favorecendo a formação de uma rede efetiva de conceitos e relações que possa ser explorada de várias formas.

Este trabalho está organizado da seguinte forma: a seção 2 discute brevemente alguns aspectos da Engenharia de Requisitos e de ontologias, incluindo o método de construção de ontologias seguido neste trabalho; a seção 3 apresenta a Ontologia de Requisitos desenvolvida; a seção 4 trata de trabalhos correlatos; e na seção 5 são tecidas algumas considerações finais.

2 Requisitos de Software e Ontologias

A Engenharia de Requisitos (ER) é a subárea da Engenharia de Software que trata do processo de definição dos requisitos de software. Esse processo é sistemático e abrange diversas atividades, tais como elicitação, análise e negociação, documentação, validação e gerência de requisitos [2].

Requisitos de software são sentenças que expressam as necessidades dos clientes e que condicionam a qualidade do software [5], ou especificações de serviços que o sistema deve prover, restrições no sistema e conhecimentos necessários para desenvolvê-lo [2]. Uma vez capturados, requisitos de software devem ser modelados, documentados, validados e acompanhados. Nesse processo, as propriedades de um requisito e os relacionamentos com outros elementos do processo de software são definidos e alterados. Portanto, definir e entender as propriedades e relações em torno de um requisito é essencial na condução do processo de ER.

Requisitos podem ser classificados segundo alguma forma de categorização, que pode ser definida de acordo com as práticas de cada organização de software. Geralmente, eles são classificados em: funcionais (representam o que o sistema deve fazer, suas funções, podendo ser subdivididos em essenciais, desejáveis e supérfluos) e não funcionais (representam os atributos do sistema enquanto software constituído, o que inclui manutenibilidade, eficiência etc).

Acerca de um requisito é interessante saber, ainda, a sua origem, os interessados (*stakeholders*) e os responsáveis por ele. De posse desses elementos, a atividade de análise e negociação é facilitada, pois é possível identificar os envolvidos e também o ponto de partida de um requisito [2].

No centro da atividade de gerência de requisitos – além do controle de alteração e de versão de um requisito e do acompanhamento de seu estado – está a rastreabilidade, isto é, a habilidade de se acompanhar a vida de um requisito em ambas as direções do processo de software e durante todo o ciclo de vida. A rastreabilidade de requisitos só é possível se houver ligações explícitas entre requisitos e outros elementos do processo de software. Dessa forma, a identificação da composição de requisitos, das dependências entre requisitos, de requisitos conflitantes, da origem dos requisitos e de seus interessados, além da identificação de em qual artefato (documento, diagrama, etc.) produzido durante o processo de software um requisito é tratado, é de fundamental importância para que a rastreabilidade possa ser implementada [2] [6].

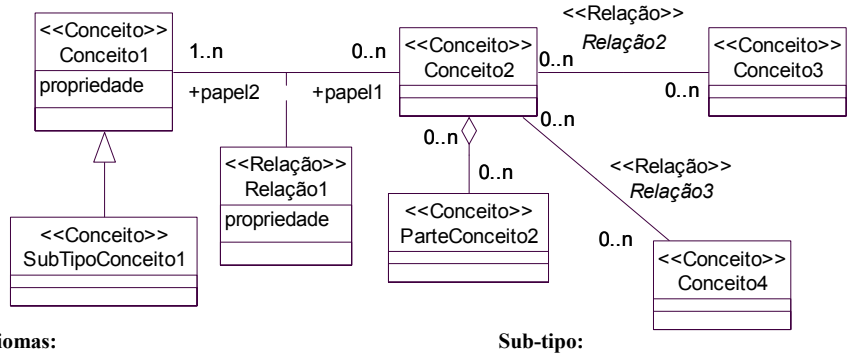
Os requisitos elicitados e analisados devem ser documentados. Na atividade de documentação, um Documento de Especificação de Requisitos de Software (DERS) é normalmente gerado, contendo os requisitos de software de um determinado projeto. Existem vários modelos de DERS e cada organização pode definir seu próprio modelo de acordo com suas necessidades. Qualquer que seja seu formato, o DERS é um dos artefatos mais importantes do processo de software, uma vez que é a base para praticamente todas as atividades de construção subseqüentes. Assim, é imprescindível avaliar a qualidade de um DERS. Para tal, características de qualidade devem ser apontadas e métricas devem ser estabelecidas para avaliar a qualidade do DERS.

Pela breve exposição acima, é possível notar o quanto é complexo o processo da ER. Dada essa complexidade, construir ferramentas de apoio às atividades desse processo é fundamental para a efetiva execução do mesmo. Idealmente, para serem amplamente utilizáveis, tais ferramentas devem ser construídas com base em modelos sólidos, consensuais. Neste contexto, ontologias são potencialmente úteis.

Uma ontologia define um vocabulário específico usado para descrever uma certa realidade, mais um conjunto de decisões explícitas, fixando de forma rigorosa o significado pretendido para o vocabulário. Uma ontologia envolve, então, um vocabulário de representação, que captura os conceitos, relações e suas propriedades em algum domínio, e um conjunto de axiomas, que restringem a sua interpretação [4]. Ontologias têm se tornado populares, em grande parte, pelo fato de terem como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre pessoas e sistemas de aplicação.

Como qualquer artefato de engenharia de software, ontologias têm de ser construídas seguindo métodos e técnicas adequados. Para desenvolver a ontologia de requisitos de software, foi adotado o método SABiO (***S****ystematic* ***A****pproach for* ***B****uilding* ***O****ntologies*) [7] [8]. SABiO define um processo para construção de ontologias, cujas principais atividades são: (i) identificação do propósito e especificação de requisitos, que visa identificar questões que a ontologia deve ser capaz de responder (questões de competência), (ii) captura da ontologia, que tem por objetivo capturar os conceitos, relações, propriedades e restrições relevantes sobre o domínio em questão; e (iii) formalização, que busca escrever os axiomas da ontologia em uma linguagem formal (neste trabalho, optou-se pela lógica de primeira ordem).

Paralelamente a todas essas atividades, ocorrem as atividades de: (iv) integração com ontologias existentes, que visa a reutilizar conceituações existentes e integrar a ontologia em desenvolvimento a uma rede de ontologias mais ampla; (v) avaliação da ontologia, que, dentre outros, trata de avaliar se a ontologia é capaz de responder às questões de competência; e (vi) documentação da ontologia, que visa a registrar o desenvolvimento da ontologia. Em sua versão mais recente [8], SABiO advoga o uso de uma extensão da UML como linguagem gráfica para representação de ontologias. Essa extensão propõe o uso de um subconjunto de elementos da UML exercendo o mesmo papel da notação de LINGO [7], a linguagem originalmente proposta. LINGO possuía primitivas para representar conceitos, relações e propriedades, e alguns tipos de relações que possuíam uma semântica bem-estabelecida, tais como relações todo-parte e sub-tipo-de, para os quais um conjunto de axiomas formais, ditos axiomas epistemológicos, era definido. Assim, apesar de se utilizar os elementos de modelo da UML, a semântica imposta é a mesma que a estabelecida para os correspondentes elementos em LINGO, conforme parcialmente mostrado na figura 1. Segundo essa extensão da UML, classes com estereótipo `<<Conceito>>` representam conceitos da ontologia. Relações são definidas como associações nomeadas. Propriedades de conceitos e relações são representadas como atributos. Relações que contêm propriedades ou que possuem aridade maior que dois são representados como classes associativas com estereótipo `<<Relação>>`. Relações de supertipo e todo-parte são representadas como relações de generalização / especialização e de agregação / composição, respectivamente. Finalmente, condicionantes entre relações são representados por restrições entre associações [9].



Axiomas:
Todo-Parte
(AE1) $\forall x \ \neg parte(x,x)$
(AE2) $\forall x,y \ parte(y,x) \leftrightarrow todo(x,y)$
(AE3) $\forall x,y \ parte(y,x) \rightarrow \neg \ parte(x,y)$
(AE4) $\forall x,y,z \ parte(z,y) \wedge parte(y,x) \rightarrow parte(z,x)$
(AE5) $\forall x,y \ disjunto(x,y) \rightarrow \neg\exists z \ parte(z,x) \wedge parte(z,y)$
(AE6) $\forall x \ atomico(x) \rightarrow \neg\exists y \ parte(y,x)$

Sub-tipo:
(AE7) $(\forall x,y,z)$ $(subtipo(x,y) \wedge subtipo(y,z) \rightarrow supertipo(x,z))$
(AE8) $(\forall x,y)$ $(subtipo(x,y) \rightarrow supertipo(y,x))$
Condicionante Ou-exclusivo:
(AE9) $(\forall a \in C2)$ $((\exists b)$ $(b \in C3) \wedge R2(a,b)) \rightarrow \neg((\exists c \in C4) \wedge R3(a,c)))$
(AE10) $(\forall a \in A)((\exists c)$ $(c \in C4) \wedge R3(a,c)) \rightarrow \neg((\exists b \in C3) \wedge R2(a,b)))$

Fig. 1. As principais notações da extensão da UML utilizada e os axiomas associados a ela.

3 Uma Ontologia de Requisitos de Software

Nesta seção é apresentada a Ontologia de Requisitos de Software proposta, tomando por base os artefatos definidos pelo método SABiO. Assim, começamos pela especificação de requisitos, listando as questões de competência levantadas:

QC1. O que é um requisito?
QC2. Qual a natureza de um requisito?
QC3. Quais são os requisitos de um projeto de software?
QC4. Para que módulos do sistema um requisito é alocado?
QC5. Quais os responsáveis por um requisito?
QC6. Quais os interessados (*stakeholders*) em um requisito?
QC7. Qual a origem de um requisito?
QC8. Qual o estado de um requisito?
QC9. Como um determinado requisito é decomposto em outros requisitos?
QC10. Que requisitos são dependentes de um determinado requisito?
QC11. Que requisitos são conflitantes entre si?
QC12. O que é uma especificação de requisitos de software (ERS)?
QC13. Que artefatos descrevem, modelam ou implementam um requisito?
QC14. Como gerenciar mudanças nos requisitos e nos artefatos a eles relacionados?
QC15. Como avaliar a qualidade em requisitos?

Uma vez que requisitos estão no universo da Engenharia de Software, é útil examinar outras ontologias nesse domínio. Neste trabalho foram inspecionadas as seguintes ontologias, construídas de forma integrada no âmbito do Projeto ODE:

- **Ontologia de Processo de Software:** desenvolvida em [7] e recentemente revisada em [10], essa ontologia é composta de ontologias de Atividade, Recurso e Procedimento e trata da conceituação básica acerca do domínio de processos de software, abordando conceitos de atividade, processo, projeto, artefato e recurso.
- **Ontologia de Artefatos** desenvolvida em [11], tem o intuito de prover um entendimento compartilhado acerca de alguns tipos de artefatos (documento, diagrama e código-fonte), definindo sub-ontologias para eles.
- **Ontologia de Gerência de Configuração de Software:** desenvolvida em [11], busca tratar de conceitos relacionados à gerência de configuração de software.
- **Ontologia de Qualidade de Software:** desenvolvida em [12], visa à conceituação sobre o domínio de qualidade de software, tratando de questões como avaliação da qualidade, características de qualidade e métricas.
- **Ontologia de Organizações de Software:** tem como propósito fornecer um vocabulário comum que possa ser utilizado para representar conhecimento útil sobre organizações de software, incluindo competências de seus membros.

A figura 2 mostra as ontologias integradas, destacando as dependências entre elas.

Da Ontologia de Processo, foram reutilizados os conceitos de *Projeto*, *Atividade* e *Artefato*. Requisitos são definidos no âmbito de um *Projeto* e, portanto, é importante relacionar os conceitos de *Requisito* e *Projeto*. Já o conceito de *Atividade* foi utilizado para compor o contexto no qual um requisito é originado, uma vez que um requisito pode ter origem em uma atividade do processo. Por fim, o conceito de *Artefato* é importante para a rastreabilidade, capturando em que artefatos um requisito é tratado.

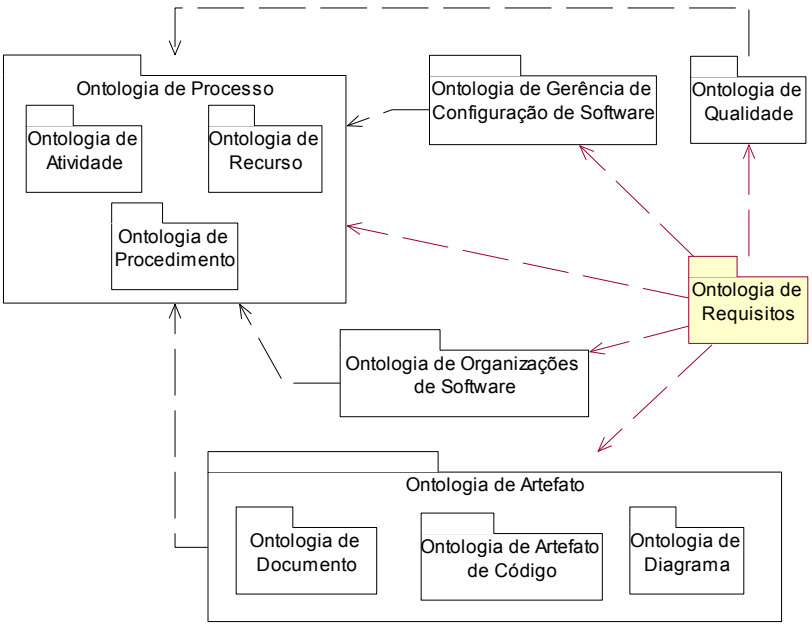


Fig. 2. A ontologia de requisitos e suas dependências com as ontologias utilizadas.

É importante ressaltar que aspectos relacionados ao processo de ER em si, tais como, atividades, sub-atividades, procedimentos etc, não foram tratados na Ontologia de Requisitos, pois já são tratados pela Ontologia de Processo. De fato, a descrição de um processo de ER pode ser feita pela instanciação da ontologia de processo.

A Ontologia de Artefato e suas sub-ontologias definem conceitos acerca de alguns tipos de artefatos e suas estruturas. Para este trabalho, foi especialmente importante a Ontologia de Documento, dado que um Documento de Especificação de Requisitos de Software (DERS) é um *Documento*. Assim, toda a conceituação usada para descrever artefatos e, mais especificamente, documentos, foi reutilizada para tratar DERSs, incluindo a definição de modelos de documento organizacionais para documentos [11].

Deve-se ressaltar que o conceito de *Artefato* é definido na Ontologia de Processo e não na Ontologia de Artefato. Isso porque a Ontologia de Artefato se propõe a conceituar alguns tipos de artefatos e não o que é um artefato. O conceito de artefato é tratado pela Ontologia de Processo, sendo definido como um produto ou insumo para atividades do processo de software [7].

Uma vez que é fundamental avaliar a qualidade em requisitos, a Ontologia de Qualidade de Software foi também utilizada, sobretudo no que se refere à qualidade de artefatos de software aplicada a DERS. Essa conceituação inclui *Características de Qualidade* diretamente e indiretamente mensuráveis, *Métricas, Medidas* etc [12].

A Ontologia de Gerência de Configuração de Software trata dos conceitos relacionados ao controle de versão e ao controle de alteração de itens de configuração, esses representando elementos do processo de software que estão sob gerência de configuração [12]. Como DERSs e requisitos devem ter sua configuração gerenciada, essa ontologia foi também integrada à ontologia de requisitos.

Da Ontologia de Organizações de Software reutilizou-se os conceitos *Pessoa e Equipe*. O primeiro representa os indivíduos de uma organização. Esse conceito é útil para se definir os interessados (*stakeholders*) e os responsáveis por um requisito. O segundo conceito, *Equipe*, define as equipes dos projetos de uma organização.

Por meio da reutilização de ontologias, associando conceitos dessas ontologias a conceitos definidos para a ontologia de requisitos, foi possível responder as questões de competência anteriormente colocadas, como veremos a seguir.

3.1 Captura e Formalização da Ontologia de Requisitos

Analisando as questões de competência anteriormente relacionadas, identificamos alguns aspectos relevantes, discutidos na seqüência:

1. Definição e Taxonomia de Requisitos (questões 1 a 4 e 12)
2. Aprovação e Interesse em Requisitos (questões 5 e 6)
3. Gerência de Requisitos (questões 7 a 11, 13 e 14)
4. Qualidade em Requisitos (questão 15)

Definição e Taxonomia de Requisitos

De maneira geral, requisitos são sentenças que descrevem serviços que um sistema deve prover, restrições que ele deve obedecer e características que ele deve possuir (*QC1*). Além disso, requisitos são definidos no âmbito de um projeto (*QC3*).

Conforme discutido na seção anterior, um DERS é um *Artefato*. Mais especificamente, um DERS é um *Documento* (*QC12*), que, segundo [11], é um artefato de software não passível de execução, constituído tipicamente de declarações textuais, normalmente associado a padrões organizacionais (roteiros e modelos de documento) que definem a forma como ele deve ser produzido.

Ainda que requisitos sejam definidos no âmbito de um projeto, é importante definir também para que módulos do projeto tal requisito é alocado (*QC4*). Para responder essa questão foram definidos os conceitos de *Escopo* e *Módulo*. Um projeto de software tem seu escopo normalmente dividido em módulos e, estabelecendo uma relação entre *Requisito* e *Módulo*, é possível acompanhar a que módulos um requisito está alocado e vice-versa.

Como podem existir vários tipos de requisito, se faz necessário criar uma taxonomia para organizá-los. Entendemos que não há um consenso acerca dos tipos de requisitos. Desse modo, optou-se por criar o conceito *Tipo de Requisito*, que admite sub-tipos, sendo que um subtipo pode ter vários supertipos e, para cada supertipo, é possível criar vários subtipos (*QC2*).

A figura 3 mostra a parte do modelo da ontologia referente às questões de competência referenciadas acima. Vale lembrar que a notação utilizada impõe implicitamente alguns axiomas. Por exemplo, os axiomas (AE1) a (AE6) da figura 1 se aplicam a todas as relações todo-parte mostradas na figura 2 (entre Escopo e Módulo, entre Módulos e entre Tipos de Requisitos), assim como os axiomas (AE7) e (AE8) da figura 1 se aplicam às relações de sub-tipo entre Artefato, Documento e Documento de Especificação de Requisitos de Software. Além desses axiomas, ditos epistemológicos, outros axiomas foram definidos, alguns deles discutidos na seqüência.

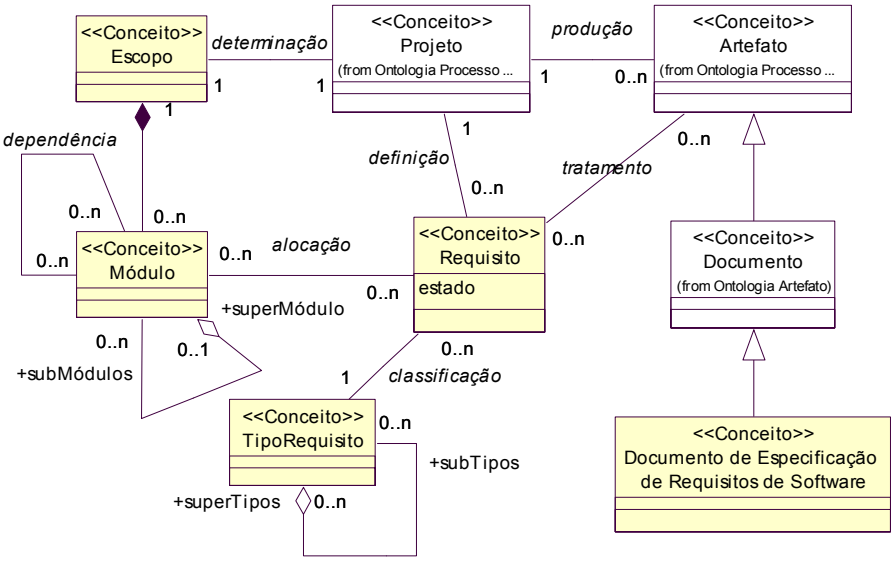


Fig. 3. Definição e Taxonomia de Requisito

Para formalizar aspectos importantes na alocação de requisitos a módulos, foram utilizados os seguintes predicados: *alocacao(r, m)*, indicando que o requisito *r* é alocado ao módulo *m*; e *submodulo(m1, m2)*, indicando que *m1* é um sub-módulo de *m2*. Esses predicados estão relacionados segundo a seguinte sentença, apontando que os requisitos alocados a um módulo estão alocados também a seus super-módulos:

$$(\forall r, m1, m2)\ (submodulo(m1,m2) \wedge alocacao(r, m1) \rightarrow alocacao(r, m2))\ (A1)$$

Além disso, é válido o axioma abaixo que define que um requisito só pode estar alocado a um módulo que esteja no escopo do projeto que definiu esse requisito. Nesse axioma o predicado *definição(r, p)* indica que o requisito *r* foi definido no projeto *p; composição(m, e)* indica que o módulo *m* compõe o escopo *e*; e *determinação(e,p)* indica que o escopo *e* foi determinado para o projeto *p*.

$$(\forall\ p,\ r,\ m)\ (alocação(r,\ m) \wedge definição(r,\ p) \wedge composição(m,\ e) \rightarrow determinação\ (e,\ p)\ )\quad (AC1)$$

Deve-se ressaltar a diferença entre os axiomas (A1) e (AC1). O axioma (A1) deriva uma nova informação (e, portanto, é dito um axioma ontológico), enquanto o axioma (AC1) não deriva nova informação, mas apenas restringe o estabelecimento de relações. Axiomas desse tipo são ditos axiomas de consolidação [7].

Dada a limitação de espaço, neste artigo não apresentamos outros axiomas definidos. Apenas para registrar, a relação de dependência entre módulos é transitiva e simétrica. Além disso, há um axioma de consolidação envolvendo as relações entre Requisito, Projeto e Artefato.

Além de modelos gráficos e axiomas formais, SABiO propõe que seja elaborado um dicionário de termos com uma entrada para cada termo definido na ontologia (conceitos, relações e propriedades). Novamente por limitações de espaço, neste

artigo apresentamos apenas os termos referentes aos conceitos definidos na ontologia de requisitos, mostrados na Tabela 1. Os termos em negrito nessa tabela correspondem a outros termos descritos no dicionário de termos completo.

Tabela 1. Parte do Dicionário de Termos da Ontologia de Requisitos.

Requisito	Representa especificações de serviços que o sistema deve prover, restrições do sistema e do processo de desenvolvimento e conhecimento necessário para a construção do sistema.
Documento de Especificação de Requisitos de Software	É um **documento**, portanto um **artefato** de software, formal que é utilizado para comunicação dos requisitos aos clientes, aos usuários etc, servindo de base para diversas **atividades** do **processo** de software. Sendo um documento, é normalmente associado a padrões organizacionais (**roteiros**) que definem a forma como deve ser produzido.
Tipo de Requisito	Conceito utilizado na caracterização de um requisito para retratar a dimensão que o requisito procura abordar, tal como funcional, não-funcional etc.
Módulo	É uma porção de sistema utilizada para organizar e agrupar, por exemplo, **artefatos**, **requisitos** e outros módulos que possuem certa afinidade.
Escopo	Determina os limites de um **projeto**, englobando o que faz parte do **projeto** e o que não faz. Um escopo pode ser decomposto em **módulos**.
Contexto	Caracteriza qual a situação na qual um **requisito** surgiu, procurando descrever quais as **pessoas** envolvidas, quais os **artefatos** analisados e em qual **atividade** do **processo** de software o requisito surgiu.

Aprovação e Interesse em Requisitos

O processo de Engenharia de Requisitos envolve várias pessoas de diversas áreas e com perspectivas diferentes. Dessa forma, é interessante que se saiba os interessados em cada requisito, com o intuito de facilitar a localização dos mesmos para facilitar negociações, esclarecimentos e discutir impactos de mudanças (*QC6*).

Além disso, é imprescindível que existam pessoas responsáveis por um dado requisito. Tais pessoas podem, por exemplo, aprovar um requisito antes que ele seja tratado por atividades subseqüentes do processo de desenvolvimento (*QC5*).

A figura 4 mostra a parte da ontologia de requisitos que trata dessas questões. O conceito *Requisito* se relaciona com o conceito *Pessoa*, da Ontologia de Organizações de Software, por meio das relações *interesse* e *responsabilidade*. Como os nomes sugerem, a primeira relação representa o interesse de pessoas em um requisito e a segunda representa os responsáveis por um requisito.

Dentre os axiomas definidos para essa parte da ontologia destacam-se os seguintes axiomas de consolidação:

$(\forall\ r, p, e, pr)\ (interesse(p, r) \rightarrow participação(p, e) \wedge alocação\_pessoal(e, pr) \wedge definção(r, pr))$ *(AC2)*

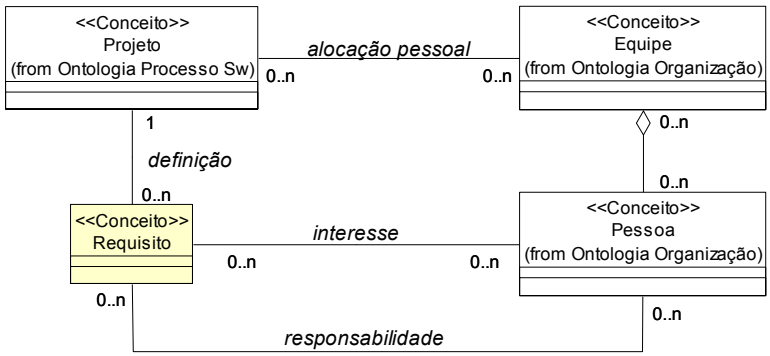


Fig. 4. Aprovação e Interesse em Requisito.

$(\forall\ r,\ p,\ e,\ pr)\ (reponsabilidade(p,\ r) \rightarrow participação(p,\ e) \wedge alocação\_pessoal(e,\ pr) \wedge definção\ (r,\ pr))$ *(AC3)*

O axioma (AC2) garante que, para uma pessoa ser um interessado em um requisito, ela deve estar alocada a uma equipe do projeto no qual o requisito foi definido. Já o axioma (AC3) garante que, para uma pessoa ser responsável por um requisito, ela também deve estar alocada a uma equipe do projeto que definiu o requisito. Nesses axiomas, os seguintes predicados foram utilizados: *interesse(p, r)* indica que a pessoa *p* tem interesse no requisito *r*; *responsabilidade(p, r)* indica que a pessoa *p* é responsável pelo requisito *r*; *participação(p, e)* indica que a pessoa *p* participa da equipe *e*; e *alocação_pessoal(e, pr)* indica que a equipe *e* está alocada ao projeto *pr*.

Gerência de Requisitos

Segundo [6], a gerência de requisitos compreende controle de mudança, controle de versão, acompanhamento do estado dos requisitos e rastreabilidade.

No que tange ao controle de mudança e ao controle de versão de requisito, pode-se perceber que estamos falando de gerência da configuração de requisitos (e documentos de especificação de requisitos de software – DERS). Como destacado na seção anterior, a Ontologia de Gerência de Configuração de Software trata dessas questões. O conceito central dessa ontologia é o conceito de *Item de Configuração*, o qual representa algo que está sob gerência de configuração e, assim, só pode ser alterado segundo um procedimento de controle de alteração formalmente estabelecido e documentado. Um *Item de Configuração* possui várias *Variações*, que podem ser tanto do tipo *Variante* quanto *Versão*. Assim, para que requisitos e DERS sejam submetidos à gerência de configuração, basta relacioná-los com *Item de Configuração*, por meio da relação *derivação*, conforme definido em [11] e parcialmente mostrado na figura 5.

Ainda em relação ao controle de mudanças e envolvendo também a rastreabilidade, é necessário estabelecer uma rede de ligações com o objetivo de detectar com mais facilidade o impacto de mudanças. Essas ligações, normalmente, buscam fornecer informações acerca de dependência (*QC10*), composição (*QC9*) e conflito (*CQ11*) entre requisitos, informações sobre onde um requisito está descrito, modelado e codificado (*QC13*) e qual a origem de um requisito (*QC7*). O modelo da figura 6 procura tratar esses aspectos.

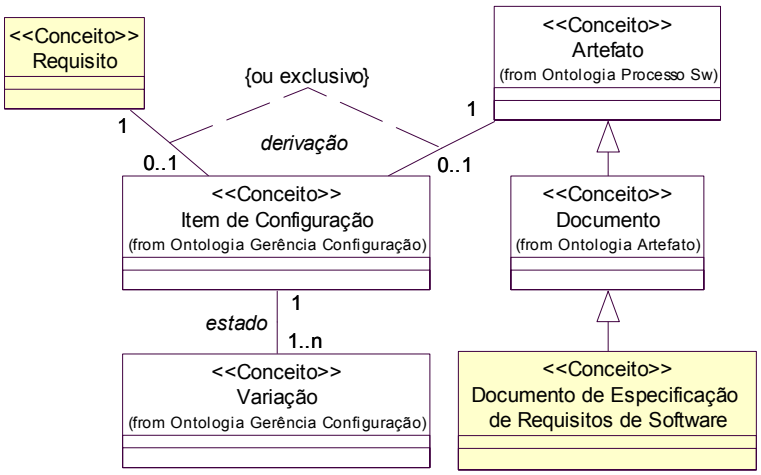


Fig. 5. Gerência de Configuração de Software aplicada a Requisitos e DERSs.

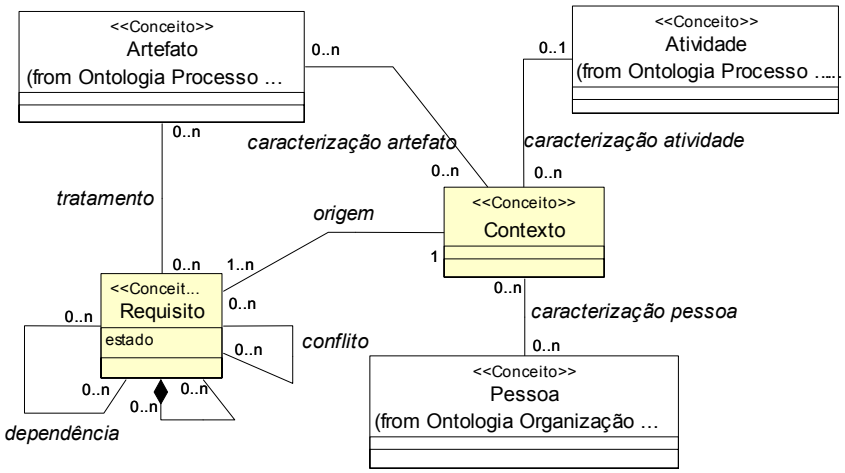


Fig. 6. Relações Importantes para a Rastreabilidade.

A relação de dependência permite estabelecer ligações de dependência entre requisitos (*QC10*), indicando que, se um requisito do qual outros requisitos dependem for alterado, é provável que os requisitos dependentes necessitem ser alterados.

No caso da composição, a situação é análoga. Por meio dessa relação, é possível tratar a decomposição de um requisito em outros (*QC9*). Se um requisito composto for alterado é provável que suas partes tenham de ser revisadas e vice-versa.

Finalmente, podem-se estabelecer relações de conflito entre requisitos (*QC11*), permitindo controlar esses conflitos, mantendo-os aceitáveis ou detectando que um conflito definitivamente não pode existir, levando à revisão dos requisitos envolvidos.

Para se ter um controle de mudanças mais efetivo, é interessante que se possa avaliar o impacto das mudanças de um requisito nos demais artefatos do processo de software. Assim, é importante saber quais artefatos (por exemplo, documento de

especificação de requisitos, modelos e diagramas de análise e projeto, ou mesmo código-fonte) tratam de um requisito, para poder trilhar mudanças em um requisito para esses artefatos e vice-versa (*QC13*).

Para manter a rastreabilidade de um requisito desde a sua concepção, se faz necessário ligar o requisito à sua origem, ou seja, ligá-lo ao contexto no qual ele se originou (*QC7*). Esse contexto pode se caracterizar pela análise de artefatos, interação entre pessoas, consulta a *sites* na Internet etc. e pode, ainda, se dar no âmbito de uma atividade do processo de software.

O estado de um requisito é útil para se ter o controle, em um determinado momento, de como anda seu desenvolvimento (*QC8*). Os possíveis estados de um requisito são definidos organização a organização, e podem ser, por exemplo: *Proposto*, *Em Alteração*, *Sob Verificação*, *Rejeitado* etc.

Por fim, vale destacar que a questão de competência *QC14* é respondida pelo conjunto da conceituação acima discutida.

Qualidade de Requisito e de ERS

A qualidade do Documento de Especificação de Requisitos de Software (DERS) é crucial para o sucesso do projeto. De acordo com a Ontologia de Qualidade de Software [12], a qualidade de um artefato de software pode ser avaliada por meio da definição de características de qualidade e da utilização de métricas para medi-las. Assim, adotando a conceituação dessa ontologia, a questão de competência *QC15* pode ser respondida, como mostra a figura 7. Nessa ontologia, cada artefato pode ter um conjunto de Características de Qualidade de Produto relevantes para a avaliação de sua qualidade. Essas características de qualidade podem ser direta ou indiretamente mensuráveis, sendo que as diretamente mensuráveis têm correlação com métricas, enquanto as indiretamente mensuráveis podem ser medidas por meio da combinação de valores de suas sub-características.

4 Trabalhos Correlatos

Poucos trabalhos têm tratado de ontologias para requisitos. Em [13] é apresentada uma ontologia de requisitos desenvolvida para apoiar um processo genérico de gerência de requisitos no domínio de projetos de engenharia. Essa ontologia é parte de outra ontologia mais geral para capturar conhecimento de projetos de engenharia. Suas questões de competência são agrupadas nas seguintes categorias: refinamento de requisitos, rastreabilidade de requisitos, satisfação de requisito, versão e alteração, relacionamentos de requisitos com partes e estrutura de produto. Embora as categorias mencionadas e as questões de competências abordadas estejam muito próximas às deste trabalho, nossa Ontologia de Requisitos foca no domínio de requisitos de software, fazendo, portando, interação com outras ontologias do domínio de engenharia de software, enquanto a ontologia proposta em [13] reutiliza conceitos de uma ontologia de produtos.

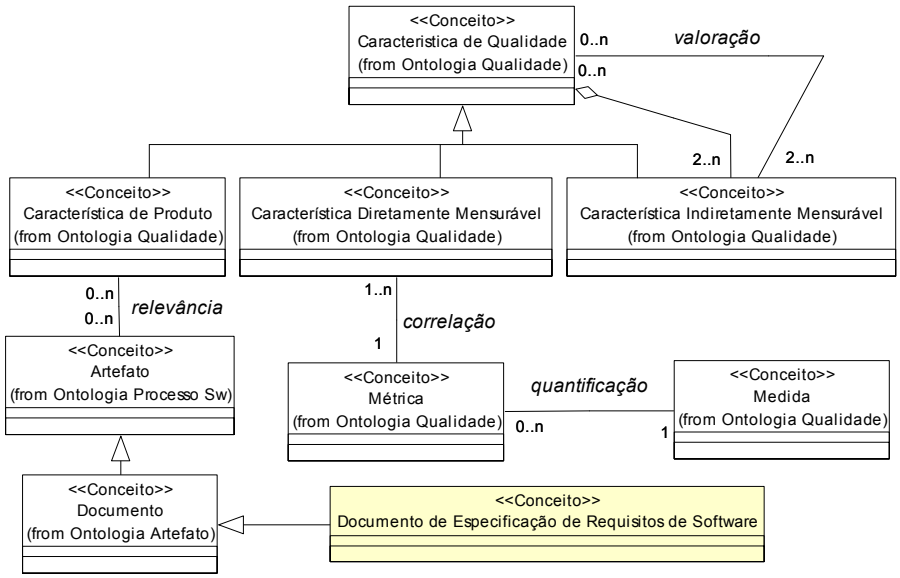


Fig. 7. Qualidade em Requisitos

No contexto de requisitos de software, Ramesh e Jarke [14] apresentam um meta-modelo que provê primitivas para categorização e descrição de modelos de rastreabilidade, permitindo tratar as seguintes questões: Que informação é representada? Quem são os interessados em desempenhar diferentes papéis no controle de objetos? Onde os objetos estão representados? Como a informação é representada? Por que um certo objeto foi criado ou modificado? Quando uma informação foi criada, capturada etc? A partir desse meta-modelo, são instanciados outros modelos de rastreabilidade que contemplam aspectos mais específicos, tais como a derivação de requisitos, a alocação de requisitos a subsistemas e a componentes, necessidades organizacionais, etc.

Ainda que o trabalho de Ramesh e Jarke não se trate de uma ontologia, claramente é correlato à Ontologia de Requisitos apresentada neste trabalho. Algumas das questões levantadas por eles foram listadas também como questões de competência neste trabalho. Contudo, aqui, procurou-se tratar o domínio de requisitos de maneira mais ampla, indo além da rastreabilidade de requisitos.

5 Considerações Finais

Neste trabalho foi apresentada uma Ontologia de Requisitos de Software objetivando definir conceitos e relações sobre o domínio de requisitos de forma facilitar a comunicação entre pessoas, entre agentes de software, construir ferramentas de apoio à Engenharia de Requisitos (ER) mais amplamente utilizáveis e auxiliar no estudo acerca de requisitos de software no contexto do Projeto ODE. Durante a construção dessa ontologia, várias outras ontologias acerca do domínio de engenharia de software

foram analisadas e reutilizadas. Dessa forma buscou-se reutilizar conceituações já estabelecidas.

Como continuação deste trabalho, está-se construindo uma ferramenta de apoio à ER baseada na ontologia proposta e integrada ao ambiente ODE.

Agradecimentos

Este trabalho foi realizado com o apoio do CNPq e da CAPES, entidades do Governo Brasileiro dedicadas ao desenvolvimento científico e tecnológico, da FAPES, Fundação de Apoio à Ciência e Tecnologia do Espírito Santo, e da VixTeam Consultoria e Sistemas, empresa parceira que têm financiado o projeto e dado feedback de sua aplicação a casos reais.

Referências

1. H.F.Hofmann,F. Lehner. "Requirements Engineering as a Success Factor in Software Projects, IEEE Software, July/August 2001.
2. G. Kotonya, I. Sommerville. "Requirements Engineering: Process and Techniques". 1º Edição. Editora Wiley. 1998.
3. R.A. Falbo, et al., "Ontologias e Ambientes de Desenvolvimento de Software Semânticos", JIISIC'2004.
4. N. Guarino. "Formal Ontology in Information System". In: Proceedings of the First Int. Conference on Formal Ontology in Information Systems, Trento, Italy, June 1998. p.3-15.
5. J. C. S. P. Leite. "Gerenciando a Qualidade de Software com Base em Requisitos". In: Qualidade de Software: Teoria e Prática. 1ª. Edição. Prentice Hall. 2001. pp 238-246.
6. S. Robertson; J. Robertson. Mastering the Requirements Process. 1º Edição. Editora ACM Press, Addison Wesley. 1999.
7. R. A. Falbo. "Integração de Conhecimento em um Ambiente de Desenvolvimento de Software". Tese de Doutorado, COPPE/UFRJ, RJ, Dezembro, 1998.
8. R. A. Falbo. "Experiences in Using a Method for Building Domain Ontologies". Proc. of International Workshop on Ontology In Action, Banff, Alberta, Canada, June 2004.
9. P.G. Mian, R.A. Falbo, "Supporting Ontology Development with ODEd", Journal of the Brazilian Computer Science, vol. 9, no. 2, pp 57-76, November 2003.
10. R.A. Falbo, G. Bertollo, "Establishing a Common Vocabulary for Software Organizations Understand Software Processes", International Workshop on Vocabularies, Ontologies and Rules for the Enterprise, VORTE'2005, Enschede, The Netherlands, September 2005.
11. B. V. Nunes. "Integrando Gerência de Configuração de Software, Documentação e Gerência de Conhecimento em um Ambiente de Desenvolvimento de Software". 2005. Dissertação (Mestrado em Informática) –Universidade Federal do Espírito Santo.
12. K. C. Duarte. "Um *Framework* de Reuso no Domínio de Qualidade de Software". Dissertação de Mestrado. Departamento de Informática (DI) da Universidade Federal do Espírito Santo (UFES). Vitória-ES. Junho de 2001.
13. J. Lin, M.S. Fox, T. Bilgic. "A requirement ontology for engineering design", in: M. Sobolewski, M. Fox (Eds.), Proceedings of Advances in Concurrent Engineering (CE'96), Toronto, 1996, pp. 343-351.
14. B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability", IEEE Transactions on Software Engineering, Vol. 27, No. 1, 2000.