Uma Ontologia de Qualidade de Software

Katia Cristina Duarte

Ricardo de Almeida Falbo

Mestrado em Informática - UFES

Av. Fernando Ferrari, s/n, Vitória – ES

{katia, falbo}@inf.ufes.br

Resumo

O primeiro passo em direção à qualidade de software consiste em entender seus conceitos para poder aplicá-los consistentemente. Infelizmente, não há consenso sobre a terminologia usada, o que provoca vários problemas, principalmente de interpretação, quando da definição de um programa de qualidade. Neste artigo, apresentamos uma ontologia de qualidade de software com o objetivo de apoiar a compreensão deste complexo domínio. A meta é utilizar essa ontologia como uma base para o entendimento comum sobre os conceitos de qualidade de software em um tutorial.

Palavras-chave: Qualidade do produto, Educação em Qualidade de Software, Ontologia.

Abstract

The first step towards software quality is to understand what it means. Unfortunately, there is not a consensus about the terminology used. This causes misunderstanding and several problems in the definition of a quality program. In this paper, we present an ontology of software quality, aiming to support the understanding of this complex domain. The goal is to use the ontology as the basis for a tutorial about software quality.

Key-words: Software Product Quality, Education on Software Quality, Ontology.

1. Introdução

Na medida em que cresce a demanda por sistemas complexos, com grande responsabilidade no contexto das organizações, a qualidade desponta como um fator essencial no desenvolvimento de software. Sendo assim, cada vez mais, há uma disposição para se investir em qualidade. Contudo, uma das primeiras dificuldades encontradas na definição e implantação de um programa de qualidade está em compreender o que, de fato, significa qualidade de software. Visando apoiar esse entendimento e, conseqüentemente, a definição de programas de qualidade e a construção de ferramentas de apoio, desenvolvemos uma ontologia de qualidade de software, apresentada parcialmente neste trabalho.

Na seção 2, discutimos brevemente o que é uma ontologia, seu propósito e forma. A seção 3 explora o domínio da qualidade de software, com o objetivo de fundamentar a apresentação da ontologia na seção 4. A seção 5 trata de como a ontologia aqui apresentada está sendo utilizada. Finalmente, na seção 6, são apresentadas as conclusões desse trabalho.

2. Ontologia

Uma ontologia é uma especificação de uma conceituação [1], isto é, uma descrição de conceitos e relações que existem em um domínio de interesse. Basicamente, uma ontologia consiste desses conceitos e relações, e suas definições, propriedades e restrições, descritas na forma de axiomas.

Ontologias são úteis para apoiar a especificação e implementação de qualquer sistema de computação complexo. Uma ontologia pode ser desenvolvida para vários fins, mas, de uma forma geral, os seguintes propósitos são atingidos:

- ajuda as pessoas a compreender melhor uma certa área de conhecimento: no desenvolvimento de uma ontologia, as pessoas envolvidas no processo se vêem diante de um desafio: explicar seu entendimento sobre o domínio em questão, o que as faz refletir e melhorar sua compreensão sobre esse domínio;
- ajuda as pessoas a atingir um consenso no seu entendimento sobre uma área de conhecimento: geralmente, em uma determinada área de conhecimento, diferentes especialistas têm entendimento diferenciado sobre os conceitos envolvidos, o que leva a problemas na comunicação. Ao se construir uma ontologia, essas diferenças são explicitadas e busca-se um consenso sobre seu significado e sua importância;
- ajuda outras pessoas a compreender uma certa área de conhecimento: uma vez que haja uma ontologia sobre uma determinada área de conhecimento desenvolvida, uma pessoa que deseje aprender mais sobre essa área não precisa se reportar sempre a um especialista. Ela pode estudar a ontologia e aprender sobre o domínio em questão, absorvendo um conhecimento geral e de consenso.

De fato, todos esses propósitos foram considerados para a construção de uma ontologia de qualidade. O objetivo maior estava relacionado ao entendimento deste complexo domínio. Muitas vezes, é difícil encontrar fontes confiáveis, que contenham informações básicas, necessárias à compreensão do que vem a ser qualidade de software, de como se pode medir a qualidade de um produto ou processo e diversos outros pontos importantes nesse contexto.

Outro aspecto considerado diz respeito ao fato de se despender bastante tempo para reunir conceitos e absorvê-los e não ser possível reaproveitar esse esforço no aprendizado de outras pessoas. Sempre que alguém precisa aprender sobre qualidade, tem todo o trabalho para obter informações a respeito, tendo que passar por dificuldades que outros passaram, a fim de percorrer o tortuoso caminho do aprendizado. O aprendiz precisa procurar material bibliográfico sobre o assunto e, quando encontra, não tem certeza de que esse material é completo, confiável, e apresenta o conteúdo de forma consensual e não sob o ponto de vista de um autor. Ainda no processo de aprendizado, pode ser útil recorrer a especialistas no assunto, o que nem sempre é possível.

Especialistas e fontes bibliográficas utilizam termos distintos para referenciar os mesmos conceitos. Tal fato, à medida que o estudo vai se desenvolvendo, pode provocar um entendimento confuso, principalmente, se for utilizado um número razoável de fontes de informação. Pode-se absorver e empregar conceitos de maneira equivocada, pois, em muitos casos, depara-se com a falta de clareza e ambigüidade com que eles são tratados.

É nesse contexto que identificamos a necessidade de se construir uma ontologia de qualidade de software, formalizando conceitos e relações pertencentes a esse domínio.

No processo de construção da ontologia, foram consultados, além de referências bibliográficas, especialistas com larga experiência na área, buscando obter um consenso sobre o entendimento dos mesmos sobre o assunto. Os conceitos, suas definições e restrições eram submetidos aos especialistas para que esses opinassem sobre a validade e aderência a seu modo de trabalho com qualidade de software. Pontos de conflito foram discutidos em reuniões e aspectos relacionados a visões particulares foram eliminados da ontologia. Além disso, para validar a ontologia procurou-se observar se métodos e normas em uso poderiam ser descritas usando a ontologia.

3. Qualidade de Software

No contexto de desenvolvimento de software, qualidade pode ser entendida como um conjunto de características a serem satisfeitas em um determinado grau, de modo que o produto de software atenda às necessidades explícitas e implícitas de seus usuários [2]. Entretanto, não se obtém qualidade do produto de forma espontânea. Ela tem de ser construída. Assim, a qualidade do produto depende fortemente da qualidade de seu processo de desenvolvimento. Neste trabalho, contudo, não discutimos o processo de software em si, mas sim nos apoiamos na ontologia de processo de software desenvolvida em [3].

Para avaliar a qualidade, é preciso haver meios de medi-la. Ou seja, é preciso obter uma medida que quantifique o grau de alcance de uma característica de qualidade. Assim, para computar uma característica de qualidade, é necessário estabelecer uma métrica capaz de quantificá-la e fazer uma medição para determinar a medida, resultado da aplicação da métrica. Por exemplo, digamos que se deseja saber o tamanho de determinado produto de software. Então, a métrica utilizada poderia ser “número de linhas de código”. A medição poderia ser “contar o número de linhas de código, desconsiderando comentários, de cada programa contido no software”. E a medida seria o número obtido na medição, uma quantidade, por exemplo, “5000 linhas de código”.

Muitas vezes, características de qualidade não podem ser medidas diretamente através de métricas, sendo necessário decompô-las em sub-características. Assim, há características que são diretamente mensuráveis e outras que são apenas indiretamente mensuráveis. Uma característica diretamente mensurável possui uma métrica correlacionada, à qual, através de uma medição, se aplica uma medida. As características indiretamente mensuráveis podem ser descritas em termos de outras características, sendo calculada, em última instância, via características diretamente mensuráveis.

Desta forma, uma certa característica indiretamente mensurável pode ser computada de diferentes maneiras, em função das sub-características escolhidas, já que essa escolha é, muitas vezes, função da fase do processo ou artefato para o qual as sub-caracterísiticas estão sendo aplicadas, do paradigma ou da tecnologia de desenvolvimento. Em outras palavras, diferentes métricas podem ser aplicadas, dependendo da atividade do ciclo de vida ou do artefato ao qual se aplica. Em todos os casos, contudo, há de se respeitar a pertinência em relação ao paradigma e à tecnologia de desenvolvimento adotados no desenvolvimento.

Processos possuem características de qualidade próprias, tais como desempenho, estabilidade e capacidade [4], mas informações sobre a qualidade do produto gerado podem ser também importantes na avaliação do processo, uma vez que a baixa qualidade do produto pode ser um reflexo de falha ou inadequação do processo utilizado. Assim, características de qualidade do processo podem ser computadas a partir de características de qualidade do

produto. O contrário, contudo, não é possível. Ou seja, características de qualidade do produto só podem ser computadas a partir de outras características de produto.

4. Uma Ontologia de Qualidade

A construção da ontologia de qualidade seguiu o método proposto em [3]. Basicamente, esse método inicia com uma especificação de requisitos da ontologia, na qual são levantadas as suas questões de competência, isto é, questões que a ontologia tem de ser capaz de responder. A seguir, dá-se a captura da ontologia, em que são identificados os conceitos e relações envolvidos no domínio, bem como as restrições, descritas na forma de axiomas. Nesta etapa, foram utilizadas técnicas de elicitação de conhecimento junto a especialistas, além de consultas a fontes bibliográficas, tais como [4, 5, 6, 7], dentre outras. Um modelo usando LINGO [3] (LINguagem Gráfica para descrever Ontologias) foi gerado, assim como um dicionário de termos contendo as definições em linguagem natural. Nesta etapa, ainda, foram definidos axiomas em linguagem natural, para capturar as restrições impostas pelo domínio. Na seqüência, deve-se proceder a formalização da ontologia. Para essa etapa, utilizamos a lógica de primeira ordem. Em paralelo a estas atividades, ocorrem a documentação, avaliação e integração com ontologias existentes. Durante a construção da ontologia de qualidade, fez-se necessária a integração com uma ontologia de processo de software [3]. Os conceitos e relações oriundos dessa última são apresentados destacados.

Dadas as limitações de espaço, neste artigo apresentamos apenas parte da ontologia, considerando parcialmente as seguintes questões de competência:

1. Qual é a natureza de uma característica de qualidade?
2. Que características de qualidade são relevantes para um dado artefato, atividade ou processo?
3. Como uma característica de qualidade pode ser medida?
4. Que métricas podem ser usadas para quantificar uma dada característica?

Analisando as questões de competência anteriormente relacionadas, identificamos alguns aspectos bastante relevantes no domínio de qualidade, que foram alvo desta ontologia:

- Características de Qualidade: Natureza e Estrutura (questões 1 e 2)
- Qualidade: Como Medir (questões 3 e 4)

4.1- Características de Qualidade: Natureza e Estrutura

A figura 1 mostra o modelo em LINGO referente à natureza e à estrutura das características de qualidade. A seguir, na tabela 1, é apresentado o dicionário dos termos definidos para a ontologia de qualidade. Os conceitos oriundos da ontologia de processo de software [3] estão descritos na tabela 2.

Como mostra a figura 1, características de qualidade podem ser classificadas em *características indiretamente mensuráveis* e *características diretamente mensuráveis* e em *características do processo* e *características do produto*.

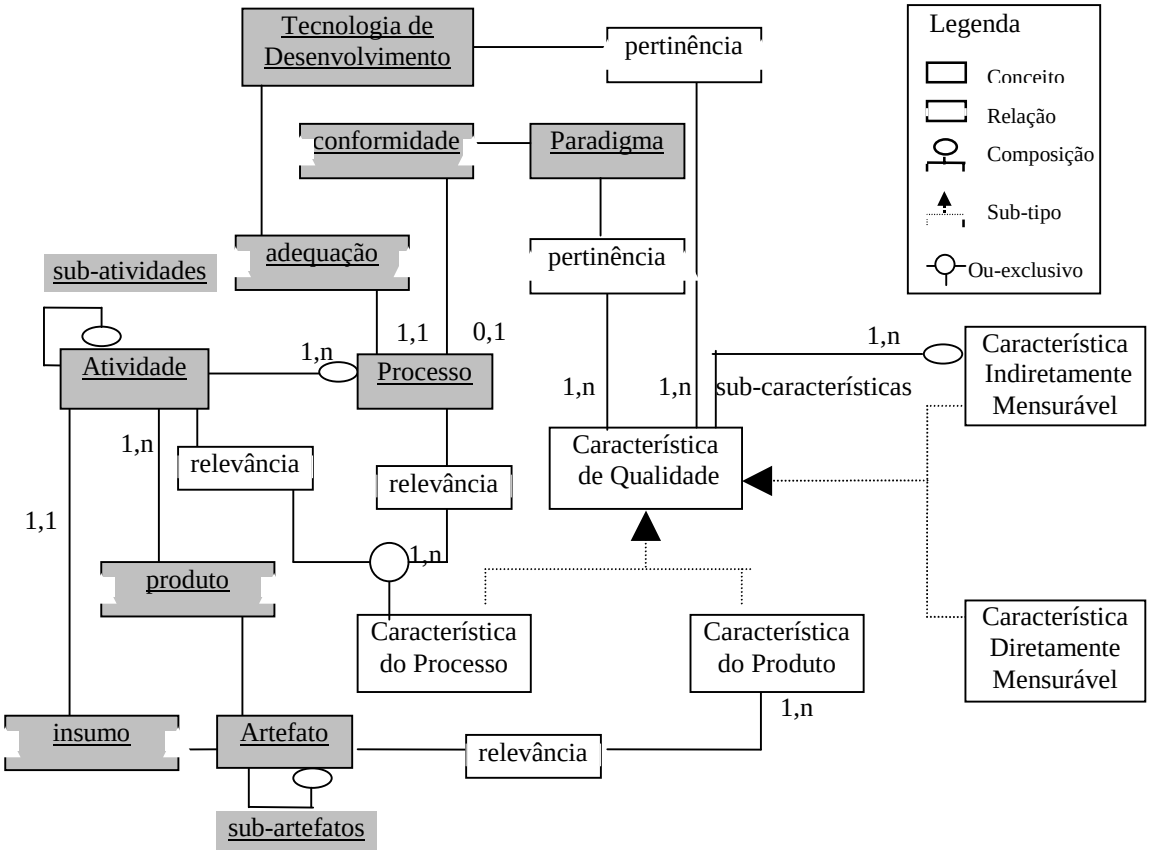

Figura 1 - Características de Qualidade: Natureza e Estrutura.

Característica de qualidade	atributos de um **artefato** ou **processo de software** através dos quais a qualidade de um software pode ser avaliada.
Característica de qualidade diretamente mensurável	**característica de qualidade** que pode ser computada diretamente através de uma **métrica**, sem a necessidade de combinar resultados de **sub-características**. **Ex**: tamanho do software.
Característica de qualidade indiretamente mensurável	**característica de qualidade** que só pode ser computada combinando resultados de **sub-características**. **Ex**: funcionalidade, manutenibilidade.
Característica de qualidade do processo	**característica de qualidade** relevante para descrever e avaliar a qualidade do **processo de software** ou de uma de suas **atividades**. **Ex**: Desempenho do processo, capacitação, produtividade.
Característica de qualidade do produto	**característica de qualidade** relevante para descrever e avaliar a qualidade de um **artefato**. **Ex**: Tamanho, legibilidade, confiabilidade.
Pertinência	relação entre os conceitos **característica de qualidade** e **paradigma**, e **característica de qualidade** e **tecnologia de desenvolvimento**, indicando que uma **característica de qualidade** é pertinente para avaliar a qualidade de um **artefato/processo** desenvolvido usando um determinado **paradigma** ou **tecnologia de desenvolvimento**.

Relevância	relação entre os conceitos **característica de qualidade do processo** e **atividade/processo** e **característica de qualidade do produto** e **artefato**, indicando que uma determinada **característica de qualidade** é relevante para avaliar a qualidade de um **artefato** ou **processo/ atividade**.
Sub-característica	relação entre **características de qualidade**, indicando que uma **característica indiretamente mensurável** é decomposta em outras características.

Tabela 1 – Dicionário de Termos da Ontologia de Qualidade.

Adequação	relação entre um **processo** e uma **tecnologia de desenvolvimento**, indicando que o **processo** é definido para a **tecnologia de desenvolvimento**.
Artefato	um **insumo** para, ou um **produto** de, uma **atividade**, no sentido de ser um objeto de transformação da atividade. Uma importante propriedade de um artefato é o seu tipo. Os tipos de **artefatos** incluem: **artefatos de código**, **componentes de software** e **documentos**. O próprio software (produto final do desenvolvimento) é também um artefato.
Conformidade	relação entre um **processo** e um **paradigma**, indicando que o **processo** adota o **paradigma**.
Insumo	relação entre um **artefato** e uma **atividade**, indicando que o **artefato** é utilizado como "matéria-prima" pela **atividade**, sendo de alguma forma incorporado a outro **artefato**, o **produto** da **atividade**.
Produto	relação entre um **artefato** e uma **atividade**, indicando que o **artefato** é produzido pela **atividade**.
Atividade	ação que transforma **artefatos** de entrada (**insumos**) em **artefatos** de saída (**produtos**). Em função de sua natureza, atividades podem ser classificadas em **atividades de gerência**, **atividades de construção** e **atividades de avaliação da qualidade**.
Paradigma	filosofia adotada na construção do software, abrangendo um conjunto de princípios e conceitos que norteiam o desenvolvimento. Ex.: paradigma estrutural, orientado a objetos, etc.
Processo	a atividade global de desenvolvimento. É, de fato, uma infra-estrutura contendo as **atividades** das diversas naturezas envolvidas no desenvolvimento de software. A definição de um processo de software depende fortemente da **tecnologia de desenvolvimento** e do **paradigma** a serem adotados no desenvolvimento e pode ser facilitada pela adoção de um modelo de ciclo de vida como referência.
Sub-atividade	faceta da relação de composição entre duas **atividades** a_1 e a_2. Se a_2 é parte de a_1 então a_2 é dita uma sub-atividade de a_1.
Sub-artefato	faceta da relação de composição entre dois **artefatos** s_1 e s_2. Se s_2 é parte de s_1 então s_2 é dito um sub-artefato de s_1.
Tecnologia de Desenvolvimento	tecnologia a ser empregada no desenvolvimento do software. Ex.: tecnologia convencional de processamento de dados, tecnologia de sistemas baseados em conhecimento, etc.

Tabela 2 – Dicionário de Termos Oriundos da Ontologia de Processo [3].

Uma *característica indiretamente mensurável* deverá ser computada através da combinação de medidas de *sub-características*. Uma *sub-característica* poderá ser direta ou indiretamente mensurável. Uma *característica diretamente mensurável* pode ser computada através de uma métrica, não necessitando combinar medidas de *sub-características*.

A fim de formalizar a existência dos tipos de características de qualidade, são definidos os seguintes predicados, representando cada um dos tipos identificados na taxonomia: *carq(cq)*, indicando que cq é uma característica de qualidade, *carqdirm(cq)*, indicando que cq é uma característica de qualidade diretamente mensurável, *carqindm(cq)*, indicando que cq é uma característica de qualidade indiretamente mensurável, *carqproc(cq)*, indicando que cq é uma característica de qualidade do processo, e *carqprod(cq)*, indicando que cq é uma característica de qualidade do produto.

A notação de sub-tipo utilizada na figura 1 reflete os seguintes axiomas:

$$(\forall\ cq)\ (\ carqindm(cq) \rightarrow carq(cq)\ )$$

$$(\forall\ cq)\ (\ carqdirm(cq) \rightarrow carq(cq)\ )$$

$$(\forall\ cq)\ (\ carqproc(cq) \rightarrow carq(cq)\ )$$

$$(\forall\ cq)\ (\ carqprod(cq) \rightarrow carq(cq)\ )$$

Axiomas dessa natureza são implicitamente descritos pela notação de LINGO: sempre que a notação para sub-tipos for empregada, existem axiomas deste tipo.

O predicado *subcarq(cq_1 , cq_2)* indica que *cq_1* é uma sub-característica da característica de qualidade *cq_2*. Assim como no caso da notação de sub-tipos, a notação de composição de LINGO reflete axiomas, neste caso, os seguintes:

$$(\forall\ cq_1, cq_2\ )\ (\ subcarq(cq_1\ ,\ cq_2) \rightarrow\ \neg\ \ subcarq(cq_2\ ,\ cq_1\ )\ )$$

$$(\forall\ cq_1, cq_2, cq_3)\ (\ subcarq(cq_1\ ,\ cq_2\ ) \wedge subcarq(cq_2\ ,\ cq_3\ ) \rightarrow subcarq(cq_1\ ,\ cq_3\ )\ )$$

$$(\forall\ cq_1\ ,\ cq_2\ )\ (\ disjunto(cq_1\ ,\ cq_2\ ) \leftrightarrow\ \neg\ \ (\exists\ \ cq_3\ )\ (\ subcarq(cq_3\ ,\ cq_1\ ) \wedge subcarq(cq_3\ ,\ cq_2\ )\ )\ )$$

$$(\forall\ cq_1\ )\ (\ carqdirm\ (cq_1\ ) \leftrightarrow \neg\ (\exists\ \ cq_2\ )\ (\ subcarq(cq_2\ ,\ cq_1\ )\ )\ )$$

De fato, este é um traço marcante de LINGO: toda notação presente na linguagem reflete um conjunto de axiomas que impõe restrições ao domínio que está sendo modelado. Esses axiomas são ditos epistemológicos, por cuidarem de aspectos relacionados à estruturação dos conceitos.

Ainda considerando axiomas derivados automaticamente de LINGO, o seguinte axioma epistemológico é derivado pela cardinalidade mínima *1* da relação sub-característica em relação ao conceito característica de qualidade indiretamente mensurável: Se *cq* é uma característica de qualidade indiretamente mensurável, então existe uma característica de qualidade *cq_1*, que é sub-característica de *cq*. Ou seja, não pode haver uma característica indiretamente mensurável que não possua uma sub-característica de qualidade.

$$(\forall\ cq)\ (\ carqindm(cq) \rightarrow (\exists\ cq_1)\ (\ subcarq(cq_1,\ cq)\ )\ )$$

Uma vez que a cardinalidade *0,n* não impõe nenhuma restrição, ela é omitida em LINGO.

Contudo, há outros axiomas que não são capturados pela notação de LINGO e regem importantes restrições. Dentre eles, destacam-se os axiomas de consolidação [3], os quais não são utilizados para derivar nova informação, mas sim para impor que certas condicionantes sejam verificadas no estabelecimento de relações, como nas situações apresentadas a seguir.

Uma característica indiretamente mensurável do processo pode ser computada combinando resultados de sub-características do processo ou de sub-características do produto. Mas uma característica indiretamente mensurável do produto só pode ser computada combinando resultados de sub-características do produto.

$$(\forall\ cq, cq_1)\ (\ subcarq(cq_1, cq) \wedge carqprod(cq) \rightarrow carqprod(cq_1)\ )$$

Se uma característica de qualidade do processo *cq* é relevante a uma atividade *a*, e cq_1, subcaracterística de *cq*, é uma característica de qualidade do produto, então existe um artefato *s* produzido por *a*, para o qual cq_1 é relevante.

$$(\forall\ cq, cq_1, a)\ (\ ativ\text{-}relevância(cq, a) \wedge subcarq(cq_1, cq) \wedge carqprod(cq_1) \rightarrow (\exists\ s)\ (\ produto(s, a) \wedge prod\text{-}relevância(cq_1, s)\ )\ )$$

onde os predicados *ativ-relevância(cq,a)* e *prod-relevância(cq,s)* indicam, respectivamente, que as características de qualidade *cq* e cq_1 são relevantes para avaliar a qualidade da atividade *a* e do artefato *s*.

Se cq_1 é uma sub-característica de *cq* e *cq* é pertinente para o paradigma *pd* ou para a tecnologia de desenvolvimento *td*, então cq_1 tem que ser pertinente para o mesmo paradigma *pd* e para a mesma tecnologia de desenvolvimento *td*.

$$(\forall\ cq, cq_1, pd)\ (\ subcarq(cq_1, cq) \wedge pdg\text{-}pertinência(cq, pd)\ \rightarrow\ pdg\text{-}pertinência(cq_1, pd)\ )$$

$$(\forall\ cq, cq_1, td)\ (\ subcarq(cq_1, cq) \wedge td\text{-}pertinência(cq, td)\ \rightarrow\ td\text{-}pertinência(cq_1, td)\ )$$

onde os predicados *pdg-pertinência(cq,pd)* e *td-pertinência(cq,td)* indicam que a característica de qualidade *cq* é pertinente para avaliar a qualidade de produtos e processos definidos segundo o paradigma *pd* e a tecnologia de desenvolvimento *td*, respectivamente.

Se uma característica de qualidade do processo *cq* é relevante para um processo *pr*, o qual está em conformidade com o paradigma *pd* e é adequado à tecnologia *td*, então *cq* tem que ser pertinente ao paradigma *pd* e à tecnologia *td*.

$$(\forall\ cq, pr, pd)\ (\ proc\text{-}relevância(cq, pr) \wedge processo\text{-}conformidade(pr, pd) \rightarrow\ pdg\text{-}pertinência(cq, pd)\ )$$

$$(\forall\ cq, pr, td)\ (\ proc\text{-}relevância(cq, pr) \wedge\ processo\text{-}adequação(pr, td) \rightarrow\ td\text{-}pertinência(cq, td)\ )$$

onde o predicado *proc-relevância(cq,pr)* indica que a característica de qualidade *cq* é relevante para avaliar a qualidade do processo *pr* e os predicados *processo-conformidade(pr,pd)* e *processo-adequação(pr,td)* indicam que o processo *pr* foi definido segundo o paradigma *pd* e é adequado à tecnologia de desenvolvimento *td*, respectivamente.

Outros axiomas envolvendo os conceitos mostrados na figura 1 foram descritos na ontologia. Contudo, devido à limitação de espaço, eles não são apresentados.

4.2 - Qualidade: Como Medir

A figura 2 mostra o modelo em LINGO referente às questões de competência 3 e 4, que dizem respeito a como medir a qualidade. A seguir, na tabela 2, é apresentado o dicionário de termos dessa porção da ontologia.

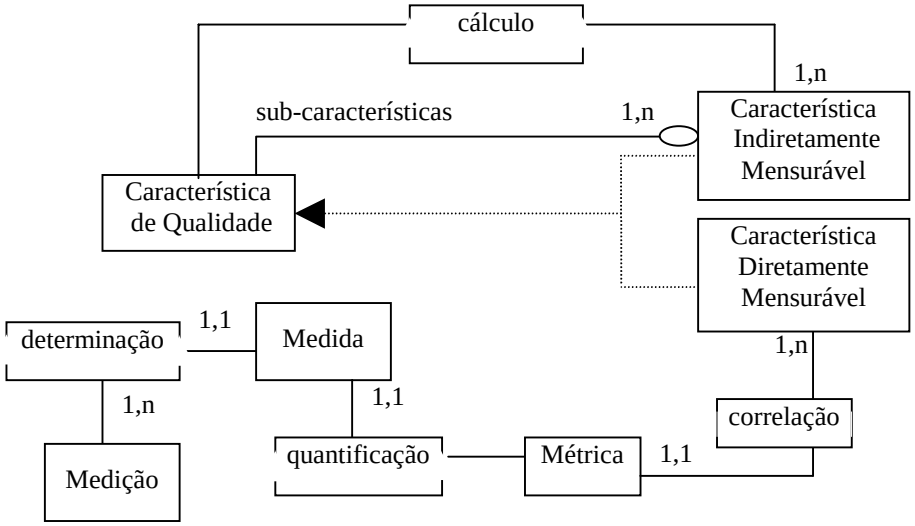

Figura 2 – Qualidade: Como Medir

Cálculo	relação entre **características indiretamente mensuráveis** e **características de qualidade**, indicando que uma **característica indiretamente mensurável** deve ser calculada através de outras **características de qualidade**, direta ou indiretamente mensuráveis.
Correlação	relação entre os conceitos **métrica** e **característica diretamente mensurável**, indicando que essa característica pode ser computada através da **métrica**. **Ex**: a métrica "número de linhas de código" pode ser usada para computar a característica diretamente mensurável "tamanho".
Determinação	relação entre os conceitos **medição** e **medida**, indicando que uma **medição** determina uma **medida**. **Ex**: contar o número de linhas de código determina um valor, que é a medida.
Métrica	é o que se deseja medir, ou seja, a unidade de **medição**. No contexto de qualidade de software, está correlacionada a uma **característica de qualidade**. **Ex**: número de linhas de código.
Medida	valor que uma **métrica** assume no ato da **medição**. **Ex**: 300 linhas de código.
Medição	**atividade de avaliação da qualidade** que consiste na determinação de uma **medida** para uma **métrica**. **Ex**: obter o número de linhas de código.
Quantificação	relação entre os conceitos **medida** e **métrica**, indicando que uma **medida** quantifica uma **métrica**. **Ex**: a medida "5000 linhas de código" quantifica a métrica "número de linhas de código".

Tabela 3 – Dicionário de Termos

Assim como para o modelo da figura 1, diversos axiomas foram definidos, tais como os descritos a seguir.

Se uma característica de qualidade *cq* é diretamente mensurável, então tem que existir uma métrica *m* correlacionada a *cq*.

$$(\forall\ cq, m)\ (\ carqdirm(cq) \rightarrow (\exists\ m)\ (\ correlação(m, cq)\ )$$

onde o predicado *correlação(m, cq)* indica que a métrica *m* está correlacionada à característica de qualidade diretamente mensurável *cq*.

Uma característica de qualidade cq_1 só pode entrar no cálculo de uma característica de qualidade indiretamente mensurável *cq*, se cq_1 é uma sub-característica de *cq*.

$$(\forall\ cq, cq_1)\ (\ cálculo(cq, cq_1) \rightarrow\ carqindm(cq) \wedge subcarq(cq_1, cq))$$

onde o predicado *cálculo(cq,* cq_1*)* indica que a característica de qualidade *cq* é calculada a partir da característica de qualidade cq_1.

5 - Usos da Ontologia

A partir da ontologia de qualidade de software, foi construída uma infra-estrutura de reuso, mais especificamente, um *framework* de objetos, para apoiar a construção de ferramentas neste domínio. O processo de derivação do framework a partir da ontologia foi conduzido segundo a abordagem proposta em [8].

O framework derivado está sendo utilizado na construção de duas ferramentas: um Tutorial sobre Qualidade de Software e uma Ferramenta CASE de apoio ao planejamento e controle da qualidade de sistemas orientados a objetos.

O objetivo do Tutorial é permitir a navegação da ontologia, mostrando, através de simulação, o significado prático dos conceitos apresentados. Assim, o tutorial poderá ser utilizado para apoiar o aprendizado sobre qualidade, de modo que uma organização que pretenda iniciar um programa de qualidade, ou qualquer pessoa que esteja interessada, possa, navegando no hiperdocumento, compreender e aprender sobre este domínio complexo.

A ferramenta CASE de suporte ao planejamento e controle da qualidade visa apoiar a elaboração de um Plano de Controle e Garantia da Qualidade de Software, definindo as atividades de avaliação da qualidade a serem realizadas, os artefatos, atividades e processos sob avaliação, as características de qualidade a serem utilizadas e as métricas usadas para computá-las. A ferramenta oferece, ainda, facilidades para coletar automaticamente algumas métricas relacionadas ao desenvolvimento orientado a objetos [9].

6 - Conclusões

Este artigo apresentou uma ontologia para o domínio de qualidade de software. Foram definidos conceitos e relações, utilizando uma linguagem gráfica, e axiomas, formalizados em lógica de primeira ordem. Essa ontologia foi utilizada para a construção de uma infra-estrutura de reuso no domínio, a partir da qual se está construindo um tutorial para apoiar o aprendizado sobre qualidade de software e uma ferramenta de apoio ao planejamento e controle da qualidade de sistemas orientados a objetos. Com esta abordagem, ambas as ferramentas estão sendo construídas sobre uma base sólida, que considera aspectos bastante investigados e considerados úteis por diversos especialistas e fontes bibliográficas.

Referências Bibliográficas

[1] Gruber, T.R., "Towards principles for the design of ontologies used for knowledge sharing", *Int. J. Human-Computer Studies*, v. 43, n. 5/6, 1995.

[2] Rocha, A.R.C., et al., Uma Experiência na Definição do Processo de Desenvolvimento e Avaliação de Software segundo as Normas ISO, Relatório Técnico ES-302/94, COPPE/UFRJ, Junho 1994.

[3] Falbo, R. A., Integração de Conhecimento em um Ambiente de Desenvolvimento de Software, Tese de Doutorado, COPPE/UFRJ, Dezembro 1998.

[4] Florac, W.A., Carleton, A.D., *Measuring the Software Process*, Addison-Wesley, 1999.

[5] ISO/IEC 9126, Information technology - Software product evaluation - Quality characteristics and guidelines for their use", 1991 (E).

[6] Fiorini, S.T., Staa, A., Baptista, R.M., *Engenharia de Software com CMM*, Editora Brasport, Rio de Janeiro, 1998.

[7] ISO 12207. Information technology – Software life cycle processes, 1995 (E).

[8] Guizzardi, G., Desenvolvimento para e com Reuso: Um Estudo de Caso no Domínio de Vídeo sob Demanda, Dissertação de Mestrado, Mestrado em Informática da UFES, Julho 2000.

[9] Lorenz, M., Kidd, J., *Object-Oriented Software Metrics: A Practical Guide*, Prentice Hall Inc., 1994.