Uma Arquitetura de Referência para Medição de Software

Ciro Xavier Maretto, Monalessa Perini Barcellos

Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO), Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, Brasil

{ciro.maretto, monalessa}@inf.ufes.br

Abstract. *Software measurement (SM) is a fundamental practice for project management and software process improvement (SPI). In models that address SPI in levels, such as MR-MPS-SW and CMMI, measurement starts at the initial levels and evolves as the maturity level increases. In the higher maturity levels, measurement includes carrying out statistical process control. In order to perform SM in an effective way, a supporting computational infrastructure is necessary. This paper presents a platform independent reference architecture for SM that has been defined on the basis of a reference ontology. As proof of concept the proposal was used to develop a specific architecture and a tool. An experimental study was carried out aiming at a preliminary evaluation.*

Resumo. *Medição de software é prática fundamental para o gerenciamento de projetos e melhoria de processos de software. Em modelos que tratam a melhoria de processos em níveis, tais como o MR-MPS-SW e o CMMI, a medição começa nos níveis iniciais e evolui na medida em que a maturidade aumenta. Nos níveis mais altos, a medição inclui controle estatístico de processos. Para realizar medição efetivamente, uma infraestrutura computacional é necessária. Este artigo apresenta uma arquitetura de referência para medição de software desenvolvida com base em uma ontologia de referência. Como prova de conceito, a proposta foi usada para desenvolver uma arquitetura específica e uma ferramenta. A avaliação preliminar da proposta foi feita em um estudo experimental.*

1. Introdução

Medição de software é uma avaliação quantitativa de qualquer aspecto de processos e produtos da Engenharia de Software, que permite seu melhor entendimento e, com isso, auxilia o planejamento, controle e melhoria do que se produz e de como é produzido [Bass et al. 1999]. A medição auxilia as organizações de diversas maneiras. Por exemplo, no contexto do gerenciamento de projetos, ajuda a elaborar planos realísticos, monitorar progresso, identificar problemas e justificar decisões [Mcgarry et al. 2002].

Em modelos de maturidade, como o CMMI (*Capability Maturity Model Integration)* [SEI 2010] e o MR-MPS-SW (Modelo de Referência para Melhoria do Processo de Software Brasileiro) [SOFTEX 2012], que organizam os processos de software em níveis, a medição se encontra nos níveis iniciais (nível 2 do CMMI e nível F do MR-MPS-SW) e evolui à medida que o nível de maturidade aumenta. Na alta maturidade (níveis 4 e 5 do CMMI e A e B do MR-MPS-SW) deve ser realizado o controle estatístico de processos (CEP), que requer atenção extra em alguns aspectos da medição, como, por exemplo, a frequência de coleta e o armazenamento dos dados. Neste artigo o

termo medição tradicional é usado para se referir à medição realizada nos níveis anteriores à alta maturidade.

Devido à natureza das atividades de medição, ferramentas de apoio são necessárias para uma implementação bem sucedida. Porém, dentre os problemas enfrentados pelas organizações, destaca-se a falta de ferramentas adequadas para apoiar o processo de medição [De Lucia et al. 2003]. Como consequência, é comum o uso de soluções baseadas em planilhas ou bancos de dados mal estruturados, o que pode comprometer a qualidade e a utilidade dos dados coletados [Dumke e Ebert 2010]. Nos níveis iniciais de maturidade as planilhas parecem atender às necessidades das organizações, mas à medida que as organizações amadurecem seus processos, os problemas dessa abordagem se tornam mais expressivos e, muitas vezes, para alcançar a alta maturidade, organizações precisam descartar os dados armazenados nas planilhas, desenvolver um repositório de medidas utilizando tecnologias adequadas (por exemplo, sistemas gerenciadores de bancos de dados) e reiniciar a coleta e armazenamento de dados dos projetos. Dessa forma, uma boa prática é a definição de uma infraestrutura de apoio que possa ser utilizada desde o início da medição até a alta maturidade ou que possa ser estendida para tal [Barcellos 2009]. Essa infraestrutura, formada por diferentes componentes, pode ser definida por meio de uma arquitetura. Uma arquitetura pode ser entendida como uma estrutura lógica na qual os componentes são organizados e integrados [Zachman 1987].

Existem algumas propostas para arquiteturas de medição de software na literatura. Porém, usualmente, são soluções específicas, desenvolvidas para um contexto particular e não auxiliam organizações a definirem suas próprias arquiteturas. Além disso, comumente, as arquiteturas propostas não cobrem aspectos relacionados à medição em alta maturidade. Para tratar essa lacuna faz-se necessária uma arquitetura que possa ser utilizada como base para as organizações definirem suas próprias arquiteturas. Nesse sentido, faz-se necessária uma arquitetura de referência. Segundo Muller (2013), uma arquitetura de referência captura a essência de um conjunto de sistemas e serve de base para o desenvolvimento de novos sistemas. Assim, arquiteturas de referência são concebidas visando, principalmente, ao reúso. Nakagawa (2009) ressalta que o uso de ontologias de domínio como base para o desenvolvimento de arquiteturas de referência contribui para o entendimento do domínio e identificação dos requisitos a serem tratados. Mais especificamente, Maretto e Barcellos (2013a) defendem o uso de ontologias de referência, isto é, ontologias desenvolvidas com o único objetivo de fazer a melhor descrição possível de um domínio na realidade, sob um certo ponto de vista e nível granularidade [Guizzardi 2007]. Para alcançar fidelidade à realidade e clareza conceitual, idealmente, ontologias de referência devem ser construídas tendo ontologias de fundamentação como base [Guarino 1998].

Visando auxiliar as organizações a definirem suas próprias arquiteturas de medição, foi desenvolvida uma arquitetura de referência para medição de software, considerando tanto medição tradicional quanto em alta maturidade. Essa arquitetura foi desenvolvida com base na Ontologia de Referência para Medição de Software descrita em [Barcellos et al. 2010a; Barcellos et al. 2010b; Barcellos et al. 2010c].

O desenvolvimento do trabalho descrito neste artigo teve início com uma revisão informal da literatura, a qual foi seguida por um mapeamento sistemático onde foram investigadas e analisadas arquiteturas de medição propostas na literatura. O próximo

passo foi definir a abordagem a ser seguida para a definição e avaliação da arquitetura de referência. Seguindo-se a abordagem definida, a arquitetura de referência foi, então, desenvolvida e foi utilizada no desenvolvimento de uma arquitetura específica e de uma ferramenta. Em seguida, foi realizado um estudo experimental para avaliar a arquitetura de referência. Por fim, os resultados de todos os passos realizados durante o desenvolvimento do trabalho foram documentados em [Maretto 2013].

Este artigo está assim organizado: após esta introdução, na seção 2, é apresentada uma breve fundamentação teórica sobre medição de software, CEP e arquiteturas de medição; na seção 3, é descrita a abordagem desenvolvida para guiar a definição de arquiteturas de referência; na seção 4, a arquitetura de referência para medição de software é apresentada; na seção 5, a avaliação da arquitetura de referência e os resultados são comentados; e, finalmente, na seção 6 são realizadas as considerações finais do artigo.

2. Medição de Software, Controle Estatístico de Processos e Arquiteturas para Medição de Software

O processo de medição de software consiste em planejamento da medição, execução da medição e avaliação da medição [ISO/IEC 2008]. Para realizar medição, inicialmente, a organização deve planejá-la. Baseando-se em seus objetivos e necessidades de informação, a organização deve definir os tipos de entidades (ex.: processos e artefatos) e suas propriedades (ex.: tamanho e custo) que serão medidos, bem como as medidas que serão usadas para quantificar essas propriedades e como essas medidas deverão ser coletadas e analisadas. Uma vez planejada, a medição pode ser realizada. A execução da medição envolve coletar dados para as medidas, armazená-los e analisá-los, a fim de obter informações úteis à tomada de decisão. Por fim, o processo de medição e seus resultados são avaliados buscando-se identificar potenciais melhorias [Barcellos et al. 2010b].

A medição de software é realizada de acordo com o nível de maturidade organizacional. Nos níveis iniciais de maturidade, o foco é principalmente na comparação entre valores planejados e realizados nos projetos. Na alta maturidade, o foco se estende para a compreensão e melhoria do desempenho dos processos, sendo necessário realizar o CEP, que usa ferramentas estatísticas para analisar o comportamento dos processos [Maretto 2013]. No contexto de comportamento de processos, há dois conceitos principais: estabilidade e capacidade. Um processo é considerado estável se o mesmo é repetível, ou seja, tem um comportamento que varia dentro de limites esperados e considerados aceitáveis, calculados a partir de dados históricos. Um processo é capaz quando seu desempenho atende os objetivos estabelecidos [Wheeler e Poling 1998].

A implementação do processo de medição, tipicamente, envolve o uso de apoio computacional, que pode variar de simples planilhas eletrônicas a sistemas mais complexos. Soluções computacionais são desenvolvidas de acordo com arquiteturas. Comumente representadas por diferentes visões (especificações e esquemas), arquiteturas correspondem a estruturas e direcionamentos a serem seguidos para a implementação de sistemas. Uma arquitetura pode se constituir em diferentes níveis de detalhes, podendo ser mais generalista, representando um conjunto de sistemas, ou mais específica e detalhada, representando um sistema específico [Muller 2013].

Para investigar as propostas de arquiteturas de medição presentes na literatura, foi realizado um mapeamento sistemático. Segundo Kitchenham e Charters (2007), um

mapeamento sistemático é um estudo exploratório em um tópico e visa identificar e analisar evidências sobre aquele tópico. O mapeamento foi realizado utilizando as bibliotecas digitais IEEE (ieeexplore.ieee.org) e Scopus (www.scopus.com). No total, foram analisadas 148 publicações, tendo sido encontradas 8 propostas de arquiteturas para medição de software. Embora as propostas encontradas forneçam apoio ao processo de medição de software, todas são soluções específicas (não visam ao reúso) e apenas duas abordam aspectos relacionados à medição em alta maturidade. Informações sobre o mapeamento sistemático, bem como descrições detalhadas das arquiteturas encontradas e análise de suas características encontram-se publicadas em [Maretto e Barcellos 2013b].

Os resultados do mapeamento sistemático forneceram evidências que motivaram o desenvolvimento de uma arquitetura de medição que possa ser reutilizada pelas organizações, auxiliando-as na definição de suas próprias arquiteturas. Antes de desenvolver a arquitetura propriamente dita, foi definida a abordagem a ser seguida para definição e avaliação da arquitetura de referência. Essa abordagem é apresentada na próxima seção.

3. Abordagem em Níveis para Definição de Arquiteturas de Referência

A Abordagem em Níveis para Definição de Arquiteturas de Referência (ANDAR) [Maretto e Barcellos 2013a] foi inspirada na Engenharia Dirigida por Modelos (MDE – *Model Driven Engineering*) [Fondement e Silaghi 2004] e na Arquitetura Dirigida por Modelos (MDA – *Model Driven Architecture*) [OMG 2003]. MDE advoga o uso de modelos com diferentes níveis de abstração e transformações de modelos de um nível para o outro. MDA usa modelos independentes de plataforma como base para gerar modelos para plataformas específicas, que são usados para implementar sistemas.

Seguindo os princípios de MDE, ANDAR é composta por níveis nos quais há modelos com diferentes níveis de abstração (do mais para o menos abstrato). Seguindo os princípios de MDA, ANDAR faz distinção entre modelos independentes de plataforma e modelos de plataformas específicas. Embora ANDAR seja inspirada em MDE e MDA, é importante ressaltar que ela não é uma proposta de MDE ou MDA. Assim, as transformações que levam do modelo de um nível para o de outro não fazem parte de seu escopo. A Figura 1 apresenta uma visão geral de ANDAR.

Figura 1. Visão geral de ANDAR.

O primeiro nível de ANDAR (**Metaconceituação**) contém modelos que descrevem objetos do mundo real, independentemente de domínio, os quais são representados por meio de ontologias de fundamentação. A função da ontologia de fundamentação é

prover a conceituação genérica que será usada como base para definir a conceituação do domínio para o qual a arquitetura de referência será desenvolvida.

O segundo nível (**Conceituação do Domínio**) diz respeito a modelos que descrevem um domínio específico. Nesse nível encontra-se uma ontologia de domínio, que é uma especificação independente de solução, que visa fazer uma clara e precisa descrição das entidades do domínio para os propósitos de comunicação, aprendizado e solução de problemas [Guizzardi 2007]. A função da ontologia de domínio é prover a conceituação do domínio no qual a arquitetura de referência será aplicada.

O terceiro nível (**Independente de Plataforma**) diz respeito a modelos desenvolvidos para o domínio sem se preocupar com aspectos da plataforma, ou seja, aspectos relacionados a tecnologias específicas. Nesse nível está a arquitetura de referência, que é definida com base na ontologia de referência do nível anterior e tem função de servir como base para a definição de arquiteturas específicas.

O quarto nível (**Plataforma Específica**) diz respeito a modelos derivados a partir dos modelos conceituais definidos no terceiro nível, porém, levando-se em consideração a utilização de tecnologias específicas. Como mostra a Figura 1, é possível a partir da arquitetura de referência definir diferentes arquiteturas de plataforma específicas.

Por fim, o último nível (**Aplicação**) diz respeito às aplicações construídas a partir de arquiteturas para plataformas específicas definidas no nível anterior. Embora aplicações não sejam modelos, elas foram inclusas na abordagem, pois podem ser utilizadas para avaliar a arquitetura de referência definida no Nível 3 [Muller 2013].

4. Arquitetura de Referência para Medição de Software (ARMS)

A Arquitetura de Referência para Medição de Software (ARMS) foi definida levando-se em consideração os seguintes requisitos: *(i)* a arquitetura deve conter os elementos essenciais a uma solução computacional para medição de software; *(ii)* a arquitetura deve apresentar os requisitos que devem ser atendidos por soluções computacionais de apoio ao processo de medição de software, considerando tanto medição tradicional quanto em alta maturidade; *(iii)* a arquitetura deve apresentar em detalhes a estrutura do repositório de medição; *(iv)* a arquitetura deve poder ser utilizada total ou parcialmente; *(v)* a arquitetura deve ser desenvolvida com base em uma ontologia de domínio bem fundamentada; *(vi)* a arquitetura deve ser independente de tecnologias.

ARMS foi desenvolvida seguindo-se ANDAR, descrita na seção anterior. No Nível 1 foi utilizada UFO (*Unified Foundational Ontology)* [Guizzardi 2005], uma ontologia de fundamentação que modela entidades do mundo real de forma genérica e tem sido desenvolvida com base em várias teorias de Ontologia Formal, Lógica Filosófica, Filosofia da Linguagem, Linguística e Psicologia Cognitiva. No Nível 2 foi utilizada a Ontologia de Referência para Medição de Software [Barcellos 2009; Barcellos et al. 2013], uma ontologia de domínio construída com base em UFO e que descreve o domínio de medição de software, incluindo aspectos relacionados tanto à medição tradicional quanto em alta maturidade. Ela é composta por seis subontologias: *Subontologia de Entidades Mensuráveis e Medidas,* que trata das entidades que podem ser submetidas à medição, das propriedades que podem ser medidas e da definição de medidas de software; *Subontologia de Objetivos de Medição*, que trata do alinhamento da medição de sof-

tware com os objetivos estratégicos; *Subontologia de Definição Operacional de Medidas*, que trata do detalhamento de aspectos relacionados à coleta e análise de medidas; *Subontologia de Medição de Software,* que trata da medição propriamente dita, ou seja, a coleta e armazenamento dos dados para as medidas; *Subontologia de Análise de Medição,* que trata da análise dos dados coletados para as medidas para obtenção das informações de apoio às decisões; e *Subontologia de Comportamento de Processos,* que trata da aplicação dos resultados da medição na análise do comportamento de processos. Como o domínio de medição de software é fortemente ligado aos domínios de organizações e processos de software, a Ontologia de Referência para Medição de Software reutiliza alguns conceitos das ontologias de processo de software e organizações de software descritas, respectivamente, em [Bringuente et al. 2011] e [Barcellos e Falbo 2009].

ARMS é baseada na conceituação provida pela Ontologia de Referência para Medição de Software, está situada no Nível 3 de ANDAR e possui três componentes, como mostra a Figura 2.

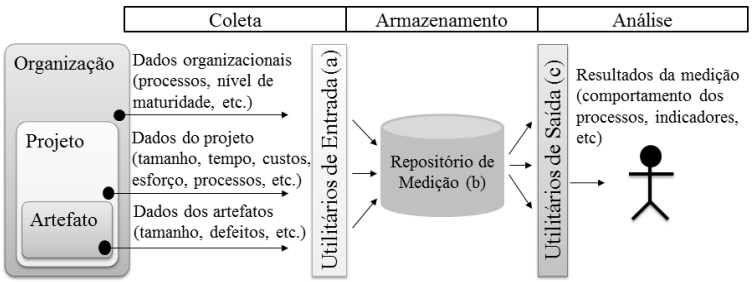

Figura 2. Visão Geral da Arquitetura de Referência para Medição de Software.

De acordo com ARMS, dados relacionados à organização, seus projetos e artefatos são capturados por utilitários de entrada (a) e armazenados em um repositório de medição (b). Os dados coletados são analisados e os resultados são apresentados para as partes interessadas através dos utilitários de saída (c).

Utilitários de Entrada é o componente responsável pela captura dos dados da medição. É descrito através de um conjunto de requisitos funcionais relacionados à medição de software tradicional e em alta maturidade. Cada requisito tem uma descrição detalhada. Em uma solução computacional para medição de software, os utilitários de entrada devem prover um conjunto de funcionalidades que atendam a esses requisitos. O *Repositório de Medição* é o componente responsável pelo armazenamento dos dados da medição. É descrito por meio de diagramas de pacotes e modelos de classes UML (*Unified Modeling Language)*, descrições detalhadas desses modelos e restrições de integridade. Por fim *Utilitários de Saída* é o componente responsável pela análise dos dados e apresentação dos resultados para apoio à tomada de decisão. Assim como o componente Utilitários de Entrada, é descrito por meio de requisitos funcionais.

A definição sobre quais componentes deveriam fazer parte da arquitetura de referência foi tomada levando-se em consideração o que se observou durante a investigação da literatura. Independente do tipo de tecnologia utilizada em propostas que descrevem soluções e arquiteturas para medição de software, a maioria inclui componentes com funções para coletar, armazenar e apresentar os dados da medição. De fato, isso era esperado, uma vez que essas são as atividades básicas da medição. Também notou-se que várias propostas consideram o armazenamento de dados oriundos de diversas fon-

tes, sendo esses dados coletados por meio de diferentes aplicativos. De forma similar, algumas propostas consideram que diferentes aplicações podem fazer uso dos dados armazenados para apoiarem a análise dos dados e proverem os resultados aos tomadores de decisão. Assim, optou-se por chamar os componentes responsáveis pela captura e apresentação dos dados de *utilitários*, a fim de dar a ideia de que esses componentes podem ser implementados por meio de uma ou mais ferramentas computacionais. A seguir, alguns detalhes sobre os componentes de ARMS são apresentados.

4.1. Utilitários de Entrada e Saída: Requisitos

A identificação dos requisitos dos Utilitários de Entrada e Saída baseou-se principalmente na Ontologia de Referência para Medição de Software [Barcellos 2009; Barcellos et al. 2013], em [Rocha et al. 2012] e nos resultados do mapeamento sistemático [Maretto e Barcellos 2013b]. Foram identificados 21 requisitos, sendo 19 relacionados ao componente Utilitários de Entrada e 2 ao componente Utilitários de Saída. Os requisitos R1 a R11 e R13 são relacionados à medição tradicional. Os requisitos R14 a R19 e R21 dizem respeito à medição em alta maturidade. Os requisitos R12 e R20, por sua vez, estão relacionados a ambas.

Tabela 1. Requisitos dos Utilitários.

Id	Requisito
R1	Deve ser possível caracterizar projetos e identificar projetos similares.
R2	Deve ser possível registrar a definição dos processos, suas versões e identificar as versões utilizadas nos projetos.
R3	Deve ser possível registrar as entidades que podem ser medidas, seus tipos e elementos que podem ser quantificados.
R4	Deve ser possível registrar objetivos estratégicos relevantes à medição.
R5	Deve ser possível registrar objetivos de medição e identificar seus tipos e suas relações com objetivos estratégicos.
R6	Deve ser possível registrar necessidades de informação a partir de objetivos de medição.
R7	Deve ser possível registrar medidas.
R8	Deve ser possível estabelecer definições operacionais para medidas.
R9	Deve ser possível identificar medidas correlatas.
R10	Deve ser possível registrar planos de medição da organização e do projeto.
R11	Deve ser possível registrar medições.
R12	Deve ser possível realizar e registrar análises de medições.
R13	Deve ser possível registrar as pessoas envolvidas em atividades de medição e os papéis por elas desempenhados.
R14	Deve ser possível analisar o comportamento dos processos e classificá-los em estáveis ou capazes.
R15	Deve ser possível registrar *baselines* de desempenho para os processos estáveis.
R16	Deve ser possível registrar o desempenho especificado para os processos.
R17	Deve ser possível determinar a capacidade dos processos.
R18	Deve permitir analisar o comportamento dos processos executados nos projetos em relação às *baselines* estabelecidas para o processo no âmbito organizacional.
R19	Deve ser possível registrar modelos de desempenho.
R20	Deve ser possível apresentar resultados da medição por meio de relatórios, gráficos ou consultas.
R21	Deve ser possível apresentar resultados da análise de comportamento de processos por meio de relatórios, gráficos ou consultas.

4.2. Repositório de Medição

O repositório de medição é o componente responsável pelo armazenamento dos dados da medição. É definido através de diagramas de pacotes, modelos de classes e restrições

que descrevem a estrutura necessária a um repositório para atender as necessidades relacionadas à execução do processo de medição. Os modelos de classes do repositório de medição foram elaborados baseados, principalmente, na Ontologia de Referência para Medição de Software [Barcellos 2009]. Para a elaboração dos modelos de classes, também foram considerados os requisitos que descrevem os Utilitários de Entrada e Saída.

De forma análoga à Ontologia de Referência para Medição de Software, a qual é dividida em seis subontologias, o repositório está organizado em pacotes. Cada subontologia da Ontologia de Referência para Medição de Software foi mapeada para um pacote. Para contemplar aspectos relacionados ao Plano de Medição (requisito R10), foi criado o pacote Plano de Medição, que trata do resultado do planejamento da medição para a organização e para seus projetos, o qual é consolidado no Plano de Medição, que inclui quais objetivos, entidades, medidas e definições operacionais serão consideradas na medição de software na organização e em seus projetos. Para tratar a caracterização de projetos (requisito R1), foi criado o pacote Caracterização de Projetos. A Figura 3 apresenta o diagrama de pacotes de ARMS. A estrutura completa do repositório de medição é muito extensa. Por limitações de espaço, neste artigo serão apresentados apenas alguns fragmentos. A descrição completa de ARMS pode ser encontrada em [Maretto 2013].

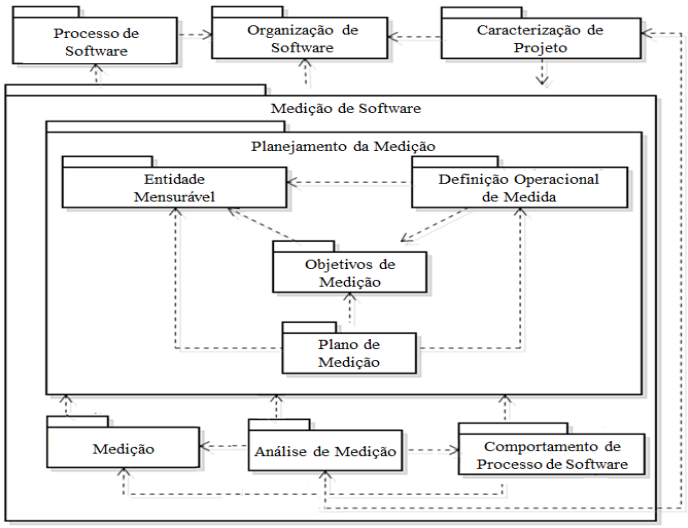

Figura 3. Diagrama de Pacotes do Repositório de Medição.

As figuras 4 e 5 apresentam fragmentos dos diagramas de classes do repositório de medição. Nas figuras, o acrônimo que precede o nome das classes indica o pacote de origem da classe, sendo: PS – Processo de Software, OS – Organização de Software, CP – Caracterização de Projeto, EM – Entidade Mensurável, DOM – Definição Operacional de Medida, OM – Objetivos de Medição, PM – Plano de Medição, M – Medição, AM – Análise de Medição, CPS – Comportamento de Processo de Software. A Figura 4 apresenta o diagrama de classes do pacote Plano de Medição. As classes em cinza pertencem a esse pacote. Após a figura, uma breve descrição é apresentada. Na descrição **negrito** indica a primeira ocorrência do nome da classe ou papel, sublinhado indica atributos das classes e *itálico* indica possíveis instâncias.

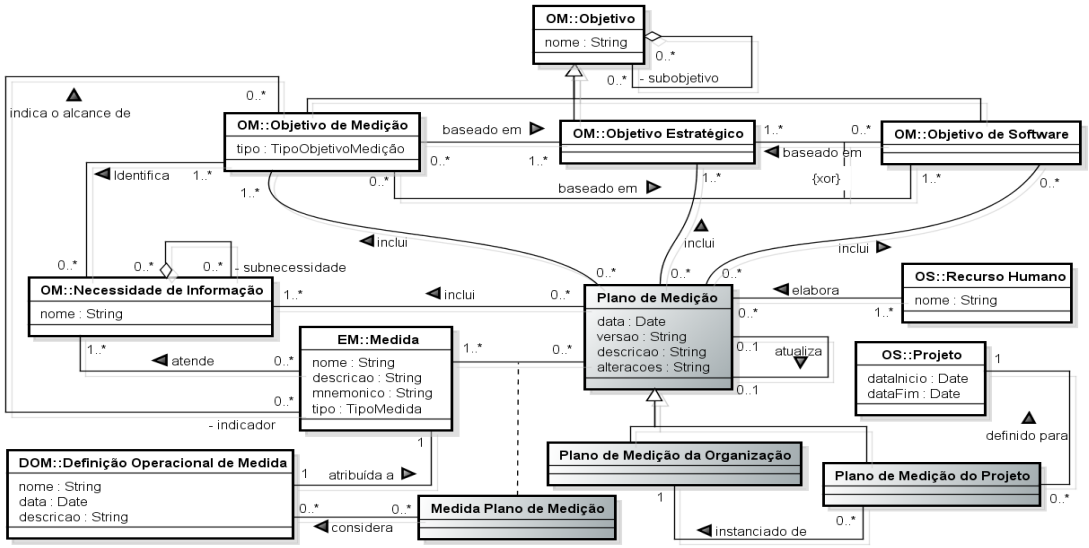

Figura 4. Diagrama de classes do pacote Plano de Medição.

Um **Plano de Medição** é o resultado da atividade de planejamento da medição e é elaborado por um ou mais recursos humanos (**Recurso Humano**). Ele reúne todas as informações relevantes para que a medição possa ser realizada. Um **Plano de Medição da Organização** é um plano que define o padrão para a realização da medição em uma organização. A partir desse plano são instanciados **Planos de Medição do Projeto**, que são planos estabelecidos para cada projeto da organização. Um Plano de Medição inclui **Objetivos Estratégicos** que são relevantes para a medição de software, **Objetivos de Software** definidos com base nos objetivos estratégicos e **Objetivos de Medição** definidos com base nos anteriores. Inclui, também, **Necessidades de Informação** identificadas a partir dos objetivos de medição e as **Medidas** necessárias para atendê-las. Para cada medida presente em um Plano de Medição (**Medida Plano de Medição**) deve ser indicada a **Definição Operacional de Medida** a ser usada. A definição operacional de medida contém informação detalhada sobre a coleta e análise da medida, tais como os procedimentos de medição e análise e os papéis que devem ser desempenhados pelos responsáveis pela coleta e análise da medida (ex.: gerente de projeto e programador). As classes que tratam as informações inclusas em uma definição operacional não estão inclusas na Figura 4.

Como exemplos de instâncias para as classes mostradas na Figura 4, têm-se: o *Plano de Medição do Projeto P*, elaborado por *João da Silva* e instanciado a partir do *Plano de Medição da Organização Org.* O *Plano de Medição do Projeto P* inclui, dentre outros, o objetivo estratégico *obter fidelização dos clientes atuais*, o objetivo de software *aumentar o nível de satisfação dos clientes com os projetos* e o objetivo de medição *diminuir o tempo médio de correção de defeitos reportados pelo cliente.* Associada a esse objetivo, o plano inclui a necessidade de informação *conhecer o tempo médio de correção de defeitos reportados pelo cliente*, e define as medidas *número de defeitos reportados pelo cliente e corrigidos, tempo despendido na correção dos defeitos* e *tempo médio de correção dos defeitos* como as medidas necessárias para atender essa necessidade de informação. Para cada uma dessas medidas, o plano inclui uma definição operacional.

Para cada diagrama de classes do repositório de medição, foram identificadas as restrições relacionadas. Como exemplo de restrição para o modelo mostrado na Figura 4: se um plano de medição *pm* inclui um objetivo de medição *om*, então *pm* deve incluir um objetivo estratégico ou objetivo de software no qual *om* se baseia.

A Figura 5 apresenta o diagrama de classes para o pacote Comportamento de Processo de Software.

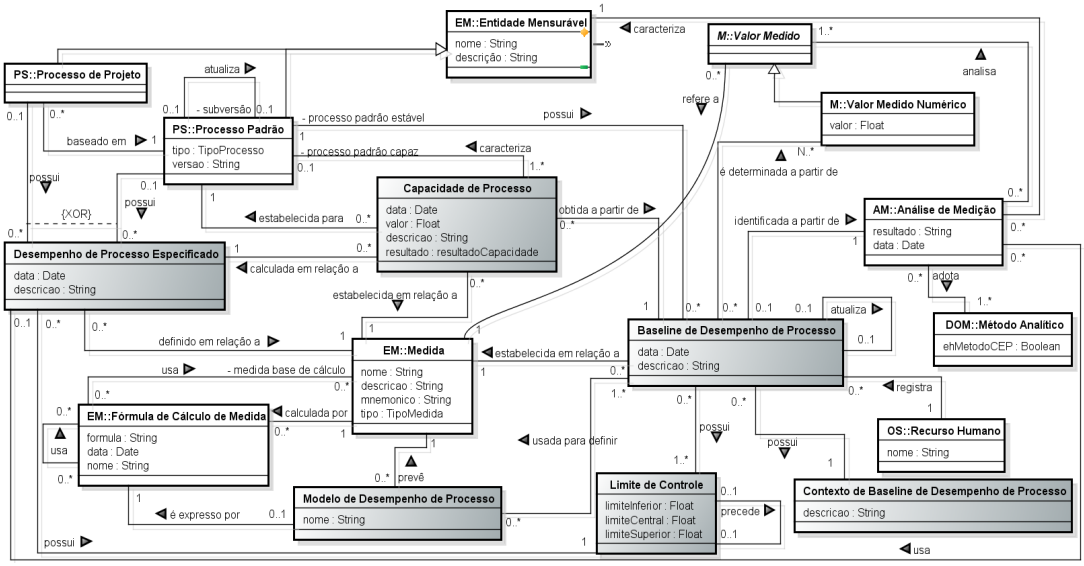

Figura 5. Diagrama de classes do pacote Comportamento de Processo de Software.

Uma **Análise de Medição** adota um **Método Analítico** (ex.: gráfico de barras, histograma) para analisar **Valores Medidos** e obter resultados de análise (resultado) em relação a uma medida. Em uma Análise de Medição que adota um método analítico do controle estatístico (ehMetodoCEP = true), tais como gráficos de controle XmR e mXmR, é possível definir uma ***Baseline* de Desempenho de Processo** para um **Processo de Software Padrão**, em relação a uma Medida. Um processo de software padrão é a descrição de um tipo de processo de software, definido no contexto de uma organização (ex.: a descrição do *processo de software padrão Engenharia de Requisitos da organização Org*). Processos padrão são usados como base para definir processos que são usados nos projetos da organização (**Processo do Projeto**).

Uma *Baseline* de Desempenho de Processo tem **Limites de Controle** (limiteSuperior, limiteCentral, limiteInferior) é determinada a partir de vários **Valores Medidos Numéricos**. Na literatura não há consenso sobre quantos valores são necessários para definir ou atualizar uma *baseline* de desempenho. Alguns autores sugerem 20 valores para estabelecer uma *baseline* e, pelo menos, 8 para atualizá-la [Rocha et al. 2012]. No entanto, isso não é uma regra. Por essa razão, decidiu-se não definir a quantidade mínima de valores que devem ser considerados para se estabelecer uma *baseline*. A cardinalidade mínima entre *Baseline* de Desempenho de Processo e Valor Medido Numérico é representada no modelo por N, devendo ser determinada durante a definição da arquitetura para plataforma específica. Por exemplo, N pode ser definido como 8 e, quando se tratar da primeira *baseline* definida para um processo, pode haver uma restrição que exija, pelo menos, 20 valores. Essas restrições podem ser implementadas em uma aplicação como alertas, que não impedem a definição de uma *baseline,* mas que informe ao

usuário que uma quantidade de valores menor do que a sugerida pode impactar na qualidade das *baseline* obtidas.

Quando um processo tem uma *baseline* de desempenho estabelecida ele é dito um **processo padrão estável**. Por exemplo, a análise de valores medidos para a medida *taxa de alteração de requisitos*, relacionada ao processo padrão de *Gerência de Requisitos*, utilizando-se o gráfico de controle XmR, pode levar à definição de uma *baseline* de desempenho de processo com os limites 0.1, 0.175 e 0.25. Uma *baseline* de desempenho de processo é estabelecida por um Recurso Humano em um **Contexto de *Baseline* de Desempenho de Processo** (por exemplo: *primeira baseline de desempenho estabelecida para o processo padrão de Gerência de Requisitos, tendo sido o processo padrão executado em 6 projetos pequenos, cujas equipes foram compostas pelos mesmos recursos humanos, sob condições usuais)*.

A *baseline* de desempenho de processo de um processo de software padrão pode ser utilizada em uma análise de medição que analisa o comportamento de um processo de projeto baseado naquele processo de software padrão. Por exemplo, a *baseline* de desempenho estabelecida para o processo padrão de *Gerência de Requisitos* apresentada no exemplo do parágrafo anterior poderia ser utilizada em uma análise de medição para verificar se o desempenho do processo de *Gerência de Requisitos do Projeto P* está de acordo com o desempenho esperado para ele, ou seja, com o desempenho descrito pela *baseline* de desempenho.

Baselines de desempenho de processo podem ser usadas na definição de **Modelos de Desempenho de Processo.** Um Modelo de Desempenho de Processo é expresso por uma **Fórmula de Cálculo de Medida** que prevê o valor de uma medida derivada (tipo = derivada) a partir de outras. O modelo de desempenho de processo é obtido a partir de valores medidos usados na definição de *baselines* de desempenho de processo. Um modelo expresso pela fórmula que relaciona esforço e tamanho, obtido a partir de valores medidos que foram usados para se estabelecer uma *baseline* de desempenho em relação à medida produtividade é um exemplo de modelo de desempenho de processo. Modelos de desempenho podem ajudar gerentes a definirem planos realísticos. Por exemplo, um gerente poderia usar um modelo de desempenho de processo que relaciona esforço e tamanho para predizer o esforço necessário em um projeto a partir do seu tamanho.

Desempenho de Processo Especificado descreve o intervalo de resultados que se espera que um processo (Processo Padrão ou Processo de Projeto) alcance considerando uma medida específica, sendo definido por limites de controle (usualmente, limites superior e inferior). Por exemplo, o processo padrão de *Gerência de Requisitos* poderia ter um desempenho de processo especificado definido em relação à medida *taxa de alteração de requisitos*, dado pelos limites inferior e superior *0* e *0.25*, respectivamente. Isso significa que é desejado que, no máximo, 25% dos requisitos sofram alterações ao longo dos projetos. Quando é esperado que um processo de projeto apresente desempenho diferente daquele especificado para seu processo padrão, é estabelecido um desempenho de processo especificado para o processo de projeto. Considerando o exemplo anterior, se em um projeto específico fosse definido que não mais do que 20% dos seus requisitos poderiam ser alterados, seria estabelecido um desempenho de processo especificado, definido em relação à medida *taxa de alteração de requisitos*, dado pelos limi-

tes inferior e superior *0* e *0.2*, respectivamente. Uma vez que é possível definir processo de processo especificado para processos padrão e processos de projeto, uma análise de medição que analisa o comportamento de um processo de projeto pode considerar o desempenho especificado estabelecido pelo processo padrão no qual o processo do projeto se baseia ou no desempenho especificado estabelecido para o processo do projeto sendo analisado.

A **Capacidade de Processo** determina a habilidade de um processo padrão estável atender a um desempenho de processo para ele especificado. Nesse sentido, ela é calculada a partir de uma *baseline* de desempenho de processo e de um desempenho de processo especificado, e é estabelecida em relação a uma medida, que deve ser a mesma medida usada na *baseline* de desempenho de processo e no desempenho de processo especificado. Quando a capacidade de um processo mostra que ele é capaz de atender ao desempenho de processo considerado, tem-se um **processo padrão capaz.**

A análise da capacidade de um processo é feita usando-se um índice de capacidade dado pela razão entre as amplitudes do desempenho de processo especificado e a *baseline* de desempenho de processo. Quando o índice de capacidade é igual ou maior que 1 e os limites da *baseline* de desempenho de processo são internos aos limites do desempenho de processo especificado, o processo é capaz. Por exemplo, para os valores da *baseline* de desempenho e do desempenho especificado dados como exemplos anteriormente, o processo padrão de *Gerência de Requisitos* é um processo capaz para a medida *taxa de alteração de requisitos*, pois, aplicando-se o procedimento de determinação de capacidade de processo, tem-se como resultado o valor 1.04, que indica que o processo é capaz.

5. Avaliando ARMS

Com o objetivo de avaliar ARMS, ela foi usada como base para a definição de uma arquitetura de plataforma específica (Nível 4 de ANDAR), que, por sua vez, foi usada para o desenvolvimento de uma ferramenta (Nível 5 de ANDAR). A definição da arquitetura específica e implementação da ferramenta (MedCEP) serviram como prova de conceito para ARMS, mostrando que é possível definir uma arquitetura específica a partir de ARMS e que, a partir dessa arquitetura, é possível implementar uma ferramenta. Segundo Oates (2006), uma prova de conceito mostra que uma proposta é exequível, porém não permite afirmar se ela funciona em um contexto real.

Na arquitetura de plataforma específica, os utilitários de entrada e saída foram implementados em um único aplicativo (a ferramenta MedCEP) e o repositório de medição foi tratado em um banco de dados relacional. As tecnologias usadas foram: o OpenXava (openxava.org), que é um framework Java com AJAX (*Asynchronous Javascript and XML*), a biblioteca Javascript RGraph (www.rgraph.net), o gestor de conteúdo e aplicação LifeRay (www.liferay.com) e o banco de dados Postgres SQL (www.postgresql.org). Na Figura 6 é apresentada a tela de MedCEP usada para apoiar a análise de comportamento de processos. Na figura são apresentados dados coletados para a medida taxa de detecção de defeitos durante a execução do processo padrão de Inspeção em alguns projetos. Os dados foram plotados em um gráfico de controle. O processo é estável (suas variações então dentro dos limites de controle), mas não é capaz (o índice de capacidade é 0.802, que é menor que 1).

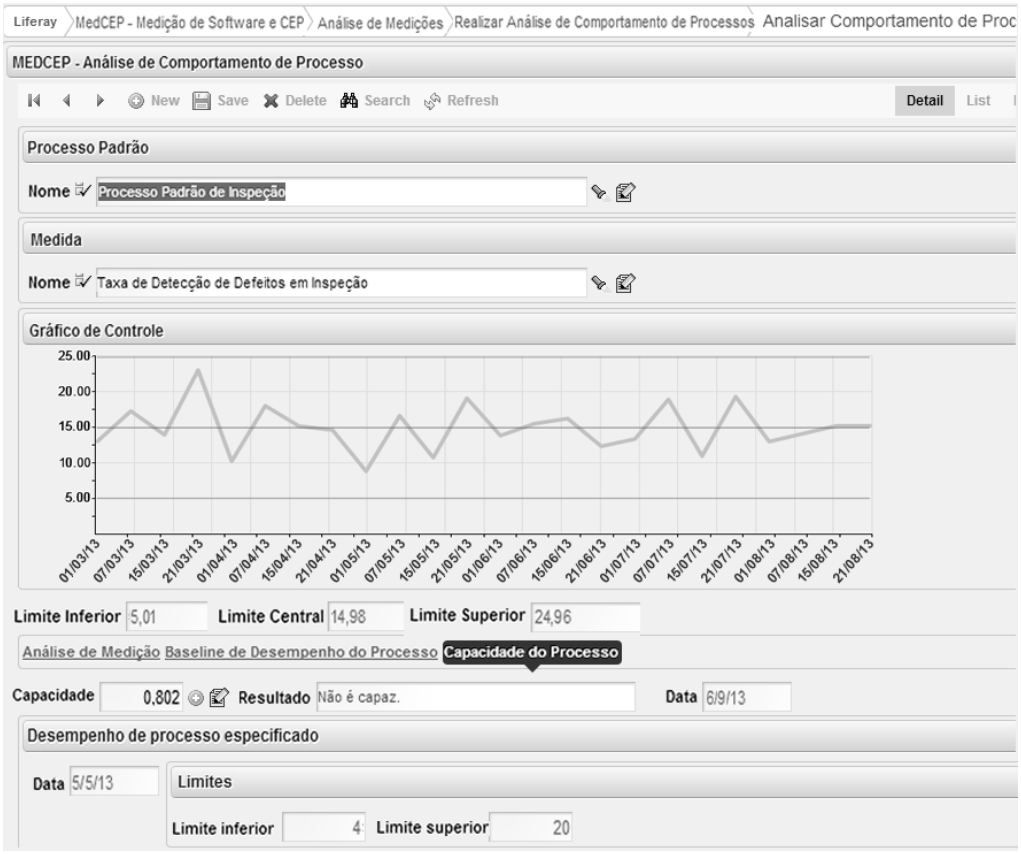

Figura 6. Tela de MedCEP para análise de comportamento de processos.

Segundo [Muller 2013], aplicações desenvolvidas a partir de uma arquitetura de referência podem ser utilizadas para avaliar a arquitetura, no sentido que permitem avaliar se ela atende às necessidades do domínio para a qual foi proposta. Seguindo essa abordagem, a ferramenta MedCEP foi utilizada para, indiretamente, avaliar ARMS. O objetivo foi avaliar se MedCEP atende aos requisitos presentes em ARMS (ou seja, se MedCEP reflete a arquitetura de referência e, dessa forma, pode ser usada para avaliar ARMS) e se ela é capaz de apoiar adequadamente a realização do processo de medição de software, o que seria uma evidência de que a arquitetura de referência trata adequadamente os aspectos necessários ao processo de medição. Inicialmente, um especialista em medição de software tradicional e em alta maturidade utilizou a ferramenta livremente. Depois disso, foi realizado um estudo experimental cujos participantes foram alunos de mestrado e doutorado. Os resultados indicaram que ARMS trata adequadamente os aspectos necessários à realização do processo de medição de software. Informações detalhadas sobre o estudo experimental (planejamento, execução, resultados obtidos e ameaças à validade) podem ser encontrados em [Maretto 2013]. É importante ressaltar que os resultados obtidos até o momento são iniciais e devem ser complementados com novos estudos envolvendo profissionais da indústria.

6. Considerações Finais

Existem na literatura algumas propostas de arquiteturas para medição de software. No entanto, usualmente, não são apresentados detalhes suficientes para sua reutilização. Na maioria das vezes, tem-se apenas uma especificação superficial da arquitetura, onde, por

exemplo, o modelo de dados do repositório não é apresentado. Além disso, poucas são as propostas que abordam medição em alta maturidade. Considerando essa lacuna, este artigo apresentou uma Arquitetura de Referência para Medição de Software, que considera medição tradicional e em alta maturidade e tem o propósito de servir como base para organizações desenvolverem suas próprias arquiteturas e soluções computacionais.

Destacam-se como contribuições do trabalho: *(i)* a Arquitetura de Referência para Medição de Software [Maretto 2013], *(ii)* o mapeamento sistemático sobre arquiteturas de medição de software [Maretto e Barcellos 2013b]; *(iii)* a ferramenta MedCEP; e *(iv)* a Abordagem em Níveis para Definição de Arquiteturas de Referência [Maretto e Barcellos 2013a].

Vale reforçar que, apesar de terem sido realizadas algumas avaliações iniciais de ARMS, novas avaliações são necessárias, incluindo novos estudos experimentais com profissionais da indústria com conhecimento e experiência prática em medição de software e CEP, e, também, pelo menos um estudo de caso na indústria.

Referências

Barcellos, M. P. (2009). Uma Estratégia para Medição de Software e Avaliação de Bases de Medidas para Controle Estatístico de Processos de Software em Organizações de Alta Maturidade. Tese de Doutorado, UFRJ, Rio de Janeiro, BR.

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2013). A strategy for preparing software organizations for statistical process control. *Journal of the Brazilian Computer Society*.

Barcellos, M. P., Falbo, R. A., Dalmoro, R. (2010a). A Well-Founded Software Measurement Ontology. In *6th International Conference on Formal Ontology in Information Systems (FOIS 2010)*, Toronto, CA, p. 213-226.

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2010b). Establishing a Well-founded Conceptualization about Software Measurement in High Maturity Levels. In *7th International Conference on the Quality of Information and Communications Technology (QUATIC 2010)*, Oporto, PT, p. 467-472.

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2010c). A Well-founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations. In *IEEE 5th Joint VORTE-MOST Workshop*, Vitória, BR, p. 253-262.

Barcellos, M. P. and Falbo, R. A. (2009). Using a foundational ontology for reengineering a software enterprise ontology. In *ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling - Challenging Perspectives*, Springer-Verlag. Berlin, DE, p. 179-188.

Bass, L., Belady, L., Brown, A., Freeman, P., Isensee, S., Kazman, R., Krasner, H., Musa, J., Pfleeger, S., Vredenburg, K., Wasserman, T. (1999) "Constructing Superior Software", Software Quality Institute Series, Macmillan Technical Publishing.

Bringuente, A. C. O, Falbo, R. A., Guizzardi, G. (2011). Using a foundational ontology for reengineering a software process ontology. In *XXVI Brazilian Symposium on Data Base*, Florianópolis, BR, p. 1-16.

De Lucia, A., Pompella, E., Stefanucci, S. (2003). Assessing the Maintenance Process of a Software Organization: an Empirical Analysis of a Large Industrial Project. *The Journal of Systems and Software*, v. 65, p. 87–103.

Dumke, R., Ebert, C. (2010). *Software Measurement: Establish – Extract – Evaluate – Execute*. Springer, New York, US.

Fondement, F., Silaghi, R. (2004). Defining model driven engineering processes. In *Third International Workshop in Software Model Engineering (WiSME)*, Lisbon, PT, p. 1-11.

Guarino, N. (1998). Formal Ontology and Information Systems. In *International Conference in Formal Ontology and Information Systems - FOIS'98*, Trento, IT, p. 3-15.

Guizzardi, G. (2005) "Ontological Foundations for Structural Conceptual Models", Universal Press, The Netherlands.

Guizzardi, G. (2007). On Ontology, ontologies, Conceptualizations, Modeling Languages and (Meta)Models. In *Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference*, Amsterdam, NL, p. 18-39.

ISO/IEC (2008). ISO/IEC 12207 – Systems and Software Engineering – Software Life Cycle Processes. International Organization for Standardization and the International Electrotechnical Commission.

Kitchenham, B., Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University, Staffordshire, UK.

Maretto, C. X. (2013). Uma Arquitetura de Referência para Medição de Software. Dissertação de Mestrado, UFES, Vitória, BR.

Maretto, C. X., Barcellos, M. P. (2013a). A Levels-based Approach for Defining Software Measurement Architectures. *Clei Electronic Journal*, v. 14, n. 3, p. 27.

Maretto, C. X., Barcellos, M. P. (2013b). Software Measurement Architectures: A Mapping Study. In *10th ESELAW: Experimental Software Engineering Latin American Workshop (ESELAW 2013)*. Montevideo, UR, p. 20–33

Mcgarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. and Hall, F. (2002) "Practical Software Measurement: Objective Information for Decision Makers", Addison Wesley, Boston, US.

Muller, G. (2013) "A reference architecture primer", Buskerud University College, Kongsberg, NO, p. 1-21.

Nakagawa, E.Y., Barbosa, E.F., Maldonado, J.C. (2009). Exploring ontologies to support the establishment of reference architectures: An example on software testing. In *Joint Working IEEE/IFIP Conference on Software Architecture 2009 & European Conference on Software Architecture 2009*, Cambridge, UK, p. 249-252.

Oates, B. J. (2006) "Researching Information Systems and Computing", SAGE Publications.

OMG. (2003). "MDA Guide Version 1.0.1". http://www.enterprise-architecture.info. Jun. 2012.

Rocha, A.R., Santos, G. S., Barcellos, M. P. (2012) "Medição de Software e Controle Estatístico de Processos", Série de Livros PBQP Software, Secretaria de Política de Informática do Ministério da Ciência, Tecnologia e Inovação.

SEI. (2010). CMMI for Development, Version 1.3, Technical Report CMU/SEI-2010-TR-033. Pittsburgh, US.

SOFTEX. (2012). "MPS.BR: Melhoria de Processo do Software Brasileiro - Guia Geral." http://www.softex.br/mpsbr. Fev. 2012.

Wheeler, D. J., Poling, R. S. (1998) "Building Continual Improvement: A Guide for Business", SPC Press, Knoxville, US.

Zachman, J. (1987). A framework for information systems architecture. *IBM Systems Journal*. p. 276–292.