UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ciro Xavier Maretto

Uma Arquitetura de Referência para Medição de Software

VITÓRIA
2013

Ciro Xavier Maretto

Uma Arquitetura de Referência para Medição de Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador(a): Monalessa Perini Barcellos

VITÓRIA
2013

Ciro Xavier Maretto

Uma Arquitetura de Referência para Medição de Software

COMISSÃO EXAMINADORA

Prof. Monalessa Perini Barcellos, D. Sc.
Universidade Federal do Espírito Santo (UFES)

Prof. Ricardo de Almeida Falbo, D. Sc.
Universidade Federal do Espírito Santo (UFES)

Prof. Gleison dos Santos Souza, D. Sc.
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Vitória, 19 de Agosto de 2013

A Deus, familiares, professores e amigos que contribuíram para este trabalho.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter permitido que eu pudesse chegar até aqui, e ter me protegido e abençoado durante toda essa jornada.

À minha mãe, meu pai e minha irmã, pois eles me deram todo o suporte necessário e foram grandes incentivadores para minha dedicação aos estudos.

A todos os familiares, que, direta ou indiretamente, contribuíram aos meus estudos. Em especial, o João Marcos, primo e companheiro de estudos em computação desde a graduação.

Aos professores do mestrado que tiveram um importante papel para a formação do meu conhecimento e discussão sobre este trabalho.

A Monalessa, professora e orientadora do mestrado, que foi paciente e se esforçou junto comigo no estudo. Certamente posso dizer que sem ela este trabalho não seria possível.

Aos colegas Antognoni, Saymon, Vinícius, Fabiano e todos os outros que cooperaram e me incentivaram. Aprendi muito caminhando ao lado deles.

Ao corpo administrativo do PPGI pelo bom atendimento e apoio.

Aos professores Ricardo Falbo e Gleison Santos, por aceitarem participar da banca e por suas valiosas contribuições.

Minha gratidão a todos os que de alguma forma influenciaram para que esse trabalho se tornasse realidade.

RESUMO

Durante a execução dos projetos de software, muitos dados precisam ser coletados, armazenados e analisados para apoiar decisões tanto no contexto dos projetos quanto da organização. A medição de software é prática fundamental para o gerenciamento de projetos e para a melhoria de processos e está presente nos principais modelos e padrões de melhoria de processos de software, como a ISO/IEC 12207 (ISO/IEC, 2008), o CMMI (*Capability Maturity Model Integration*) (SEI, 2010) e o MR-MPS-SW (Modelo de Referência para Melhoria de Processo de Software Brasileiro) (SOFTEX, 2012). Em modelos que tratam a melhoria de processos em níveis, tais como o CMMI e o MR-MPS-SW, a medição é introduzida nos níveis iniciais (nível 2 do CMMI e nível F do MR-MPS-SW) e evolui à medida que a maturidade dos processos aumenta. Na alta maturidade, caracterizada nos níveis 4 e 5 do CMMI e B e A do MR-MPS-SW, a medição evolui, sendo necessário realizar o controle estatístico dos processos (CEP). Para realizar a medição de software de forma efetiva, é necessária uma infraestrutura computacional para apoiar a coleta, o armazenamento e a análise de medidas. Nesse sentido, este trabalho propõe uma arquitetura de referência para medição de software, que considera tanto a medição tradicional quanto em alta maturidade. A arquitetura proposta é independente de plataforma e foi definida com base na Ontologia de Referência para Medição de Software (BARCELLOS, 2009). Ela pode ser utilizada como base para a definição de arquiteturas específicas para soluções computacionais que apoiem o processo de medição de software. Como prova de conceito, a arquitetura de referência proposta foi utilizada como base para a definição de uma arquitetura específica e uma ferramenta. Além disso, como uma avaliação preliminar do trabalho, um estudo experimental foi realizado.

Palavras-chave: Medição de Software, Controle Estatístico de Processos, Arquitetura de Referência, Alta Maturidade.

ABSTRACT

During the execution of software projects, it is necessary to collect, store and analyze data in order to support project and organizational making decisions. Software measurement is a fundamental practice for project management and process improvement. It is present in the main models and standards that address software process improvement, such as ISO/IEC 12207 (ISO/IEC, 2008), CMMI (Capability Maturity Model Integration) (SEI, 2010) and MR-MPS-SW (Reference Model for Process Improvement of Brazilian Software) (SOFTEX, 2012). In models that address software process improvement in levels, such as CMMI and MR-MPS-SW, measurement starts in the initial levels (CMMI level 2 and MR-MPS-SW level F) and evolves as the maturity level increases. In the high maturity, characterized in CMMI levels 4 and 5 and MR-MPS-SW level B and A, measurement includes carrying out statistic process control (SPC). In order to perform the measurement process in an effective way, a computational infrastructure able to support measurement collection, storage and analysis is necessary. In this sense, this work proposes a reference architecture for software measurement that considers both traditional and high maturity measurement. The proposed architecture is platform independent and has been defined based on the Reference Ontology for Software Measurement (BARCELLOS, 2009). It can be used as a basis for defining specific architectures for computational solutions that support the software measurement process. As a proof of concept, the reference architecture was used as a basis for a specific architecture and a tool. In addition, aiming at evaluating the reference architecture proposed, an experimental study was carried out and some preliminary results were obtained.

Keywords: Software Measurement, Statistical Process Control, Reference Architecture, High Maturity.

SUMÁRIO

Capítulo 1

Introdução

Este capítulo apresenta o contexto, motivação e objetivos do trabalho, bem como o método de pesquisa adotado e a organização do texto desta dissertação.

1.1 Contexto

Medição de software é uma avaliação quantitativa de qualquer aspecto dos 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). O elemento básico da medição, que propicia a análise quantitativa, são as medidas. Elas caracterizam, em termos quantitativos, alguma propriedade de um objeto da Engenharia de Software (BASILI e ROMBACH, 1994). O objetivo mais importante de sua aplicação é prover informação quantitativa para apoiar a tomada de decisões (FENTON e NEIL, 2000).

A medição de software é utilizada nas organizações de diversas maneiras. Por exemplo, no contexto do gerenciamento de projetos de software, a medição ajuda a elaborar planos realísticos, monitorar progressos, identificar problemas e justificar decisões (McGARRY *et al.*, 2002). Ao longo dos projetos, os dados coletados devem ser armazenados em um repositório de medidas para que sejam utilizados pelas organizações na gerência de seus projetos e na melhoria de seus processos (BARCELLOS, 2009).

O CMMI (*Capability Maturity Model Integration*) (SEI, 2010) e o MR-MPS-SW (Modelo de Referência MPS para Software) (SOFTEX, 2012) organizam os processos de software e orientam a melhoria de processos em níveis de maturidade. 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 níveis A e B do MR-MPS-SW) ocorre o controle estatístico de processos, que permite conhecer o comportamento dos processos a fim de prever seu desempenho em suas próximas execuções e verificar sua capacidade de alcançar os objetivos estabelecidos. O controle estatístico de processos requer atenção extra em alguns aspectos da medição dentre eles o armazenamento dos dados.

Construir e alimentar um repositório de medidas de forma eficiente e que considere as necessidades organizacionais de acordo com seu nível de maturidade não é uma tarefa

simples. É comum organizações começarem a registrar suas medidas em planilhas e, às vezes, em alguns sistemas, havendo pouca ou nenhuma integração entre eles (DUMKE e EBERT, 2010). Nos níveis iniciais de maturidade, as planilhas parecem atender às necessidades organizacionais, 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, 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, que é formada por diferentes componentes, pode ser definida por meio de uma arquitetura. De acordo com Zachman (1987), uma arquitetura pode ser entendida como uma estrutura lógica na qual os componentes são organizados e integrados. No contexto de medição de software, uma arquitetura deve considerar aspectos relacionados à coleta, armazenamento e análise de medições. Em uma arquitetura de medição, um dos principais componentes é o repositório de medição. Segundo Bernstein (1997), um repositório pode ser definido como um banco de dados compartilhado de informações sobre artefatos de engenharia. Em uma arquitetura de medição, o repositório de medição armazena os dados da medição, que não se restringem aos dados coletados para as medidas, e age como provedor de dados para as análises.

Este trabalho encontra-se no contexto de organizações de software que desejam realizar medição de software e visam à alta maturidade, mas enfrentam dificuldades para desenvolver uma infraestrutura computacional capaz de apoiar o processo de medição.

1.2 Motivação

Medição é uma atividade básica da Engenharia de Software, no entanto, relatos da literatura mostram que ainda há diversos problemas relacionados à realização de medição de software pelas organizações. Vários desses problemas dizem respeito à medição tradicional, isto é, à medição realizada nos níveis iniciais de modelos de maturidade como o MR-MPS-SW e o CMMI. Esses problemas tornam-se mais evidentes quando as organizações dão início às práticas da medição em alta maturidade, as quais envolvem o controle estatístico de processos (BARCELLOS *et al.*, 2013).

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 da ausência de ferramentas, as organizações desenvolvem suas próprias soluções, as quais usualmente não atendem às necessidades da medição. Por exemplo, 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).

Há na literatura algumas propostas que descrevem arquiteturas para apoio computacional ao processo de medição. No entanto, essas propostas não descrevem seus componentes em um nível suficiente de detalhes. Por exemplo, a maioria das propostas não apresenta o modelo de dados, descrições detalhadas e restrições do repositório presente na arquitetura, o que impossibilita que esse modelo seja usado como referência para a construção de uma solução similar por uma organização. Além disso, a maioria das propostas não considera apoio ao controle estatístico de processos e nem possibilita a construção de uma solução gradativa, de acordo com a evolução das necessidades da medição (MARETTO e BARCELLOS, 2013).

O uso do controle estatístico em processos de software é recente e há ainda dúvidas sobre sua aplicação (WELLER e CARD, 2008). Buscando capturar a conceituação do domínio de medição de software tratando aspectos da medição desde os níveis iniciais até a alta maturidade (que inclui o controle estatístico de processos), Barcellos (2009) desenvolveu uma Ontologia de Referência para Medição de Software. Essa ontologia descreve o conhecimento relevante acerca de medição de software, útil às organizações que desejam realizar medição no contexto da melhoria de processos. Embora essa ontologia possa ser utilizada como fonte de conhecimento para o desenvolvimento de um repositório de medição adequado, essa tarefa não é trivial.

Considerando esse cenário, neste trabalho a Ontologia de Referência para Medição de Software (BARCELLOS, 2009; BARCELLOS *et al.*, 2013) é utilizada como base e é proposta uma arquitetura de referência para medição de software, cujos componentes são descritos em detalhes, visando possibilitar sua reutilização por organizações que desejem desenvolver soluções computacionais para medição de software. A arquitetura pode ser utilizada desde o início da medição até a medição em alta maturidade, podendo ser utilizada em sua totalidade ou parcialmente, dependendo das necessidades de medição.

1.3 Objetivos da Pesquisa

Este trabalho tem como objetivo geral definir uma arquitetura de referência para medição de software que possa auxiliar seus usuários na definição da arquitetura de soluções computacionais para medição de software, considerando tanto medição tradicional quanto em alta maturidade. Esse objetivo geral pode ser detalhado nos seguintes objetivos específicos:

(i) Elaborar uma abordagem para a definição de arquiteturas para medição de software;

(ii) Identificar e descrever em detalhes os componentes da arquitetura de referência para medição de software;

(iii) Aplicar a abordagem de definição de arquiteturas proposta, utilizando a arquitetura de referência como base para implementar uma ferramenta de apoio à medição, a fim de verificar a viabilidade de utilização da arquitetura de referência.

(iv) Avaliar a arquitetura de referência proposta.

1.4 Método de Pesquisa

Este trabalho foi conduzido de acordo com os seguintes passos:

i) *Revisão da Literatura:* neste passo ocorreu a aquisição de conhecimento sobre os temas relacionados ao trabalho, a saber: medição de software, controle estatístico de processos, ontologias, arquiteturas de referência e arquiteturas de medição. Inicialmente, foi realizada uma revisão informal da literatura, onde a pesquisa por publicações relacionadas foi realizada de forma não sistemática, tendo sido lidos artigos, livros, dissertações, teses e relatórios técnicos considerados relevantes ao trabalho. Nesse momento não houve restrições quanto ao uso de mecanismos de busca nem ao formato das publicações, bastando o material ter reconhecimento científico. Após a revisão informal, foi realizada uma investigação formal da literatura (mapeamento sistemático) onde se investigou arquiteturas de medição e se analisou suas características. Os resultados do mapeamento sistemático foram publicados em um artigo no 10th *Experimental Software Engineering Latin American Workshop* (ESELAW) (MARETTO e BARCELLOS, 2013).

ii) *Desenvolvimento da Arquitetura de Referência:* nesta etapa as informações obtidas a partir da revisão da literatura foram utilizadas para apoiar a definição da

arquitetura de referência. Antes de definir a arquitetura propriamente dita, foi desenvolvida uma abordagem que descreve a definição de arquiteturas de medição seguindo-se um conjunto de níveis subsequentes, partindo do nível mais abstrato para o menos abstrato, até a obtenção de aplicações de apoio à medição. Em seguida, foram definidos os componentes da arquitetura de referência e cada um deles foi detalhado. Durante esta etapa uma das principais fontes de conhecimento foi a Ontologia de Referência para Medição de Software - ORMS (BARCELLOS, 2009; BARCELLOS *et al.*, 2013).

iii) Instanciação da Arquitetura de Referência: nesta etapa a arquitetura de referência foi utilizada como base para a implementação da versão preliminar de uma ferramenta de apoio ao processo de medição. A arquitetura de referência foi instanciada para uma arquitetura específica, a qual foi implementada. A implementação da ferramenta teve como objetivo servir como uma prova de conceito (OATES, 2006) da arquitetura proposta e como uma avaliação inicial de sua viabilidade. Durante esta etapa, foram realizados ajustes na arquitetura de referência, a partir das observações realizadas durante o uso da arquitetura de referência.

iv) *Avaliação da Arquitetura de Referência:* nesta etapa foi realizado um estudo experimental com alguns alunos de mestrado e doutorado do Programa de Pós-Graduação em Informática da UFES, a fim de verificar se a ferramenta desenvolvida a partir da arquitetura de referência seria capaz de atender aos requisitos considerados relevantes para uma solução computacional de apoio ao processo de medição. Os resultados do estudo foram utilizados como base para a realização de novos ajustes na arquitetura de referência proposta.

v) *Escrita da Dissertação:* os resultados obtidos durante a execução dos passos anteriores foram documentados nesta dissertação.

1.5 Organização da Dissertação

Neste capítulo inicial foram apresentadas as principais ideias desta dissertação, descrevendo o contexto de aplicação, motivações, objetivos e metodologia de pesquisa. Além desta introdução, este texto é composto pelos seguintes capítulos e apêndices:

- **Capítulo 2 (Arquiteturas para Medição de Software):** apresenta os principais conceitos relacionados à medição de software e arquiteturas definidas para esse domínio, bem como os principais resultados obtidos em

um mapeamento sistemático que investigou arquiteturas para medição de software.

- **Capítulo 3 (Arquitetura de Referência para Medição de Software):** apresenta a abordagem de definição de arquiteturas e descreve a arquitetura de referência proposta. Neste capítulo os componentes da arquitetura são descritos em detalhes.
- **Capítulo 4 (Instanciação e Avaliação da Arquitetura de Referência):** discute o uso da arquitetura de referência na definição de uma arquitetura específica, apresenta a ferramenta de medição desenvolvida utilizando-se a arquitetura instanciada e os resultados de um estudo experimental realizado para avaliar a arquitetura proposta.
- **Capítulo 5 (Conclusões e Perspectivas Futuras):** apresenta as considerações finais do trabalho, as contribuições e propostas de trabalhos futuros para continuidade e aprimoramento do trabalho.
- **Apêndice A (Mapeamento Sistemático da Literatura):** apresenta na íntegra o mapeamento sistemático da literatura sobre arquiteturas de medição realizado no contexto deste trabalho.
- **Apêndice B (Modelos de Classes, Restrições e Mapeamentos):** apresenta na íntegra os modelos conceituais e as restrições do Repositório de Medição, que é um dos componentes da arquitetura de referência, e o mapeamento entre os conceitos da Ontologia de Referência para Medição de Software e as classes dos modelos conceituais.
- **Apêndice C (Relação entre Requisitos dos Utilitários de Entrada e Saída e Classes do Repositório de Medição):** apresenta a relação entre os requisitos que descrevem os Utilitários de Entrada e Saída (componentes da arquitetura de referência) e as classes do Repositório de Medição.
- **Apêndice D (Formulários Usados no Estudo Experimental):** apresenta os formulários que foram utilizados no estudo realizado.

Capítulo 2

Arquiteturas para Medição de Software

Neste capítulo são apresentados os principais conceitos relacionados à medição de software e arquiteturas definidas para esse domínio, bem como os principais resultados obtidos em um mapeamento sistemático que investigou arquiteturas para medição de software.

2.1. Introdução

Durante a execução de projetos de software é necessário coletar, armazenar e analisar dados que apoiem a tomada de decisão no contexto dos projetos e da organização. Medição de software é uma prática fundamental para a gerência de projetos e melhoria de projetos, estando presente nos principais modelos que tratam da melhoria de processos, tais como ISO/IEC 12207, CMMI e MR-MPS-SW.

Para que uma organização realize medição de maneira eficiente é necessária uma infraestrutura que apoie a coleta, armazenamento e análise de dados da medição. Essa infraestrutura, formada por componentes relacionados, pode ser chamada arquitetura de medição.

Neste capítulo são abordados os aspectos teóricos referentes ao tema deste trabalho. Para isso, ele está estruturado em cinco seções além desta introdução. As seções 2.2 e 2.3 abordam medição de software e controle estatístico de processos. A seção 2.4, aborda ontologias. A seção 2.5 trata de arquiteturas de medição de software e apresenta os principais resultados de um mapeamento sistemático que investigou este tema. Por fim, na seção 2.6 são realizadas as considerações finais do capítulo.

2.2. Medição de Software

Segundo Dumke (2010), medição é o processo pelo qual números ou símbolos são atribuídos a propriedades de entidades do mundo real de forma a descrevê-las. Ao longo das atividades de engenharia de software, diversas medições podem ser realizadas visando à obtenção de informações relevantes, como, por exemplo, o tamanho dos projetos, os custos de desenvolvimento e a quantidade de defeitos.

No contexto de projetos de software, a medição pode auxiliar a elaboração de planos realísticos e pode prover informações úteis ao acompanhamento do alcance aos objetivos, à identificação de problemas e à tomada de decisões informadas (McGARRY *et*

al., 2002). No contexto organizacional, a medição pode auxiliar na análise do desempenho dos processos e apoiar o estabelecimento de metas factíveis, bem como a identificação de ações que visem melhorar a competitividade da organização (TARHAN e DEMIRORS, 2006).

Embora os benefícios obtidos a partir da implementação da medição sejam muitos, realizar medição demanda esforço e, consequentemente, há custos envolvidos. Similar a outras atividades de controle, estima-se que a medição corresponda de 0.3% a 1% do esforço desprendido no desenvolvimento. Sendo que, no início do uso da medição pela organização, essa porcentagem é mais elevada devido à estruturação inicial requerida: realização de treinamentos, elaboração de *templates*, definição dos mecanismos de coleta e implantação de ferramentas (DUMKE e EBERT, 2010).

Dumke e Ebert (2010) defendem que a eficiência da medição de software está fortemente relacionada ao apoio de ferramentas associadas a todo o processo de medição. Card *et al.* (2008), por sua vez, afirmam que apesar de ferramentas serem muito importantes para apoiar a realização do processo de medição, o uso delas não garante o sucesso de um programa de medição. Segundo McGarry *et al.* (2002), o sucesso de um programa de medição está diretamente relacionado a dois fatores: (i) a coleta, análise e divulgação dos resultados das medições devem estar diretamente relacionadas às informações requeridas para as tomadas de decisão; e (ii) o processo de medição deve ser bem estruturado e repetível.

Existem diversos padrões que orientam sobre o processo de medição de software, dentre eles: a ISO/IEC 15939 (ISO/IEC, 2007), a ISO/IEC 12207 (ISO/IEC, 2008), o Std. 1061 (IEEE, 1998), o PSM – *Practical Software Measurement* (McGARRY *et al.*, 2002), o CMMI (SEI, 2010) e o MR-MPS-SW (SOFTEX, 2012). Em linhas gerais, o processo de medição de software inclui as seguintes atividades: planejamento da medição, execução da medição e avaliação da medição (ISO/IEC, 2008), conforme ilustra a Figura 2.1.

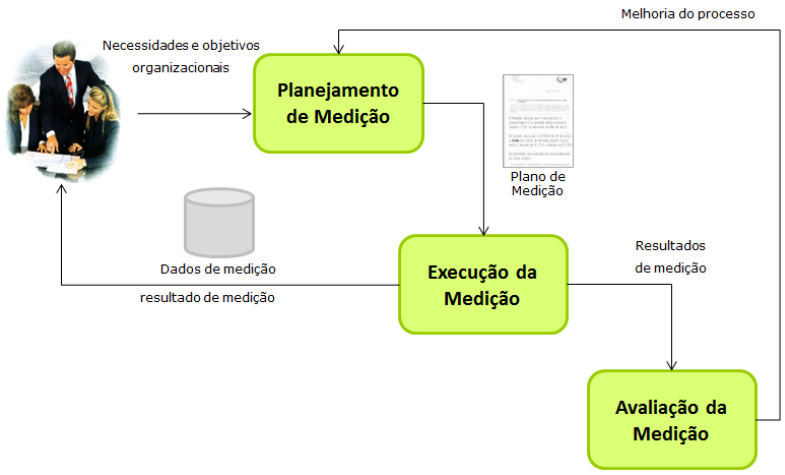

Figura 2.1 - Atividades gerais do processo de medição de software.

No planejamento da medição, com base nos objetivos e necessidades da organização, são definidas as entidades (processos, produtos, etc.) que serão consideradas para medição, quais de suas propriedades (custos, defeitos, etc.) serão mensuradas e quais medidas serão utilizadas. Para cada medida deve ser estabelecida uma definição operacional, que detalha aspectos relacionados à coleta e análise de dados. Nesse contexto, Dumke e Ebert (2010) destacam a importância de se ter definições operacionais precisas, buscando-se evitar que pessoas diferentes entendam a medida de maneiras diferentes e realizem medições inconsistentes. Os resultados do planejamento da medição são registrados no Plano de Medição.

Existem alguns métodos que auxiliam as organizações no planejamento da medição, visando à identificação de medidas que sejam realmente úteis. De acordo com BASILI e Green (1994) a definição de medidas úteis depende da identificação dos fatores críticos que são capazes de determinar se os objetivos de negócio serão ou não alcançados. Nesse sentido, uma das abordagens mais conhecidas é o GQM (*Goal Question Metric*) (BASILI *et al.*, 1994), que considera que, para cada objetivo estabelecido, é possível determinar questões cujas respostas estão associadas a medidas. Algumas variações desse método são o GQ(I)M *(Goal Question (Indicator) Measure)* (GOETHERT e FISHER, 2003), que, baseado no entendimento de que identificar questões e medidas sem visualizar um indicador muitas vezes não é suficiente, propõe o alinhamento de medidas e indicadores com os objetivos, e o GQM*Strategies (BASILI *et al.*, 2007), que acrescenta um conjunto de extensões no topo do método GQM, tornando explícitos os objetivos de negócio, as estratégias e os objetivos de software.

Após ser planejada, a medição pode ser executada. A execução consiste em coletar e armazenar os dados para as medidas definidas, de acordo com suas definições operacionais especificadas na etapa de planejamento.

Em relação ao armazenamento dos dados da medição Braungarten *et al.* (2005) afirmam que ele pode ser feito em planilhas eletrônicas, bancos de dados ou repositórios. De acordo com os autores, devido à simplicidade e baixo custo, as planilhas são usadas em grande parte das organizações. No entanto, elas não são adequadas a médio ou longo prazo, pois a consistência estrutural não pode ser assegurada e o acesso aos dados pode ser dificultado quando novas medidas ou mais dados são necessários. Bancos de dados são mais eficientes que planilhas, especialmente quando se trata de um grande volume de dados. Apesar dessa vantagem, tipicamente requerem maior investimento para construção e manutenção. Além disso, às vezes, organizações utilizam vários bancos de dados para tratarem diferentes aspectos da medição, o que pode levar ao comprometimento da integridade dos dados e a redundâncias que podem afetar a obtenção de informações corretamente. Repositórios, por sua vez, consideram uma visão mais ampla da medição do que bancos de dados e podem lidar com questões de integração de dados para permitir armazenamento e acesso adequados. Em contrapartida, a construção e o gerenciamento de repositórios costumam exigir esforço ainda maior que o empregado nos bancos de dados.

Com os dados coletados e armazenados, diferentes análises podem ser realizadas. A análise dos dados fornece informações para a tomada de decisão, favorecendo a identificação e execução de ações apropriadas. Durante a análise, frequentemente os dados são representados graficamente, a fim de propiciar uma melhor visualização e favorecer a análise dos dados. Gráficos podem representar tendências de dados, variações e tornar as relações entre diferentes medidas mais claras. Segundo McGarry *et al.* (2002), os gráficos mais comumente utilizados na medição de software são o gráfico em linha, usado para medidas ao longo do tempo, o gráfico em barras, usado para contagem de um grupo de componentes ou eventos, e o gráfico de dispersão, usado para relação entre dois fatores. Como resultado das análises algumas ações podem ser identificadas, como, por exemplo, estender prazos, adicionar recursos, mudar a abordagem de desenvolvimento e realocar recursos (McGARRY *et al.*, 2002).

Modelos de maturidade, tais como o CMMI (SEI, 2010) e MR-MPS-SW (SOFTEX, 2012) incluem medição como um dos processos fundamentais para a melhoria de processos. No CMMI a medição é introduzida no nível 2, enquanto que no MR-MPS-SW isso ocorre no nível F. A medição realizada nesses níveis é dita medição tradicional e tem

como principal objetivo apoiar o gerenciamento de projetos e processos por meio da comparação entre valores planejados e valores realmente praticados. Nos níveis mais elevados de maturidade (níveis 4 e 5 do CMMI e níveis B e A do MR-MPS-SW) a medição tradicional não é mais suficiente. Nesses níveis é necessário realizar o controle estatístico dos processos críticos, a fim de conhecer seu comportamento, determinar seu desempenho em execuções anteriores e prever seu desempenho em projetos em curso e futuros, verificando se são capazes de alcançar os objetivos estabelecidos. Essa medição é chamada medição em alta maturidade (BARCELLOS *et al.*, 2013).

2.3. Controle Estatístico de Processos

O controle estatístico de processos (CEP) teve origem na manufatura, com o propósito de apoiar a implementação de programas de melhoria contínua em linhas de produção. Os bons resultados do seu uso levaram à aplicação em outras áreas, dentre elas o desenvolvimento de software (ROCHA *et al.*, 2012). A sua utilização envolve o uso de gráficos de controle e métodos estatísticos específicos para analisar os dados coletados para medidas ao longo dos projetos e fornecer informações sobre o desempenho dos processos.

A utilização de gráficos de controle e métodos estatísticos provê aos engenheiros de software e gerentes de projetos uma visão quantitativa do comportamento dos processos de software. Metaforicamente, os gráficos de controle estão para os gerentes de projeto e engenheiros de software assim como o painel de controle de um automóvel está para seu motorista. O motorista depende do que é mostrado no painel de controle para tomar decisões, tais como: abastecer o carro, colocar água, diminuir a velocidade nas curvas ou parar em uma oficina mecânica quando um indicador no painel revela a presença de problemas. De forma análoga, os gráficos de controle compõem um painel de controle e guiam as decisões dos gerentes de projetos e engenheiros de software, fornecendo o conhecimento necessário sobre o comportamento dos processos (ROCHA *et al.*, 2012).

A principal diferença entre o controle estatístico de processos e a estatística clássica é o uso de métodos baseados em dados estáticos no tempo. Na estatística clássica a ordem dos dados não afeta os resultados da análise. Esse primeiro tipo de análise é eficiente para realizar comparações, por exemplo, a taxa de alteração de requisitos de projetos, entretanto, apenas com isso não é possível saber se está ou não havendo melhora no comportamento de um processo. Portanto, através de análises cronológicas o controle estatístico de processos possibilita detecção de tendências no comportamento e antecipação a problemas relacionados ao desempenho dos processos (BENNEYAN *et al.*, 2003).

A aplicação do controle estatístico a processos de software envolve algumas mudanças no processo de medição tradicional, uma vez que a exigência em relação a alguns aspectos aumenta. Exemplos desses aspectos são: a qualidade das definições operacionais das medidas, a frequência de coleta de dados, a granularidade das medidas, a homogeneidade de dados e o agrupamento dos dados para análise (BARCELLOS, 2009; TARHAN e DEMIRORS, 2012).

A utilização do controle estatístico permite conhecer o comportamento dos processos e fazer previsões sobre seu desempenho. Nesse contexto, dois conceitos são importantes estabilidade e capacidade. Um processo só é considerado estável se o mesmo é repetível, ou seja, tem um comportamento que varia dentro de limites esperados e considerados aceitáveis. As causas das variações aceitáveis são ditas causas comuns (SHEWART, 1980) e são inerentes aos aspectos envolvidos na execução normal do processo, como, por exemplo, a variação da produtividade das equipes. Processos estáveis são processos cujo comportamento pode ser previsto, e a previsibilidade é algo desejável pelas organizações.

Um processo estável está sob controle estatístico, ou seja, um processo no qual todas as variações no comportamento são aceitáveis. Os limites que descrevem até que ponto as variações no comportamento de um processo são aceitáveis são estabelecidos com base em dados históricos da execução do processo e descrevem uma *baseline* de desempenho, que servirá para prever o comportamento do processo em suas próximas execuções e, também, para comparar se o comportamento dessas execuções está de acordo com o comportamento esperado (BARCELLOS, 2009). A Figura 2.2 ilustra a presença de variações provocadas por causas comuns em um processo.

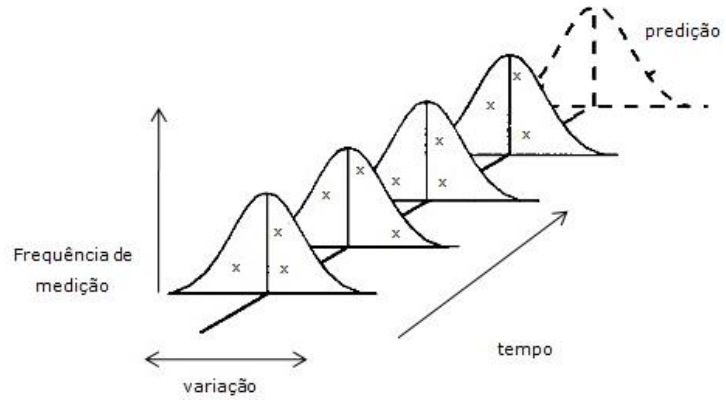

Figura 2.2 - Variações provocadas por causas comuns (ROCHA *et al.*, 2012).

Processos que variam além dos limites esperados são ditos processos instáveis e as causas que provocam desvios que excedem os limites de variação aceitável para o

comportamento do processo são chamadas causas especiais, (SHEWART, 1980). Processos instáveis são um problema para as organizações, uma vez que não é possível prever seu comportamento em execuções futuras. A Figura 2.3 ilustra a presença de variações provocadas por causas especiais em um processo.

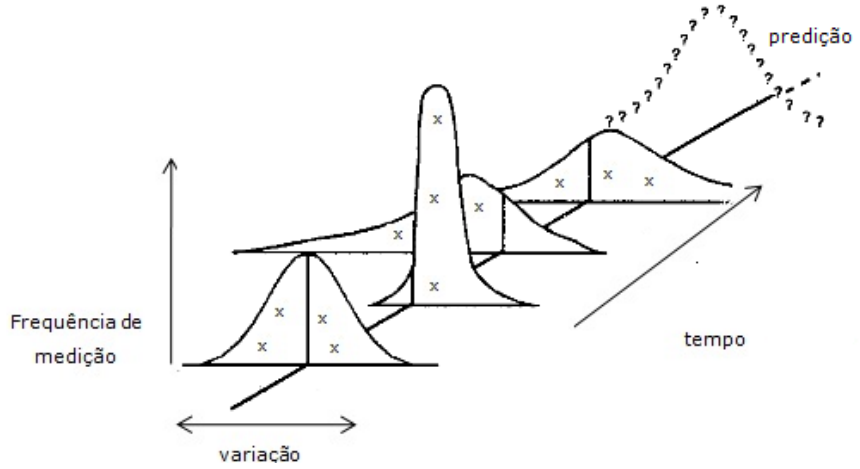

Figura 2.3 - Variações provocadas por causas especiais (ROCHA *et al.*, 2012).

Quando o desempenho de um processo estável permite que ele alcance os objetivos para ele estabelecidos, o processo é dito capaz (FLORAC e CARLETON, 1999). Uma vez que um processo seja capaz, é possível realizar ações que visem melhorar sua capacidade, ou seja, diminuir os limites de variação que são considerados aceitáveis para seu comportamento. A capacidade do processo é conhecida como voz do processo. Já a capacidade desejada para o processo, que leva em consideração os objetivos da organização e do cliente, é chamada de voz do cliente. É desejável que a voz do processo atenda a voz do cliente, mas também é preciso reconhecer que, algumas vezes, a capacidade desejada para o processo não é possível de ser obtida e, portanto precisa ser revista (WHEELER e POLING, 1998).

No contexto de modelos de maturidade como o CMMI e o MR-MPS-SW, o controle estatístico de processos é utilizado em dois níveis: organizacional e dos projetos. No nível organizacional ocorre a **análise de desempenho de processos**. Nesse nível, os dados coletados ao longo dos projetos são utilizados para descrever o comportamento dos processos da organização. Dados de diversos projetos que tenham o mesmo perfil, ou seja, que sejam similares entre si, são agrupados e são aplicados métodos do controle estatístico para analisar o comportamento dos processos e estabelecer *baselines* e modelos de desempenho[1]. No nível dos projetos ocorre a **gerência quantitativa dos projetos**, que

[1] Modelos representados por fórmulas que descrevem as relações quantitativas entre medidas (por exemplo, esforço = K x tamanho, onde K é uma constante obtida a partir de dados históricos).

consiste em analisar o desempenho dos processos nos projetos, utilizando as *baselines* estabelecidas no nível organizacional. Assim, em cada projeto, o desempenho dos processos no projeto é comparado com o desempenho para ele esperado e, caso não seja condizente, ações corretivas devem ser realizadas. Ainda, o planejamento do projeto pode ser realizado utilizando os modelos de desempenho definidos no nível organizacional (ROCHA *et al.*, 2012).

Para apoiar a representação e análise dos dados do comportamento dos processos, existem várias ferramentas: diagramas *scatter*, histogramas, diagramas de causa e efeito, gráficos de tendências, gráficos de barras, gráficos de controle, gráficos de pareto, dentre outros (FLORAC e CARLETON, 1999). Os gráficos de controle são a ferramenta básica do controle estatístico de processos, pois são eficientes em mostrar a variação do processo e permitir a avaliação da sua estabilidade. Os gráficos que controle são brevemente apresentados a seguir.

2.3.1. Gráficos de Controle

Gráficos de controle associam métodos de controle estatístico e representação gráfica para quantificar o comportamento de processos auxiliando a detectar os sinais de variação no comportamento dos processos e a diferenciá-los dos ruídos. Os ruídos dizem respeito às variações que são aceitáveis e são intrínsecas aos processos (causas comuns). Já os sinais indicam variações que precisam ser analisadas (causas especiais) e, se possível, eliminadas (FLORAC e CARLETON, 1999).

Existem vários tipos de gráficos de controle, e cada um possui uma melhor aplicabilidade a situações específicas. A forma como os dados serão plotados e como os limites de controle serão calculados são definidos pelo tipo do gráfico. Cada tipo possui um conjunto de métodos quantitativos ou estatísticos associados (BARCELLOS, 2009).

A estrutura básica de um gráfico de controle é ilustrada na Figura 2.4. Os limites central, superior e inferior são calculados a partir de um conjunto de valores coletados para a medida representada no gráfico, sendo que os limites superior e inferior ficam a uma distância de três desvios padrão (σ) em relação à linha central.

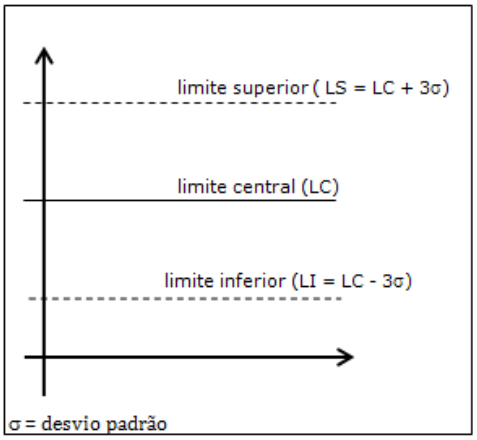

Figura 2.4 - Formato básico de um gráfico de controle (ROCHA *et al.*, 2012).

Como um exemplo simples de uso do controle estatístico de processos, suponha que uma organização deseje analisar o comportamento do seu processo de inspeção. Para isso, decidiu utilizar a medida taxa de detecção de defeitos em inspeção, dada pela razão entre o número de defeitos detectados na inspeção e o tamanho do produto inspecionado. Para analisar o comportamento do processo, os dados coletados em 20 inspeções foram plotados em um gráfico de controle, como ilustrado na Figura 2.5.

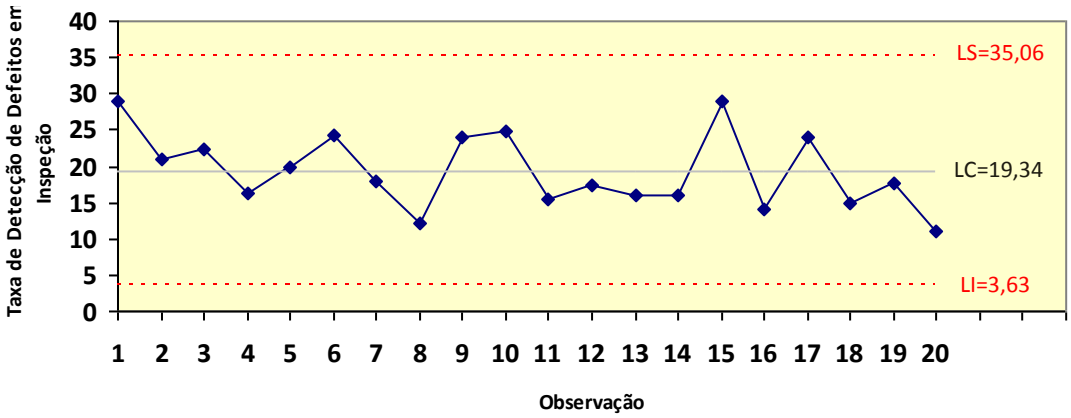

Figura 2.5 - Exemplo de gráfico de controle.

Analisando-se o gráfico é possível observar que o processo de inspeção é estável, quando analisado utilizando-se a medida taxa de detecção de defeitos. Sendo um processo estável, uma *baseline* de desempenho deve ser estabelecida. A *baseline* do processo de inspeção é descrita pelos limites superior (35,06), inferior (3,63) e central (19,34). Esses valores descrevem, então, o comportamento esperado para as próximas execuções do processo de inspeção.

Uma vez que o processo de inspeção é um processo estável, a organização pode analisar se ele é um processo capaz. Para determinar a capacidade de um processo é necessário conhecer qual é o comportamento desejado para aquele processo, ou seja, quais

são os limites (superior e inferior) que descrevem a voz do cliente. Os limites do comportamento desejado devem, então, ser comparados com os limites do comportamento real do processo, descrito pela sua *baseline*. Uma forma de se obter a capacidade de um processo é utilizar o índice de capacidade, dado por Cp =(LSe – LIe)/ (LSb – LIb), onde Cp = Índice de Capacidade, LSe = Limite Superior do Desempenho Especificado, LIe = Limite Inferior do Desempenho Especificado, LSb = Limite Superior da Baseline de Desempenho e LIb = Limite Inferior da Baseline de Desempenho. Um índice de capacidade maior ou igual a 1 indica um processo capaz, enquanto que um índice de capacidade menor que 1 indica um processo não capaz. Para o exemplo apresentado e supondo-se que o desempenho especificado para o processo seja 9 para limite inferior e 33 para limite superior, ao se verificar o índice de capacidade (Cp = 0,25) tem-se que, embora o processo de inspeção seja estável, ele não é capaz. Nesse caso, devem ser realizadas ações para melhorar o processo ou o desempenho especificado deve ser revisto.

Vale ressaltar que em algumas situações é possível ter um índice de capacidade maior do que 1 e, ainda assim, ter um processo incapaz. Por exemplo, um processo com limites superior e inferior da *baseline* dados, respectivamente, por 18 e 10 e com limites do desempenho especificado dados por 15 e 5, teria Cp = 1,25 e, ainda assim, não seria um processo capaz. Por essa razão, sugere-se que, além do cálculo do índice de capacidade, a análise da capacidade de um processo seja realizada representando-se em um mesmo gráfico os limites da *baseline* de desempenho e do desempenho especificado, a fim de observar se os limites da *baseline* são internos aos do desempenho especificado, o que caracteriza um processo capaz.

2.4. Ontologias

Na Filosofia, uma ontologia pode ser vista como um sistema particular de categorias levando em conta uma certa visão do mundo, independente da linguagem. Por outro lado, para comunidades relacionadas à Ciência da Computação, tais como Inteligência Artificial, Engenharia de Software e *Web* Semântica, uma ontologia é um artefato de engenharia formado por um vocabulário usado para descrever certa realidade e por um conjunto de conjecturas explícitas relacionadas ao significado pretendido das palavras do vocabulário. Esse conjunto de conjecturas tem, geralmente, a forma da teoria de lógica de primeira ordem, onde as palavras do vocabulário aparecem como nomes de predicados unários ou binários, respectivamente chamados *conceitos* e *relações*. No caso mais simples, uma ontologia descreve uma hierarquia de conceitos relacionados por relações de

classificação. Em casos mais sofisticados, axiomas apropriados são adicionados a fim de expressar outros relacionamentos entre conceitos e restringir a interpretação pretendida (GUARINO, 1998).

No âmbito da Ciência da Computação, ontologias têm sido reconhecidas como um instrumento conceitual bastante útil desde a década de 60. Na Engenharia de Software elas têm sido tipicamente utilizadas para reduzir ambiguidades conceituais, tornar transparentes as estruturas de conhecimento, apoiar o compartilhamento de conhecimento e a interoperabilidade entre sistemas (USCHOLD e JASPER, 1999), tendo destaque sua utilização no ramo da Engenharia de Domínio, onde, na prática, uma ontologia de domínio pode ser utilizada como um modelo de domínio, uma vez que ambos buscam fornecer um entendimento uniforme e não ambíguo de objetos e suas relações, provendo uma conceituação acerca de um determinado domínio (FALBO *et al.*, 2002).

Na literatura existem várias classificações para as ontologias. Neste trabalho considera-se a classificação proposta por Guarino (1998), que tem como seu critério de classificação o nível de generalidade. A Figura 2.6 mostra os tipos de ontologias e os seus relacionamentos.

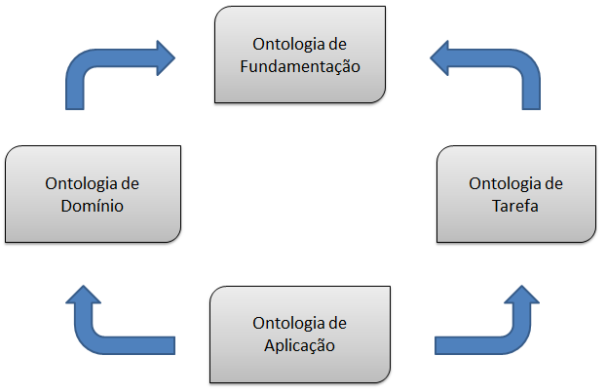

Figura 2.6 - Relacionamentos entre os tipos de ontologias (GUARINO, 1998).

Ontologias de fundamentação descrevem conceitos gerais, que são independentes de problema ou domínio. Este tipo de ontologia é especificado a partir de estudos de ciências como filosofia e psicologia cognitiva, tratando conceitos de espaço, tempo, objeto, evento e outros. Ontologias de domínio descrevem os conceitos e relacionamentos de um domínio específico, como, por exemplo, cardiologia, petróleo e gás. Ontologias de tarefa, por sua vez, fazem descrições quanto a uma tarefa ou atividade específica, como, por exemplo, um diagnóstico ou uma venda. Tanto a ontologia de domínio quanto a de tarefa devem, idealmente, serem obtidas pela especialização de termos descritos em uma ontologia de fundamentação. Por fim, ontologias de aplicação descrevem conceitos que

combinam especializações de conceitos presentes nas ontologias de domínio e de tarefa, assim, os conceitos correspondem a papéis desempenhados por entidades do domínio quando estão executando uma atividade.

O uso de ontologias de fundamentação como base para a construção de ontologias de domínio auxilia na obtenção de ontologias fidedignas à realidade e com clareza conceitual, características consideradas imprescindíveis para modelos conceituais (GUIZZARDI *et al.*, 2008a).

No contexto deste trabalho, duas ontologias são relevantes: a ontologia de fundamentação UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005) e a ontologia de domínio ORMS (Ontologia de Referência para Medição de Software) (BARCELLOS, 2009; BARCELLOS *et al.*, 2013). Essas duas ontologias são apresentadas a seguir.

2.4.1. *The Unified Foundational Ontology* (UFO)

UFO (GUIZZARDI, 2005; GUIZZARDI, *et al.*, 2008a; GUIZZARDI, *et al.*, 2008b) é uma ontologia de fundamentação que tem sido desenvolvida baseada em um número de teorias das áreas de Ontologias Formais, Lógica Filosófica, Filosofia da Linguagem, Linguística e Psicologia Cognitiva. Ela é composta por três partes principais: UFO-A, UFO-B e UFO-C.

UFO-A é uma ontologia de objetos (*Endurants*). Ela é o cerne de UFO. Uma distinção fundamental em UFO-A se dá entre indivíduos (*Particular*) e universais (ou tipos) (*Universal*). Indivíduos são entidades que existem na realidade e que possuem uma identidade única. Tipos ou universais são padrões de características que podem ser materializados em um número de diferentes indivíduos.

UFO-B, por sua vez, é uma ontologia de eventos (*Perdurants*). Eventos são indivíduos compostos de partes temporais. A principal diferença entre eventos (*Perdurants*) e objetos (*Endurants*) é que um objeto existe ou não existe no tempo, enquanto eventos ocorrem no tempo.

Por fim, UFO-C é uma ontologia de entidades sociais (*Endurants* e *Perdurants*) construída com base nas partes A e B de UFO. Uma de suas principais distinções se dá entre agentes e objetos. Agentes são capazes de realizar eventos, enquanto objetos participam de eventos.

UFO vem sendo utilizada como base para a construção e reengenharia de diversas ontologias de domínio. Como exemplos têm-se a reengenharia da Ontologia de Processos de Software (BRINGUENTE, et al., 2011) e da Ontologia de Organizações de Software

(BARCELLOS e FALBO, 2009) e criação da Ontologia de Requisitos de Software (FALBO e NARDI, 2008) e da Ontologia de Referência para Medição de Software (BARCELLOS, 2009; BARCELLOS *et al.*, 2013).

2.4.2. Ontologia de Referência para Medição de Software (ORMS)

ORMS (BARCELLOS, 2009; BARCELLOS *et al.*, 2013) é uma ontologia de domínio que abrange tanto os aspectos da medição tradicional quanto em alta maturidade. Ela foi desenvolvida fundamentada em UFO, é composta por seis subontologias e reutiliza conceitos das ontologias de Organização de Software (BARCELLOS e FALBO, 2009) e Processos de Software (BRINGUENTE *et al.*, 2011).

A Figura 2.7 apresenta as subontologias da ORMS, as ontologias integradas e as relações entre elas. Na figura, as (sub)ontologias são representadas como pacotes e as relações de dependência entre os pacotes indicam que uma (sub)ontologia utiliza conceitos de outra. Após a figura, uma breve descrição das (sub)ontologias é apresentada.

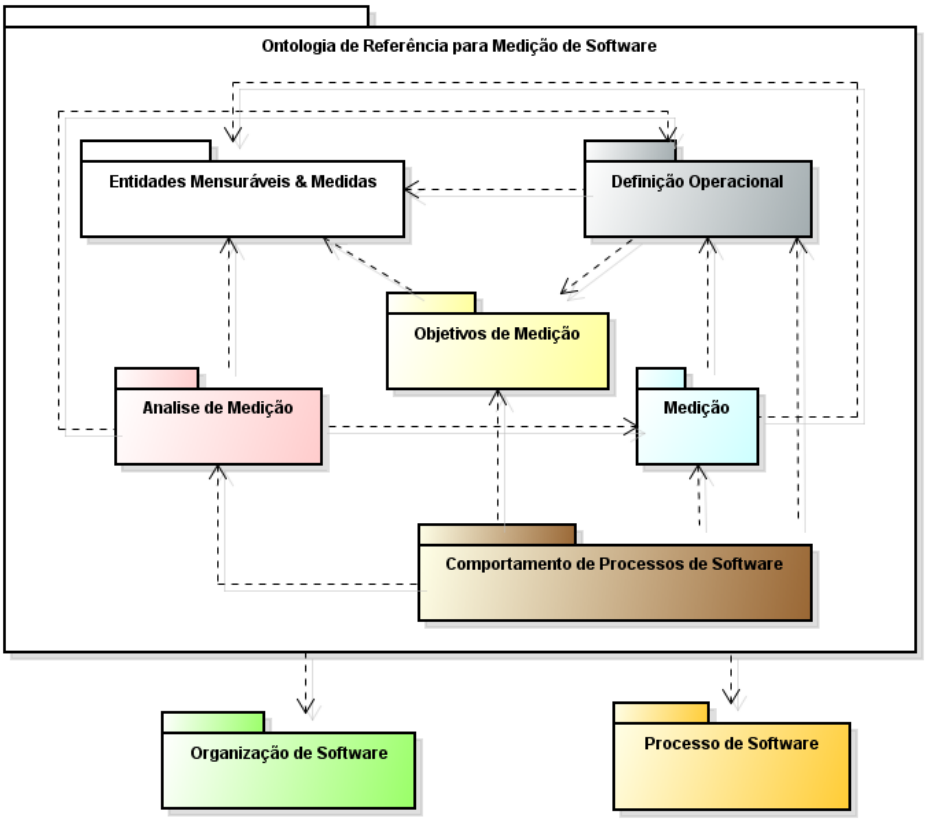

Figura 2.7 - Visão geral da Ontologia de Referência para Medição de Software.

- *Ontologia de Organização de Software*: trata de conceitos presentes no contexto de organizações de software, como projeto, equipe e alocação de recursos.

- *Ontologia de Processo de Software*: trata de conceitos relacionados a processos de software, como, por exemplo, processo padrão e atividades.
- *Subontologia de Entidades Mensuráveis e Medidas:* 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:* trata do alinhamento da medição de software com os objetivos estratégicos.
- *Subontologia de Definição Operacional de Medidas:* trata do detalhamento de aspectos relacionados à coleta e análise de medidas.
- *Subontologia de Medição de Software:* trata da medição propriamente dita, ou seja, a coleta e armazenamento dos dados para as medidas.
- *Subontologia de Análise de Medição:* trata da análise dos dados coletados para as medidas para obtenção das informações de apoio às decisões.
- *Subontologia de Comportamento de Processos:* trata da aplicação dos resultados da medição na análise do comportamento de processos.

Na Figura 2.8 é apresentado um pequeno fragmento contendo os conceitos mais básicos da subontologia Entidades Mensuráveis e Medidas. Os estereótipos são conceitos de UFO dos quais os conceitos de ORMS são especializados. O uso de estereótipos não é a forma mais indicado para representar a especialização de conceitos de UFO, pois apresenta alguns problemas e limitações. No entanto, considerando que as discussões sobre a fundamentação dos conceitos de ORMS em UFO não são foco deste trabalho e visando simplificar a apresentação dos modelos optou-se por utilizar essa abordagem.

ORMS é bastante extensa e, apesar de ter sido integralmente utilizada neste trabalho, apenas um pequeno fragmento é apresentado. A descrição completa da ORMS, incluindo os modelos completos, axiomas e descrições, pode ser encontrada em (BARCELLOS, 2009; BARCELLOS *et al.*, 2013). Discussões sobre a utilização de UFO como fundamentação para ORMS estão registradas em (BARCELLOS *et al.*, 2010a) (BARCELLOS *et al.*, 2010c).

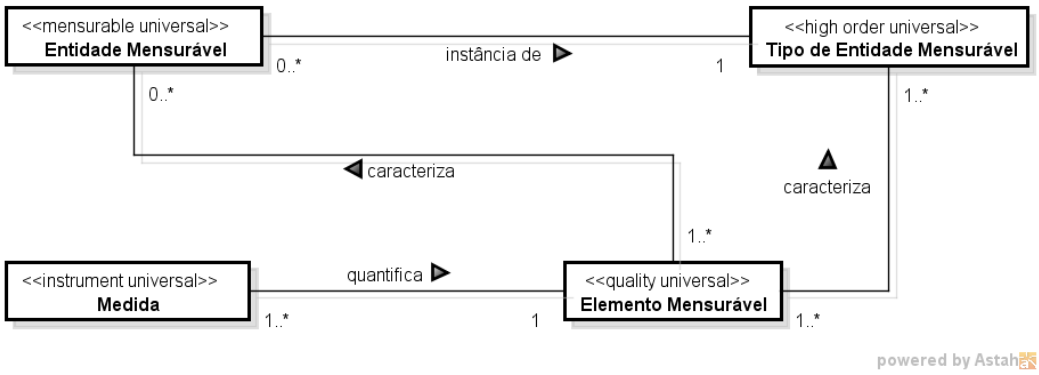

Figura 2.8 - Fragmento central da subontologia Entidades Mensuráveis e Medidas.

Uma Entidade Mensurável (*Mensurable Universal* em UFO) é algo que pode ser medido, como, por exemplo, um processo, um projeto ou um artefato. Uma entidade mensurável é de um determinado tipo (por exemplo, Processo, Projeto ou Artefato). Um Tipo de Entidade Mensurável (*High Order Universal* em UFO) é caracterizado por Elementos Mensuráveis (*Quality Universal* em UFO). Por exemplo, tamanho é um elemento mensurável que caracteriza entidades do tipo Projeto. Elementos mensuráveis são quantificados por Medidas (*Instrument Universal* em UFO). Por exemplo, o elemento mensurável tamanho, poderia ser quantificado pela medida número de pontos de função.

2.5. Arquiteturas para Medição de Software

Conforme dito no capítulo de Introdução, uma arquitetura é uma estrutura lógica em que os componentes são organizados e integrados (ZACHMAN, 1987). Comumente representadas por diferentes visões (especificações e esquemas), as arquiteturas correspondem a estruturas e direcionamentos a serem seguidos durante 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. O número de visões e o nível de abstração determinam o quão específica é uma arquitetura.

Nos últimos anos, um novo conceito relacionado à arquitetura tem se feito presente na literatura de Engenharia de Software: arquitetura de referência.

2.5.1. Arquiteturas de Referência

Segundo Muller (2013), uma arquitetura de referência captura a essência da arquitetura de um conjunto de softwares ou de um domínio e possui o propósito de guiar o desenvolvimento de arquiteturas para novos sistemas ou atualizações. Diferente do

conceito mais generalista de arquitetura puramente dita, uma arquitetura de referência tem o propósito evidente de reúso.

As arquiteturas de referência proveem uma taxonomia comum, uma visão arquitetural compartilhada, uma modularização e um contexto unificado (MULLER, 2013). A taxonomia contribui em facilitar a comunicação (por exemplo, por meio de um modelo de domínio). A visão arquitetural concentra e alinha esforços de múltiplas equipes quanto à especificação de arquiteturas. A modularização ajuda a dividir o esforço, permitindo que toda ou apenas parte seja usada, e o contexto único assegura uma posterior integração na evolução do sistema.

Assim como arquiteturas de software em geral, arquiteturas de referência podem representar diferentes visões e considerar diferentes níveis de detalhe e de abstração. Portanto, arquiteturas de referência podem ser definidas por meio de diferentes artefatos e com diferentes níveis de detalhes. Consequentemente, um desafio ao se definir uma arquitetura de referência é saber o quão suficiente ela deve ser em sua composição e descrição.

Segundo Muller (2013), definir uma arquitetura que seja compreendida pelas partes interessadas e que aborde as principais questões do domínio são alguns dos direcionamentos que podem ser seguidos para construir uma arquitetura de referência de qualidade.

No entanto, conforme argumenta Nakagawa *et al.* (2009), estabelecer arquiteturas de referência não é uma tarefa trivial. Capturar a essência de um domínio requer um levantamento bem elaborado e uma correta organização e apresentação das informações. Nesse sentido, o uso de ontologias pode ser uma boa abordagem.

Nakagawa *et al.* (2009) investiga os impactos do uso de ontologias na elaboração de arquiteturas de referência e, como resultado, destaca que a principal contribuição das ontologias está no apoio ao levantamento de informações e entendimento do domínio. O estudo também aponta que as visões de arquiteturas tradicionais não são suficientes para representar arquiteturas de referência, uma vez que, de forma geral, não preveem elementos para explicar cada conceito presente na arquitetura.

O processo de desenvolvimento de uma arquitetura de referência contém quatro atividades principais, conforme mostra a Figura 2.9 (GALSTER e AVGERIOU, 2011; MULLER, 2013; NAKAGAWA *et al.*, 2009).

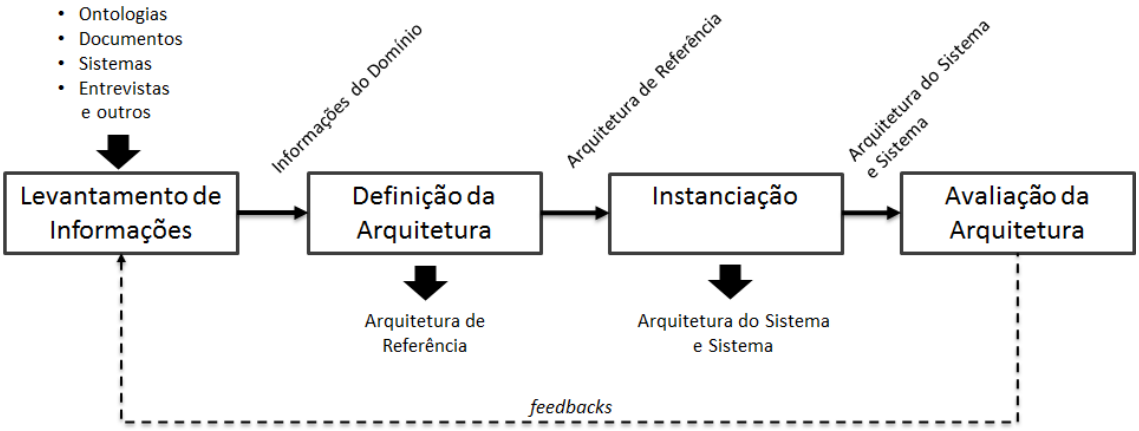

Figura 2.9 - Processo de desenvolvimento de uma arquitetura de referência (adaptado de (NAKAGAWA *et al.*, 2009)).

A primeira atividade consiste no *levantamento de informações* a respeito do domínio a ser tratado. Nesse momento devem ser realizadas investigações e pesquisas para se obter a maior quantidade possível de informações sobre o domínio em foco. O próximo passo consiste na *definição da arquitetura de referência* propriamente dita, a qual deve ser descrita através de documentos e modelos, representando as diferentes visões da arquitetura. Em seguida, ocorre a *instanciação* da arquitetura de referência em arquiteturas de sistemas. Por fim, é realizada a *avaliação da arquitetura*, a fim de avaliar, dentre outros, se sua descrição é satisfatória para que os utilizadores compreendam o domínio e as visões descritas, se a modularização permite a utilização da arquitetura em partes e se a arquitetura reflete o estado atual do domínio. Os resultados das avaliações de instâncias de uma arquitetura de referência fornecem informações que devem ser utilizadas para adequá-la.

Conforme comentado anteriormente, arquiteturas de referência são definidas principalmente visando à reutilização. Dessa forma, uma vez que uma arquitetura de referência tenha sido definida, avaliada e considerada adequada, espera-se que ela seja utilizada como base para a definição de arquiteturas de software para diversos sistemas. A Figura 2.10 apresenta a visão de Muller (2013) sobre o processo de utilização de arquiteturas de referência.

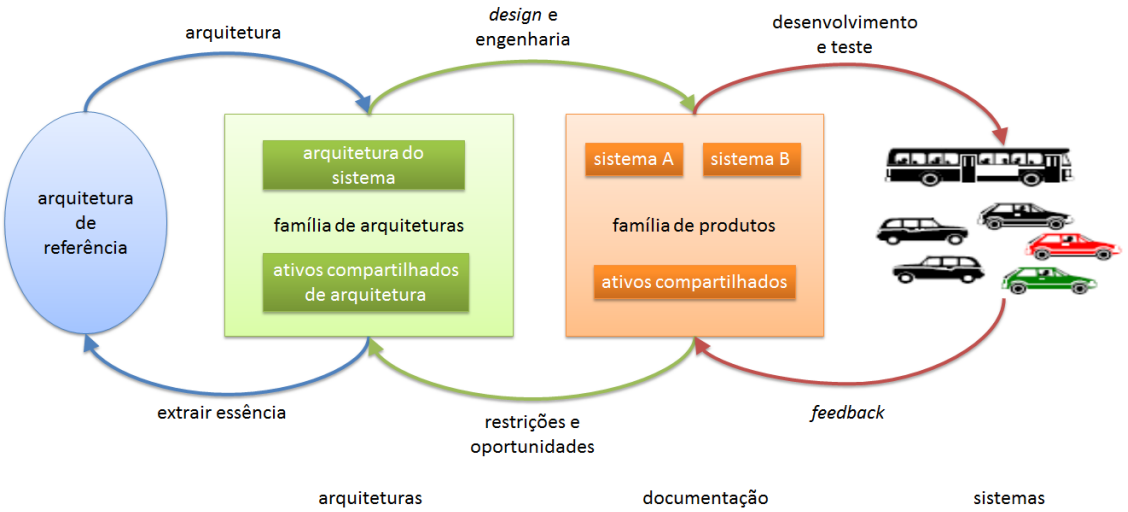

Figura 2.10 - Esquema de funcionamento da instanciação de uma Arquitetura de Referencia (MULLER, 2013).

Uma arquitetura de referência é utilizada como base para a definição da arquitetura de um ou mais sistemas. As diversas arquiteturas definidas para vários sistemas compõem uma família de arquiteturas e é provável que, embora essas arquiteturas apresentem particularidades, elas possuam uma parte compartilhada, que é a parte presente na arquitetura de referência. As arquiteturas de software definidas são, então, utilizadas como base para a implementação dos sistemas, os quais são utilizados no mundo real. Os resultados da utilização dos sistemas podem prover *feedback* capaz de nutrir a arquitetura de referência com informações essenciais que podem ser utilizadas para evoluir continuamente a arquitetura de referência.

2.5.2. Arquiteturas para Medição de Software

Para que uma organização realize medição de maneira eficiente é necessária uma infraestrutura que apoie a coleta, armazenamento e análise de dados da medição (DUMKE e EBERT, 2010). Nessa linha, existem na literatura algumas propostas de arquiteturas de apoio à medição de software.

Buscando-se conhecer propostas de arquiteturas para a medição de software descritas na literatura, decidiu-se realizar uma investigação formal por meio de um mapeamento sistemático. Segundo Kitchenham e Charters (2007), um mapeamento sistemático (também conhecido como estudo exploratório) realiza um amplo estudo em um tópico específico e visa identificar evidências disponíveis sobre esse tema.

Neste texto, o termo arquitetura de medição é utilizado para designar uma estrutura lógica na qual componentes são organizados e integrados a fim de apoiar, total ou parcialmente, o processo de medição. O componente central em uma arquitetura de

medição é o repositório de medição (DUMKE e EBERT, 2010), responsável pelo armazenamento e disponibilização dos dados da medição. Dada sua importância, algumas vezes o repositório de medição se confunde com a arquitetura de medição, havendo publicações que usam o termo repositório de medição quando descrevem arquiteturas para medição que abordam coleta, armazenamento ou análise (MARETTO e BARCELLOS, 2013). Por essa razão, durante o estudo foram investigadas publicações que apresentam tanto propostas de arquitetura de medição quanto de repositório de medição. Ainda, uma vez que durante a revisão informal percebeu-se o uso do termo infraestrutura com o mesmo significado adotado neste trabalho para arquitetura, publicações que abordam infraestruturas para medição também foram investigadas.

Nesta seção são apresentados os principais resultados do mapeamento sistemático realizado. A descrição completa do estudo encontra-se no Apêndice A.

Para realizar o estudo, foi utilizado o processo de apoio à condução de estudos baseados em revisão sistemática definido em (MONTONI, 2007), o qual é composto por três atividades:

(i) *Desenvolver o Protocolo:* nesta atividade o pesquisador realiza a prospecção sobre o tema de interesse do estudo, definindo o contexto e o objeto de análise. Em seguida, o protocolo que será o guia para execução do estudo é definido, testado e avaliado. O protocolo deve conter todas as informações necessárias para executar a pesquisa (questões de pesquisa, critério para seleção das fontes, critério para seleção das publicações, procedimentos para armazenar e analisar os resultados, e assim por diante). O protocolo deve ser testado a fim de verificar sua viabilidade, ou seja, se os resultados obtidos são satisfatórios e se a execução do protocolo é viável em termos de tempo e esforço. Os resultados do teste permitem a melhoria do protocolo quando necessária. Se o protocolo é viável, um especialista deve avaliá-lo e, uma vez aprovado, o protocolo pode ser usado para conduzir a pesquisa.

(ii) *Conduzir a Pesquisa:* nesta atividade o pesquisador executa o protocolo definido e, assim, seleciona, armazena e realiza análises quantitativas e qualitativas dos dados coletados.

(iii) *Relatar Resultados:* nesta atividade o pesquisador empacota os resultados gerados ao longo da execução do estudo, devendo ser publicado em alguma conferência, revista ou biblioteca de trabalhos científicos.

O objetivo do estudo foi analisar os registros da literatura no contexto de arquiteturas para medição de software, com o propósito de identificar e analisar: (i) as propostas de arquiteturas para medição de software; (ii) as características dessas propostas, incluindo as tecnologias empregadas; e (iii) se as propostas são capazes de apoiar o controle estatístico de processos.

Seguindo o protocolo de pesquisa definido, a pesquisa foi realizada nas bibliotecas digitais IEEE[2] e Scopus[3], tendo sido inicialmente selecionadas 148 publicações. Após análise seguindo os critérios de seleção estabelecidos no protocolo de pesquisa, 12 publicações foram consideradas realmente úteis ao estudo, sendo descritas 8 propostas de arquiteturas. A Tabela 2.1 lista as propostas encontradas juntamente com uma breve descrição.

Tabela 2.1 - Propostas encontradas.

Proposta	Descrição
P01 - *Framework* Genérico de Medição Baseado em MDA (MORA *et al.*, 2008; MORA *et al.*, 2009; MORA *et al.*, 2010; MORA *et al.*, 2011)	*Framework* de medição de software para suporte à medição de entidades de software através de metamodelos e transformações. Por exemplo, dado um modelo de um diagrama ER (Entidades e Relacionamentos), medidas como quantidade de tabelas e quantidade de relacionamentos podem ser calculadas automaticamente usando o framework. Para isso, o *framework* utiliza um modelo do domínio e um modelo de medição, que diz quais entidades serão medidas e usando quais métodos. Esses modelos passam por processamento de transformação QVT (*Query View Transformation*), que gera as medidas. A transformação leva em consideração um metamodelo do domínio e um metamodelo de medição que é reflexo de uma ontologia de medição.
P02 - WebEv (*Web for the Evaluation*) (AVERSANO e BODHUIN, 2004; AVERSANO *et al.*, 2004)	Sistema que utiliza um *framework* de medição baseado no GQM (*Goal Question Metric*) (BASILI *et al.*, 1994) para avaliação de processos de negócio e dá apoio à coleta, armazenamento e análise de medidas. Foi definido em termos de medidas, mecanismos de coleta de dados e guia para usar os dados coletados.
P03 - NSDIR (*National Software Data and Information Repository*) (GOTH, 2001)	Consiste em um repositório de *benchmarking* organizacional para projetos de software da Força Aérea Americana. Foi operacional de 1994 a 1998. Apesar de o uso do NSDIR pela Força Aérea Americana ter acabado em 1998, os esforços da Academia e Indústria foram continuados pelo CeBASE (*Center for Empirically-Based Software Engineering*).
P04 - MRS (*Measurement Repository System*) (BASTANI *et al.*, 2000)	É um repositório de medidas usado por um grupo de empresas de telecomunicação. Um dos propósitos era a avaliação de suprimentos e produtos através da geração de relatórios que compilavam os dados de todas as empresas participantes. Como grande preocupação o repositório tem a segurança e privacidade das informações.
P05 - MMR Tool (PALZA *et al.*, 2003)	Proposta de um repositório genérico e flexível de medição para coleta, armazenamento, análise e publicação de dados de medição de software. Foi projetado para dar suporte a todos os níveis do CMMI e foi aplicado na *Ericsson Research* Canadá.
P06 - SPDW+ (*Software Development Process Performance Data Warehousing*) (SILVEIRA *et al.*, 2010)	Apresenta a arquitetura de *data warehousing* SPDW+ como solução de repositório centralizado de medição, coleta automática de medidas de software e mecanismos de análise. O SPDW+ é um aprimoramento do SPDW que foi operacional por 3 anos na HP Brasil. Foi desenvolvido visando ao apoio da melhoria de processos em organizações maduras.

[2] http://ieeexplore.ieee.org
[3] http://www.scopus.com

Tabela 2.2 - Propostas encontradas (Continuação).

Proposta	Descrição
P07 - A Universal Metrics Repository (HARRISON, 2004)	Propõe uma estrutura para um repositório de medidas flexível, capaz de se adaptar a diferentes modelos de ciclo de vida, metodologias e processos de desenvolvimento de software. A proposta usa os conceitos de visão transformacional de desenvolvimento de software, que considera que o processo de desenvolvimento de software é uma serie de transformações de artefatos.
P08 - PropostaPAU (PAUL et al., 1999)	Apresenta um framework genérico que incorpora banco de dados, um conjunto de medidas relacionadas a testes e avaliação de software, bem como um conjunto de técnicas analíticas para a extração de informações e conhecimento. A abordagem propõe usar esse framework e suas técnicas para extrair informações detalhadas e conhecimento a partir das bases de medidas de software.

Em relação às características das propostas, decidiu-se por organizá-las em categorias, a fim de prover melhor entendimento e facilitar análises comparativas entre as propostas. As categorias definidas foram: tecnologia, arquitetura, coleta, armazenamento, análise e apoio ao CEP. Quanto ao apoio ao CEP, foi registrado "Sim" para as propostas cujas publicações deixam explícito que há apoio ao CEP. Para propostas cujo apoio ao CEP não está explícito no texto da publicação, mas, aparentemente, a proposta é capaz de apoiá-lo, registrou-se "Provavelmente Aplicável". Para propostas cuja descrição não menciona apoio ao controle estatístico de processos e não foi possível identificar esse apoio registrou-se "Não".

Vale mencionar que as publicações descrevem suas propostas com diferentes níveis de detalhes e com diferentes focos. Consequentemente, o nível de detalhe das informações sobre as características das propostas também é heterogêneo. Por exemplo, algumas propostas descrevem em detalhes características da arquitetura adotada, enquanto outras apenas citam o modelo geral no qual a arquitetura se baseia ou, ainda, nada dizem a respeito disso. Levando-se em conta as características das propostas, a Tabela 2.3 mostra um resumo dos achados. Em seguida são apresentadas algumas discussões sobre as propostas encontradas. Cabe ressaltar que não foi propósito do estudo comparar as propostas encontradas e determinar qual é a melhor (ou pior) delas, mas fornecer uma visão geral das propostas encontradas e suas principais características.

Tabela 2.3 - Visão geral das características das propostas identificadas.

	Características					
Proposta	**Tecnologia**	**Arquitetura**	**Coleta**	**Armazenamento**	**Análise**	**Apoio ao CEP**
P01	Uso de DSL (*Domain-Specific Language*) e ferramentas baseadas na plataforma Eclipse.	Baseada em MDA (*Model Driven Architecture*).	Automática (por meio de transformação de modelos).	Arquivos XML.	-	Não.
P02	Utilização de Java (Java JDBC e Java Servlet API).	Único sistema *web* que é responsável por receber e apresentar as medidas.	Semiautomática (via formulário *web*).	Banco de dados.	Recursos de análise quantitativa.	Provavelmente aplicável.
P03	Sun Solaris Unix, Oracle e cliente em Visual Basic com ODBC (*Open Database Connectivity*).	Repositório central que armazena dados coletados por software cliente.	Manual e Semiautomática (por meio de formulários físicos ou eletrônicos).	Banco de dados.	Ferramentas de análise estilo *benchmark*.	Não.
P04	-	Repositório central que armazena dados coletados por software cliente.	Semiautomática (por meio de formulário eletrônico).	Banco de dados.	Geração de relatórios trimestrais.	Não.
P05	Uso de tecnologias e ferramentas Microsoft (SQL 2000 Server, Analysis Services Enterprise Edition, Internet Information Server, Intranet Share Portal Server, ASP)	Baseada em ambientes de data *warehouse*.	Semiautomática, mas prevê o uso de ETL (*Extraction, Transformation and Loading*) para coleta de dados volumosos e periódicos.	*Data warehouse*. O modelo do banco é genérico para flexibilidade dos dados.	Consultas SQL (*Structured Query Language*) e OLAP (*On-line Analytical Processing*) cubes. Dados são apresentados via portal *web*. Possibilidade de exportar dados para ferramentas de estatística.	Sim.
P06	Uso de tecnologias e ferramentas Microsoft. (SQL Server 2005, BI Studio, Visual Studio 2005, SQL Server Integration Services and IIS 6.0)	Orientada a serviços (SOA – *Service Oriented Architecture*) e baseada em ambientes de *data warehouse* com 4 componentes.	Semiautomática e automática, por meio do uso de ETL.	*Data warehouse*.	Uso de ferramentas de BI (*Business Intelligence*) com interface *web*, incluindo OLAP e *dashboard*.	Sim.

Tabela 2.4 - Visão geral das características das propostas identificadas (Continuação).

	Características					
Proposta	**Tecnologia**	**Arquitetura**	**Coleta**	**Armazenamento**	**Análise**	**Apoio ao CEP**
P07	Usa MySQL (apenas o repositório é implementado).	-	-	Banco de dados. O modelo do banco é genérico para flexibilidade dos dados.	-	Não.
P08	-	-	Semiautomática.	Banco de dados.	Uso de histogramas, emprego de estatística básica e técnicas como: *Multiresolution Analysis, Classification Trees*, *Neural Networks* e *Influence Diagrams*.	Não.

De uma maneira geral, as propostas encontradas são bem distintas umas das outras e, na maioria das vezes, não é possível realizar comparações substanciais entre elas baseando-se nas informações contidas nas publicações. A seguir são apresentadas algumas considerações sobre as características de cada categoria analisada nas propostas.

- *Tecnologia:* É possível perceber que não há homogeneidade na utilização de tecnologias nas propostas, variando da utilização de softwares livres a tecnologias proprietárias. Isso pode ser reflexo da grande variedade de soluções tecnológicas existentes no mercado.
- *Arquitetura:* Todas as propostas, exceto o *Framework* Genérico de Medição Baseado em MDA (MORA *et al.*, 2008; MORA *et al.*, 2009; MORA *et al.*, 2010; MORA *et al.*, 2011), incluem como parte de sua arquitetura um repositório central para armazenamento e recuperação de dados, usando uma arquitetura cliente-servidor. As propostas MRS (BASTANI *et al.*, 2000) e NSDIR (GOTH, 2001) possuem programas clientes específicos para comunicação com o servidor. WebEv (AVERSANO e BODHUIN, 2004; AVERSANO *et al.*, 2004), MMR Tool (PALZA *et al.*, 2003), e SPDW+ (SILVEIRA *et al.*, 2010), por sua vez, utilizam recursos *web*. As propostas SPDW+ (SILVEIRA *et al.*, 2010) e MMR Tool (PALZA *et al.*, 2003) possuem arquiteturas baseadas em ambientes de *data warehouse,* incluindo um componente para coleta dos dados (ETL), um componente de armazenamento (*data warehouse*) e um componente para análise com funcionalidades analíticas (OLAP). A SPDW+ (SILVEIRA *et al.*, 2010) inclui um quarto componente responsável pela integração de dados, que atua como um repositório temporário para padronização dos dados coletados. O *Framework* Genérico de Medição Baseado em MDA (MORA *et al.*, 2008; MORA *et al.*, 2009; MORA *et al.*, 2010; MORA *et al.*, 2011) é uma arquitetura conceitual e uma adaptação do MDA. Ele é dividido em níveis que vão do MOF (*Meta-Object Facility*) aos dados das medidas, passando por um metamodelo de medição elaborado baseado em uma ontologia de medição de software.
- *Coleta:* É possível observar três tipos de coleta: manual, semiautomática e automática. A coleta manual refere-se ao uso de formulários físicos nos quais as pessoas registram os dados coletados de medição. Na coleta semiautomática os dados também são registrados por pessoas, mas conta com apoio computacional (por exemplo, formulários e planilhas eletrônicos e sistemas de informação). A coleta automática refere-se ao uso de ferramentas computacionais e mecanismos

que obtêm dados de medição sem intervenção humana. Nesse sentido, a maior parte das propostas usa coleta semiautomática. As publicações que descrevem as propostas MMR Tool (PALZA *et al.*, 2003) e MRS (BASTANI *et al.*, 2000) mencionam a intenção de usar mecanismos automatizados de coleta nessas propostas, mas esses mecanismos não são apresentados nas publicações. Apenas em duas propostas a automatização da coleta foi implementada: no *Framework* Genérico de Medição Baseado em MDA, (MORA *et al.*, 2008; MORA *et al.*, 2009; MORA *et al.*, 2010; MORA *et al.*, 2011), por meio da transformação de modelos, e no SPDW+ (SILVEIRA *et al.*, 2010), por meio do componente ETL. Vale ressaltar que essas propostas lidam com tipos bem específicos de medidas (por exemplo, quantidade de tabelas e de relacionamentos em um dado modelo de dados e número de erros em uma porção de código fonte), as quais são mais propícias para a coleta automática. Propostas que lidam com medidas cuja automatização da coleta é mais difícil adotam a coleta semiautomática. Isso pode ser visto como um indício da dificuldade e, em alguns casos impossibilidade, de se adotar a coleta automática de medidas. Somente a proposta NSDIR (GOTH, 2001) usa a coleta manual e os dados coletados em formulários físicos são, depois, registrados em formulários eletrônicos.

- *Armazenamento:* as propostas usam três diferentes soluções de armazenamento: bancos de dados relacional, como em WebEv (AVERSANO e BODHUIN, 2004; AVERSANO *et al.*, 2004), arquivos XML (*eXtensible Markup Language*), no *Framework* Genérico de Medição Baseado em MDA (MORA *et al.*, 2008; MORA *et al.*, 2009; MORA *et al.*, 2010; MORA *et al.*, 2011), e soluções baseadas em bancos de dados (*data warehouse*) como em SPDW+ (SILVEIRA *et al.*, 2010). Apesar de a maioria das propostas adotarem soluções baseadas em bancos de dados relacionais, observou-se que cada uma permite o armazenamento de um conjunto diferente de dados da medição. Acredita-se que isso ocorre principalmente porque a estrutura de armazenamento (o esquema do banco de dados) é definida a partir da especificação de quais entidades e elementos serão medidos e quais as necessidades de informação que se espera que os dados atendam. Notou-se, ainda, que algumas propostas proveem flexibilidade em relação a quais dados de medição podem ser armazenados. Por exemplo, na proposta MMR Tool (PALZA *et al.*, 2003), é utilizada uma estrutura de metanível do domínio de medição como modelo de dados com o propósito de permitir adaptação a diferentes contextos de medição.

Por outro lado, o *Universal Metrics Repository* (HARRISON, 2004) é por si só uma proposta de banco de dados flexível que visa armazenar quaisquer dados de quaisquer medidas relacionadas a diversas entidades. Por fim, observou-se que as propostas que incluem apoio ao controle estatístico de processos (SPDW+ (SILVEIRA *et al.*, 2010) e MMR Tool (PALZA *et al.*, 2003)) adotam soluções baseadas em *data warehouse* para o armazenamento dos dados.

- *Análise:* A maior parte das propostas encontradas possui algum mecanismo para análise ou apresentação dos dados. Algumas propostas, tais como SPDW+ (SILVEIRA *et al.*, 2010) e PAU (PAUL *et al.*, 1999), possuem mecanismos e ferramentas mais complexas. A análise pode ser puramente qualitativa, como em WebEv (AVERSANO e BODHUIN, 2004; AVERSANO *et al.*, 2004), ou ter caráter de *benchmarking*, como em NSDIR (GOTH, 2001) e MRS (BASTANI *et al.*, 2000), onde dados gerais de projetos e produtos podem ser analisados para a identificação de boas práticas. As propostas que apoiam o controle estatístico de processos (SPDW+ (SILVEIRA *et al.*, 2010) e MMR Tool (PALZA *et al.*, 2003)) adotam recursos para análise mais sofisticados (ambas fazem uso de ferramentas OLAP).
- *Apoio ao CEP*: A maior parte das propostas identificadas não fornece apoio ao controle estatístico de processos. Por exemplo, a proposta NSDIR (GOTH, 2001) inclui um repositório que armazena dados gerais sobre projetos e produtos com a principal finalidade de serem usados como *benchmarking*. Não são armazenados dados sobre a definição dos processos ou de sua execução, o que não permite a realização do CEP. Apenas duas propostas (SPDW+ (SILVEIRA *et al.*, 2010) e MMRTool (PALZA *et al.*, 2003)) incluem apoio ao CEP. Ambas foram concebidas no contexto de duas grandes empresas que buscavam os níveis mais altos de maturidade. Como comentado nas categorias anteriores, essas duas propostas são baseadas em ambientes de *data warehouse* e utilizam tecnologias Microsoft.

2.6. Considerações Finais do Capítulo

Neste capítulo foi apresentada a fundamentação teórica relevante para este trabalho, incluindo aspectos relacionados à medição de software, controle estatístico de processos, ontologias, arquiteturas de referência e arquiteturas para medição de software.

Durante a investigação da literatura, percebeu-se que quando se fala em arquiteturas para medição de software, as propostas são comumente apresentadas com foco nos

repositórios de medição e em aspectos tecnológicos. Algumas vezes, a arquitetura de medição chega a se confundir com a arquitetura do repositório de medição. Isso ocorre devido à grande importância desse componente para a realização da medição de software, uma vez que ele é o responsável pelo armazenamento e recuperação dos dados, que são a matéria-prima da medição.

Apesar de haverem propostas para arquiteturas de medição na literatura, conforme apresentado na seção 2.5 deste capítulo, essas propostas não são descritas em suficiente nível de detalhe para permitir sua reutilização por parte de organizações que desejam implementar uma solução computacional para medição. Além disso, várias propostas focam nas tecnologias que podem ser utilizadas e não da estrutura dos componentes. Dumke e Ebert (2010), por exemplo, afirmam que a arquitetura de um repositório de medição pode ser baseada em tecnologias como *data warehouse, data mining,* OLAP (*On-line Analytical Processing*), mediadores e serviços.

Arquiteturas de referência fornecem um meio de capturar a essência das arquiteturas para um referido domínio e podem ser utilizadas como base para a definição de arquiteturas para soluções computacionais específicas.

No próximo capítulo é apresentada a arquitetura de referência para medição de software proposta neste trabalho.

Capítulo 3
Arquitetura de Referência para Medição de Software

Este capítulo apresenta a abordagem para definição de arquiteturas e descreve a arquitetura de referência para medição de software proposta.

3.1 Introdução

A prática da medição de software nas organizações envolve definir medidas, coletar e analisar dados para essas medidas com o propósito de apoiar a tomada de decisão. Além de normas e padrões que apoiam a definição do processo de medição e orientam sobre suas práticas, os recursos tecnológicos também são importantes para o sucesso de sua implementação.

Em uma organização, a escolha das soluções tecnológicas deve estar alinhada às necessidades organizacionais e serve como base para a execução dos processos (LAUDON e LAUDON, 2011). Dessa forma, soluções computacionais podem ser utilizadas para apoiar a execução do processo de medição. De fato, considerando a natureza das atividades de medição, o apoio computacional para sua realização é praticamente indispensável. Uma solução computacional pode ser descrita por meio de uma arquitetura que define um conjunto de componentes relacionados.

Como discutido no Capítulo 2, arquiteturas de referência podem ser utilizadas como base para a definição de soluções computacionais específicas. Quando se trata de uma definição de arquitetura de referência, os principais conceitos do domínio devem ser abordados em uma descrição clara dos componentes da arquitetura. A apresentação dos conceitos e dos componentes comumente é feita por meio de modelos/diagramas, assim as informações podem ser transmitidas de maneira objetiva e estruturada. Um ponto de partida para a elaboração de uma arquitetura de referência são as ontologias de domínio, pois são capazes de representar os conceitos do domínio de forma estruturada e sem ambiguidade (NAKAGAWA *et al.*, 2009).

Neste capítulo a arquitetura de referência para medição de software proposta neste trabalho é apresentada. Antes de desenvolver a arquitetura de referência propriamente dita, foi definida uma abordagem para definição de arquiteturas de referência, que foi seguida

neste trabalho. Essa abordagem é apresentada na seção 3.2. Na seção 3.3, é apresentada uma visão geral da arquitetura de referência e algumas informações sobre seu desenvolvimento. Nas seções 3.4 e 3.5 os componentes da arquitetura são detalhados e, finalmente, na seção 3.6 são apresentadas as considerações finais do capítulo.

3.2 Abordagem em Níveis para Definição de Arquiteturas de Referência

A necessidade de se dar especial atenção aos modelos no contexto do desenvolvimento de software tem sido reconhecida. Uma iniciativa em linha com essa afirmação é o paradigma de Engenharia Dirigida por Modelos (MDE – *Model Driven Engineering*) (FONDEMENT e SILAGHI, 2004), que advoga pelo uso de modelos em níveis de abstração diferentes e transformações dos modelos de um nível a outro para que um sistema seja desenvolvido. Uma abordagem proposta nesse contexto é a MDA (*Model Driven Architecture*) (OMG, 2003), onde um modelo independente de plataforma (PIM – *Plataform Independent Model*) é construído e usado como base para, a partir de transformações, gerar modelos de plataformas específicas (PSM – *Plataform Specific Model*), os quais são utilizados para a implementação de sistemas.

Com inspiração no paradigma de Engenharia Dirigida por Modelos (MDE), a abordagem para definição de arquiteturas de referência proposta e utilizada neste trabalho é organizada em níveis, onde em cada nível há um modelo em um nível de abstração diferente, do mais abstrato para o menos abstrato. Ainda, inspirada na abordagem de Arquitetura Dirigida por Modelos (MDA), a proposta faz distinção entre modelos independentes de plataforma e modelos de plataformas específicas. Mas, é importante ressaltar que, apesar de a abordagem definida ser inspirada em MDE e MDA, este trabalho não é uma proposta de MDE ou MDA e a definição das transformações que levam de um modelo para outro não fazem parte de seu escopo.

A abordagem definida é composta por cinco níveis, como mostra a Figura 3.1. Nela, as caixas retangulares representam os níveis da abordagem e as caixas com as bordas arredondadas localizadas no interior dos níveis representam os elementos que estão presentes em cada um deles. O nível de abstração diminui de cima para baixo. As setas indicam que o modelo de um nível é usado como base para a definição de modelo do nível seguinte. Após a figura a descrição de cada nível é apresentada.

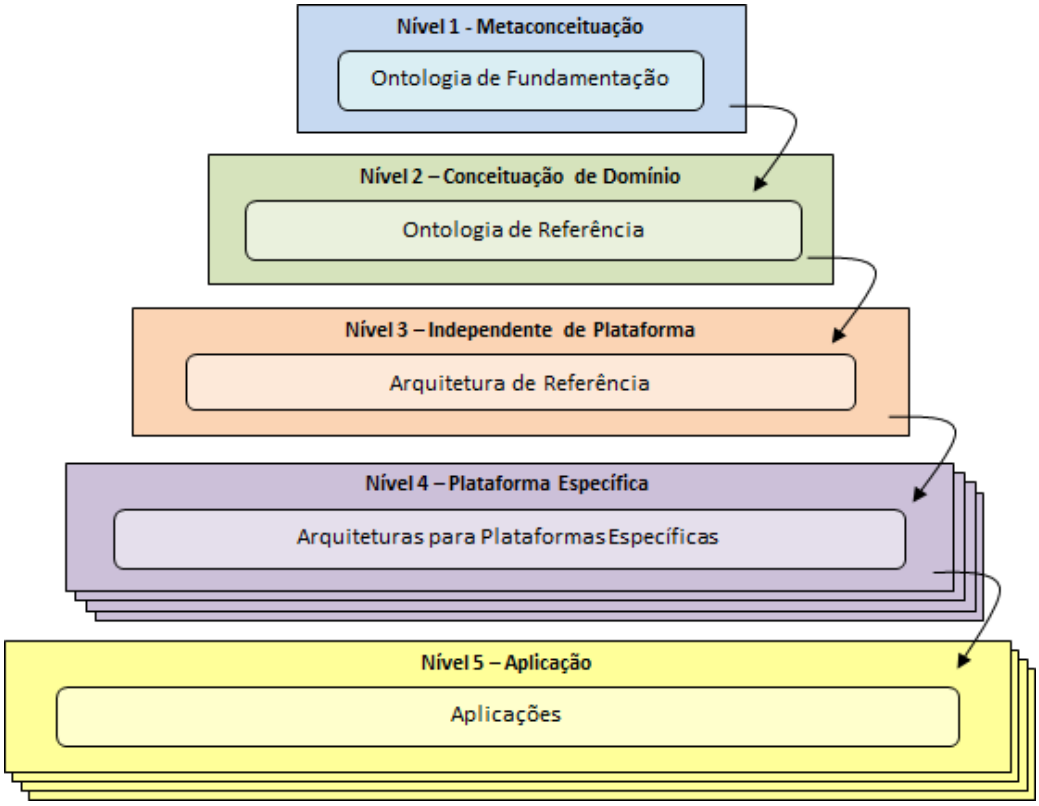

Figura 3.1 - Visão geral da abordagem para definição da arquitetura para medição de software.

O primeiro nível da abordagem (**Metaconceituação**) diz respeito a 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, cuja função é prover a conceituação do domínio no qual a arquitetura de referência será aplicada Uma vez que a ontologia de domínio deve descrever o domínio da melhor maneira possível, ela deve ser uma ontologia de domínio de referência, ou seja, uma ontologia construída com o objetivo de fazer a melhor descrição possível de um domínio na realidade, em relação a um certo nível de granularidade e ponto de vista. Uma ontologia de domínio de referência é um tipo especial de modelo representando um modelo de consenso em uma comunidade. 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 ontologia de domínio de referência deve ser definida com base na ontologia de fundamentação do nível anterior. 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. Ao se descrever uma arquitetura de referência, devem ser apresentados os componentes da arquitetura e seus papéis, os modelos que os descrevem, bem como qualquer outro artefato que auxilie na descrição da arquitetura de referência em um nível conceitual.

O quarto nível (**Plataforma Específica**) diz respeito a modelos elaborados acrescentando-se aspectos tecnológicos ao modelo definido no nível anterior. Esses modelos são derivados a partir dos modelos conceituais definidos no terceiro nível, levando-se em consideração a utilização de tecnologias específicas. Além disso, alguns requisitos específicos também podem ser considerados, o que pode levar, por exemplo, à não utilização de algumas partes da arquitetura de referência ou à necessidade de tratamento de aspectos não abordados na arquitetura de referência e que são necessários para a organização para a qual a arquitetura específica será definida. Assim, diferentes do nível anterior, modelos deste nível consideram, por exemplo, quais tecnologias de banco de dados serão utilizadas para implementar uma solução computacional para um determinado domínio. Consequentemente, os modelos estruturais elaborados no nível anterior devem ser adaptados de acordo com as tecnologias de banco de dados adotadas.

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 incluídas na abordagem, pois podem ser utilizadas para avaliar a arquitetura de referência proposta (MULLER, 2013).

O termo aplicação adotado no último nível da abordagem diz respeito às mais diversas soluções computacionais que podem ser produzidas. Uma ferramenta ou uma solução computacional envolvendo vários sistemas são exemplos de aplicações que poderiam ser produzidas nesse nível.

A abordagem descrita nesta seção foi utilizada para o desenvolvimento da arquitetura de referência proposta. 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 que foi descrita no Capítulo 2. 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. Essa ontologia também foi apresentada no Capítulo 2. Na próxima seção é apresentada uma visão geral da Arquitetura de Referência para Medição de Software proposta.

3.3 A Arquitetura de Referência para Medição de Software

Para definir a arquitetura de referência, o primeiro passo foi realizar um levantamento de informações a respeito do domínio de medição de software. Para isso, foram realizadas investigações da literatura (informal e formal) e a Ontologia de Referência para Medição de Software foi estudada em detalhes.

O próximo passo, então, foi definir a arquitetura propriamente dita. Nesse momento, foram definidos os requisitos que a arquitetura proposta deveria ser capaz de atender. São eles:

a) A arquitetura deve conter os elementos essenciais a uma solução computacional para medição de software;
b) 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;
c) A arquitetura deve apresentar em detalhes a estrutura do repositório de medição;
d) A arquitetura deve poder ser utilizada total ou parcialmente;
e) A arquitetura deve ser desenvolvida com base em uma ontologia de domínio bem fundamentada[4];
f) A arquitetura deve ser independente de tecnologias.

É possível notar que os requisitos e) e f) estão em linha com a abordagem de definição em níveis utilizada.

Após a definição desses requisitos, foi definido um modelo representando a visão geral dos componentes da arquitetura e, em seguida, esses componentes foram detalhados.

Após a arquitetura de referência ter sido definida, seguindo-se a abordagem em níveis, ela foi utilizada como base para a definição de uma arquitetura específica (nível 4 da abordagem) que, por sua vez, foi utilizada no desenvolvimento de uma aplicação de apoio à medição de software (nível 5 da abordagem), a qual foi submetida à avaliação. A instanciação e avaliação da arquitetura são discutidas no Capítulo 4.

[4] Que usa uma ontologia de fundamentação como base.

Observando-se o processo de definição de uma arquitetura de referência apresentado no Capítulo 2, é possível perceber que a definição da Arquitetura de Referência para Medição de Software ocorreu de forma alinhada àquele processo. A Figura 3.2 apresenta o processo de definição de uma arquitetura de referência descrito no Capítulo 2, acrescido de informações sobre o desenvolvimento da arquitetura de referência proposta neste trabalho.

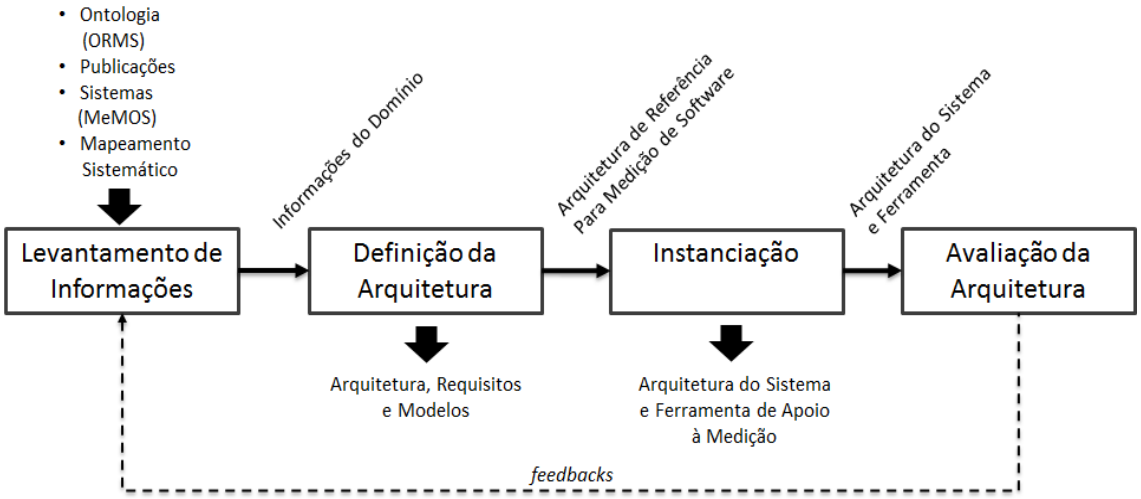

Figura 3.2 - Processo de elaboração da Arquitetura de Referência para Medição de Software (adaptado de (NAKAGAWA *et al.*, 2009)).

3.3.1. Visão Geral da Arquitetura de Referência

A Figura 3.3 apresenta a visão geral da Arquitetura de Referência para Medição de Software. Dados provenientes da organização, dos projetos e de seus artefatos são capturados por meio de utilitários de entrada e são armazenados em um repositório de medição. Esses dados são recuperados, analisados e os resultados são apresentados aos interessados por meio dos utilitários de saída[5].

[5] Neste trabalho, entende-se por utilitário qualquer ferramenta computacional que apoie a captura (utilitários de entrada) ou apresentação de dados (utilitários de saída).

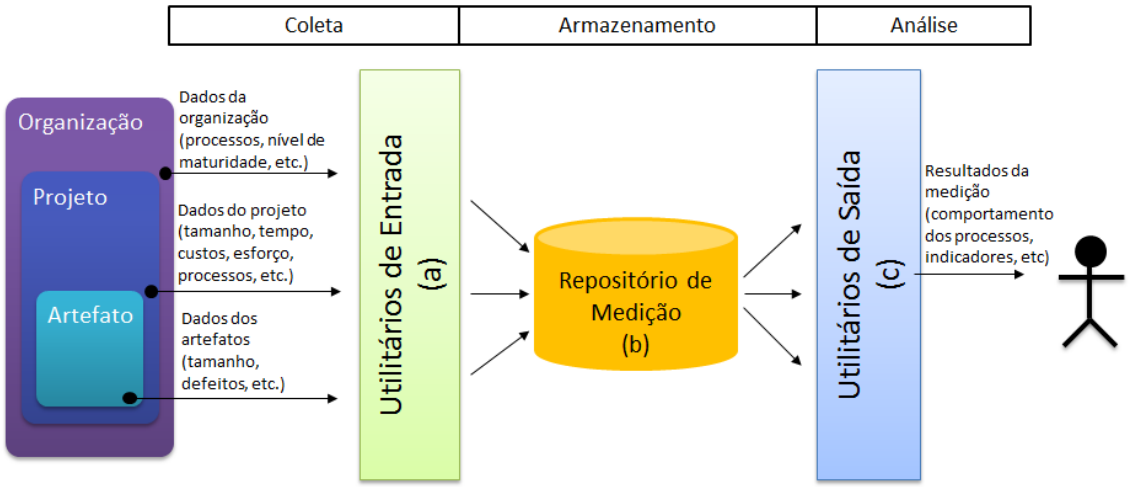

Figura 3.3 - Visão Geral da Arquitetura de Referência para Medição de Software.

Conforme mostra a Figura 3.3, a arquitetura de referência para medição é composta por três componentes:

(a) *Utilitários de Entrada*: componente responsável pela captura dos dados relevantes à medição de software, a saber: dados referentes à organização como um todo, aos projetos e aos artefatos produzidos nos projetos. Em uma solução computacional para medição de software, os utilitários de entrada devem possuir um conjunto de funcionalidades que permita a captura dos dados necessários à medição de software. Na arquitetura de referência, esse componente é descrito por meio de requisitos funcionais.

(b) *Repositório de Medição*: componente responsável pelo armazenamento dos dados de medição. Tem como base principal a Ontologia de Referência para Medição de Software e é descrito por meio de modelos de classes em UML (*Unified Modeling Language)*, descrições detalhadas desses modelos e restrições de integridade.

(c) *Utilitários de Saída:* componente responsável pela análise dos dados e apresentação dos resultados para apoio à tomada de decisão. É descrito por meio de requisitos funcionais.

A definição de quais componentes deveriam fazer parte da arquitetura de referência foi tomada levando-se em consideração o que se observou durante a investigação da literatura. Independentemente 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. Também notou-se que várias propostas consideram o armazenamento de dados oriundos de diversas fontes, 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. Por essa razão, 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 mais de uma ferramenta computacional.

A seguir, os componentes da arquitetura de referência são descritos em detalhes.

3.4 Utilitários de Entrada e Saída: Requisitos

Conforme dito na seção anterior, os componentes Utilitários de Entrada e Utilitários de Saída presentes na arquitetura de referência são descritos por meio de um conjunto de requisitos. Esses requisitos buscam representar as funcionalidades que são necessárias para que uma solução computacional seja capaz de apoiar adequadamente a execução do processo de medição de software, considerando tanto a medição tradicional quanto em alta maturidade.

A identificação dos requisitos baseou-se principalmente na Ontologia de Referência para Medição de Software (BARCELLOS, 2009; BARCELLOS *et al.*, 2013), em (ROCHA, *et al.*, 2012) e na documentação da ferramenta MeMOS (FRAUCHES, 2011). 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.

R1. Deve ser possível caracterizar projetos e identificar projetos similares.

A identificação de critérios que caracterizem os projetos de uma organização é indispensável para a identificação dos projetos similares, o que permitirá o agrupamento adequado de dados para a análise (KITCHENHAM *et al.*, 2006). Alguns exemplos de critérios de caracterização de projetos são: distribuição geográfica dos participantes do projeto; experiência da equipe do projeto em relação ao domínio, tecnologias e processos utilizados; domínio do software desenvolvido; modelo de ciclo de vida utilizado; e tipo de cliente. Uma vez que projetos sejam caracterizados, é necessário que exista um mecanismo que permita a identificação de projetos similares, considerando os critérios de caracterização utilizados (CARD, 2005; KITCHENHAM *et al.*, 2007).

R2. Deve ser possível registrar a definição dos processos, suas versões e identificar as versões utilizadas nos projetos.

Para que os dados coletados para um processo sejam utilizados para analisar seu desempenho, é necessário que eles sejam referentes a uma mesma definição desse processo. Dessa forma, deve ser possível registrar as definições dos processos e identificar as diferentes definições estabelecidas para um mesmo processo, ou seja, suas versões (TARHAN e DEMIRORS, 2006).

R3. Deve ser possível registrar as entidades que podem ser medidas (entidades mensuráveis), seus tipos e elementos que podem ser quantificados.

Para que a medição seja realizada, é necessário definir quais os tipos de entidades mensuráveis (por exemplo: Processo, Projeto e Artefato), as entidades mensuráveis (por exemplo, o Projeto X) e os elementos mensuráveis (propriedades das entidades mensuráveis, como, por exemplo, tamanho dos projetos) que se deseja medir.

R4. Deve ser possível registrar objetivos estratégicos relevantes à medição.

Para que a medição esteja em consonância com os objetivos organizacionais, o que é indispensável para que a medição seja realizada eficientemente, os objetivos de negócio relevantes para a medição devem ser registrados.

R5. Deve ser possível registrar objetivos de medição e identificar seus tipos e suas relações com objetivos estratégicos.

A partir dos objetivos estratégicos devem ser derivados objetivos de medição. A derivação de objetivos de medição pode ser feita diretamente a partir dos objetivos estratégicos ou, quando apropriado, a partir dos objetivos estratégicos podem ser derivados objetivos de software e, a partir destes, são estabelecidos os objetivos de medição (BRIMSON, 2004; KITCHENHAM *et al.*, 2006). Segundo (BARCELLOS *et al.*, 2013), objetivos de medição podem ser de três tipos: objetivos de monitoramento e controle dos projetos, objetivos de avaliação da qualidade dos produtos e processos e objetivos de análise de desempenho de processos. O primeiro e o segundo tipos requerem medição tradicional e, tipicamente, são definidos desde o início de um programa de medição. O terceiro tipo requer medição em alta maturidade, que inclui o controle estatístico de processos. Objetivos desse tipo são definidos quando práticas dos níveis mais altos de maturidade são realizadas.

R6. Deve ser possível registrar necessidades de informação identificadas a partir de objetivos de medição.

A partir dos objetivos de medição, devem ser identificadas as informações que são necessárias para analisar o alcance aos objetivos estabelecidos. Essas informações são fundamentais para a identificação das medidas a serem utilizadas.

R7. Deve ser possível registrar medidas.

As medidas são o instrumento básico para a realização da medição. O registro de uma medida inclui seu nome, descrição, mnemônico, tipo (base ou derivada), unidade de medida, tipo de escala, valores de escala e fórmula de cálculo (se aplicável). Também é necessário identificar o elemento e tipo de entidade mensurável quantificados pela medida. Medidas são definidas para atender necessidades de informação. Dessa forma, a relação entre a medida e a(s) necessidade(s) de informação por ela atendida(s) deve ser identificada.

R8. Deve ser possível estabelecer definições operacionais para medidas.

A definição operacional de uma medida descreve em detalhes aspectos relacionados à coleta e análise dos dados. A qualidade das definições operacionais é fator indispensável ao controle estatístico de processos, pois para analisar o comportamento dos processos é necessário que sejam utilizados dados coletados de maneira consistente. A consistência dos dados coletados depende da repetitividade da medição que, por sua vez, está diretamente relacionada com a completeza e precisão da definição operacional utilizada (KITCHENHAM *et al.*, 2001). Além das informações registradas na definição da medida, sua definição operacional deve incluir: procedimento de medição, papel do responsável (por exemplo, analista de sistemas, gerente do projeto) pela medição, momento da medição (em qual atividade a medição deve ser realizada, como, por exemplo, atividade de documentação de requisitos), periodicidade de medição (por exemplo, semanalmente, mensalmente, uma vez em cada ocorrência da atividade que descreve o momento de medição), procedimento de análise de medição, papel do responsável pela análise de medição, momento da análise de medição, periodicidade da análise de medição.

R9. Deve ser possível identificar medidas correlatas.

A identificação de medidas correlatas auxilia a análise dos dados coletados e, especificamente no contexto do controle estatístico de processos, apoia a investigação das causas de variação no comportamento dos processos, auxiliando a identificação das ações

corretivas adequadas (EICKELMANN e ANANT, 2003; CAIVANO, 2005). Exemplos de medidas correlatas são medidas que apresentam relações de causa e efeito e medidas utilizadas para compor outras.

R10. Deve ser possível registrar planos de medição da organização e do projeto.

O plano de medição reúne todas as informações relevantes para que a medição possa ser realizada. Ele consiste no resultado do planejamento de medição realizado para a organização ou para um projeto específico. Um plano de medição deve incluir: os objetivos estratégicos relevantes à medição, os objetivos de medição derivados a partir dos objetivos estratégicos, as necessidades de informação identificadas a partir dos objetivos de medição, as medidas a serem utilizadas para atender as necessidades de informação, a definição operacional a ser utilizada para cada medida, a pessoa responsável pela elaboração do plano e a data de criação do plano. Um plano de medição da organização contém o planejamento geral da medição em uma organização. O plano de medição de um projeto é definido a partir do plano de medição da organização, podendo ser igual ao plano de medição da organização ou conter diferenças para garantir o alinhamento com as necessidades do projeto para o qual foi definido.

R11. Deve ser possível registrar medições.

As medições produzem os dados que serão utilizados nas análises. Para cada medição realizada, é necessário registrar a data em que foi executada, a medida e definição operacional utilizadas, o valor medido, o executor da medição, a entidade medida, o momento da execução da medição e o contexto da medição, que descreve as condições sob as quais a medição foi realizada.

R12. Deve ser possível realizar e registrar análises de medições.

As análises de medições produzem os resultados que serão utilizados para apoiar o atendimento às necessidades de informação e alcance aos objetivos. Para realizar a análise, é necessário indicar a medida que se deseja analisar e os dados que se deseja considerar para a análise. Os dados selecionados devem ser plotados em gráficos e apresentados[6] para que seja possível obter o resultado da análise. Para cada análise realizada devem ser identificados: a data em que a análise foi feita, a medida e a definição operacional utilizadas, os dados considerados para análise, o executor da análise de medição, o momento da

[6] Embora R12 seja um requisito do componente Utilitários de Entrada, a apresentação do gráfico contendo os dados selecionados é uma responsabilidade do componente Utilitários de Saída (vide R20).

execução da análise de medição e o resultado da análise. Quando se trata de análise no contexto da medição tradicional, os dados devem ser plotados em gráficos de apoio à análise tradicional, como, por exemplo, gráfico de barras. Tratando-se de análise de comportamento de processos, os dados devem ser representados utilizando-se gráficos de controle, que são os métodos analíticos essenciais do controle estatístico de processos.

R13. Deve ser possível registrar as pessoas envolvidas em atividades de medição e os papéis por elas desempenhados.

Medições e análises de medição devem ser realizadas de acordo com as definições operacionais das medidas utilizadas. Dessa forma, idealmente, a pessoa que realiza uma medição ou análise de medição deve desempenhar o mesmo papel estabelecido para o responsável pela medição/análise de medição na definição operacional da medida utilizada. Para que essa verificação seja possível, devem-se registrar as pessoas envolvidas em atividades de medição e os papéis por elas desempenhados quando realizam as atividades de medição/análise.

R14. Deve ser possível analisar o comportamento dos processos e classifica-los como estáveis ou capazes.

A análise dos dados coletados para medidas que descrevem o desempenho de processos utilizando-se métodos do controle estatístico pode revelar se eles são estáveis e capazes. Dessa forma, deve ser possível identificar processos submetidos ao controle estatístico de processos como estáveis ou capazes. Um processo é estável ou capaz em relação a uma determinada medida. Ao se classificar um processo como estável ou capaz, deve-se, também, identificar a medida utilizada na análise do comportamento do processo.

R15. Deve ser possível registrar *baselines* de desempenho para os processos estáveis.

Um processo estável possui *baseline* de desempenho, a qual é estabelecida em relação à medida utilizada na análise de comportamento de processo que identificou o processo como estável. Uma *baseline* de desempenho é definida por três limites de controle (superior, central e inferior). Devem-se registrar, além da *baseline* propriamente dita, a data em que foi estabelecida e informações sobre o contexto em que ela foi definida (por exemplo: *baseline* estabelecida a partir de dados de 4 projetos desenvolvidos durante a implementação das práticas do nível B do MPS.BR na organização, sendo todos os

projetos simples e pequenos). A *baseline* de desempenho de um processo pode ser atualizada ao longo do tempo. Além disso, para um mesmo processo, podem ser estabelecidas *baselines* para contextos diferentes (por exemplo, uma para quando o processo é utilizado em projetos pequenos e simples, e outra para quando o processo é utilizado em projetos grandes e complexos). Consequentemente, um mesmo processo pode ter diversas *baselines* estabelecidas para ele. É importante registrar as *baselines* de desempenho dos processos, pois, uma vez que elas descrevem o desempenho dos processos, elas são utilizadas como referência para analisar se o comportamento dos processos nos projetos está de acordo com o desempenho descrito por sua *baseline*. Além disso, as *baselines* de desempenho são essenciais à análise da capacidade dos processos (vide requisito R17).

R16. Deve ser possível registrar o desempenho especificado para os processos.

Para determinar se um processo estável é capaz ou não, é necessário conhecer o desempenho esperado para o processo. Em outras palavras, é preciso conhecer o desempenho especificado para o processo, o qual é definido por limites (inferior e superior) que descrevem os resultados que se espera que o processo alcance em relação a uma determinada medida. O desempenho especificado para o processo pode ser revisto, podendo ser estabelecidos diversos desempenhos especificados para um mesmo processo ao longo do tempo.

R17. Deve ser possível determinar a capacidade dos processos.

A capacidade de um processo é dada por um índice de capacidade[7], que indica se o processo é capaz ou não. Ela é calculada comparando-se o desempenho do processo, descrito pelos limites da sua *baseline* de desempenho, com o desempenho especificado para o processo, descrito pelos limites do desempenho especificado. A capacidade de um processo é determinada em relação a uma medida, que deve ser a mesma medida relacionada ao desempenho especificado para o processo e à *baseline* de desempenho utilizadas no cálculo da capacidade.

[7] O procedimento para cálculo do índice de capacidade foi descrito no Capítulo 2.

R18. Deve permitir analisar o comportamento dos processos executados nos projetos em relação às *baselines* estabelecidas para o processo no âmbito organizacional.

No contexto dos projetos é necessário avaliar se os processos executados apresentam o comportamento para eles esperados, a fim de que alcancem os objetivos para eles estabelecidos. Para avaliar o comportamento de um processo no contexto de um projeto, a *baseline* de desempenho estabelecida para o processo no âmbito organizacional deve ser utilizada para descrever o desempenho esperado para o processo no projeto.

R19. Deve ser possível registrar modelos de desempenho.

Modelos de desempenho são expressos por fórmulas que estabelecem as relações quantitativas entre medidas (por exemplo, uma fórmula que estabelece a relação entre tamanho e esforço). São úteis, principalmente, para a realização de estimativas, apoiando o planejamento e o monitoramento de projetos. Um modelo de desempenho é estabelecido a partir de dados coletados para medidas em processos estáveis, pois para estabelecer relações quantitativas entre medidas, elas devem descrever um comportamento repetível.

R20. Deve ser possível apresentar resultados da medição por meio de relatórios, gráficos ou consultas.

Os dados da medição devem ser acessíveis por meio de consultas ou relatórios. Deve ser possível consultar informações básicas sobre os itens registrados (por exemplo, visualizar as informações contidas na definição operacional de uma medida), bem como obter informações consolidadas a partir dos dados registrados durante a execução do processo de medição. Dessa forma, devem ser apresentados, na forma de consulta ou relatório: os planos de medição da organização e dos projetos, as medições realizadas (medida e definição operacional utilizadas, data da medição, valor medido, executor da medição, momento da medição e contexto da medição) e as análises de medições (medida e definição operacional utilizadas, data da análise, executor da análise, momento da execução da análise, dados plotados em um gráfico e o resultado da análise). As consultas devem considerar opções de parâmetros para a seleção dos dados (por exemplo, consultar as medições realizadas para uma determinada medida, em um projeto em particular).

Além disso, durante a análise de medições (vide R12), os dados devem ser plotados em gráficos.

R21. Deve ser possível apresentar resultados da análise de comportamento de processos por meio de relatórios, gráficos ou consultas.

Os dados da análise de comportamento dos processos devem ser acessíveis por meio de consultas ou relatórios. Deve ser possível consultar informações básicas sobre os itens registrados (por exemplo, consultar o desempenho especificado para um dado processo), bem como obter informações consolidadas a partir dos dados registrados durante a análise de comportamento de processos. Dessa forma, devem ser apresentados, na forma de consulta ou relatório os dados relacionados ao comportamento de um dado processo, que inclui: o processo, sua classificação (instável, estável ou capaz), a medida e definição operacional consideradas, os valores medidos utilizados para descrever o comportamento do processo de desempenho plotados em um gráfico de controle, o resultado na análise, a *baseline* de desempenho do processo (se o processo for estável), o desempenho especificado considerado e a capacidade do processo (se o processo for estável).

3.5 Repositório de Medição

O repositório de medição é o componente responsável pelo armazenamento dos dados da medição. Esse componente é definido através de 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). Apesar de os conceitos e as relações dessa ontologia serem representados por meio de modelos de classes, esses modelos não são adequados para representar a estrutura de um repositório de medição, pois a ontologia é uma representação da conceituação do domínio de medição, enquanto que o repositório visa atender as necessidades relacionadas a dados para a realização do processo de medição apoiada por solução computacional. Assim, para elaborar os modelos de classes do repositório, foi realizado um mapeamento dos conceitos e relacionamentos da ontologia para classes e relacionamentos que possam ser utilizados como base para o projeto e implementação de um repositório de medição. Para a elaboração dos modelos de classes, também foram considerados os requisitos que descrevem os utilitários de entrada e saída que interagem com o repositório de medição.

De forma análoga à Ontologia de Referência para Medição de Software (BARCELLOS, 2009), a qual é dividida em sete subontologias, o repositório está

organizado em pacotes. Exceto pelas subontologias Entidades Mensuráveis e Medidas de Software que foram tratadas em um único pacote, cada subontologia da Ontologia de Referência para Medição de Software foi mapeada para um pacote.

Tendo sido percebida a necessidade de contemplar aspectos relacionados ao Plano de Medição que não foram abordados na ontologia, foi criado um novo pacote, chamado Plano de Medição. Esse pacote 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.

Além disso, outro novo pacote, chamado Caracterização de Projeto, foi incluído. Esse pacote é necessário para o atendimento ao requisito R1 (Deve ser possível caracterizar projetos e identificar projetos similares), apresentado na seção anterior. Apesar de a ontologia suportar a caracterização de projetos (através dos conceitos Entidade Mensurável, Tipo Entidade Mensurável, Elemento Mensurável, Medida e Valor Medido), isso não é explícito. Assim, decidiu-se tratar a caracterização de projetos de forma explícita, definindo-se classes e relacionamentos para isso.

Para complementar a organização dos pacotes que descrevem a estrutura do repositório de medição, os pacotes relacionados ao planejamento da medição foram agrupados em um pacote chamado Planejamento da Medição.

Além dos pacotes relacionados diretamente à medição, seguindo a Ontologia de Referência para Medição de Software, foram criados dois pacotes para tratar os aspectos relevantes à medição de software e relacionados a Processos de Software (GUIZZARDI, *et al.*, 2008b; BRINGUENTE *et al.*, 2011) e Organizações de Software (BARCELLOS e FALBO, 2009). A Figura 3.4 apresenta o diagrama de pacotes do repositório de medição. O pacote Comportamento de Processos de Software está relacionado à medição em alta maturidade, enquanto que os demais estão relacionados à medição tradicional. Vale notar que os pacotes relacionados à medição tradicional também são necessários à medição em alta maturidade.

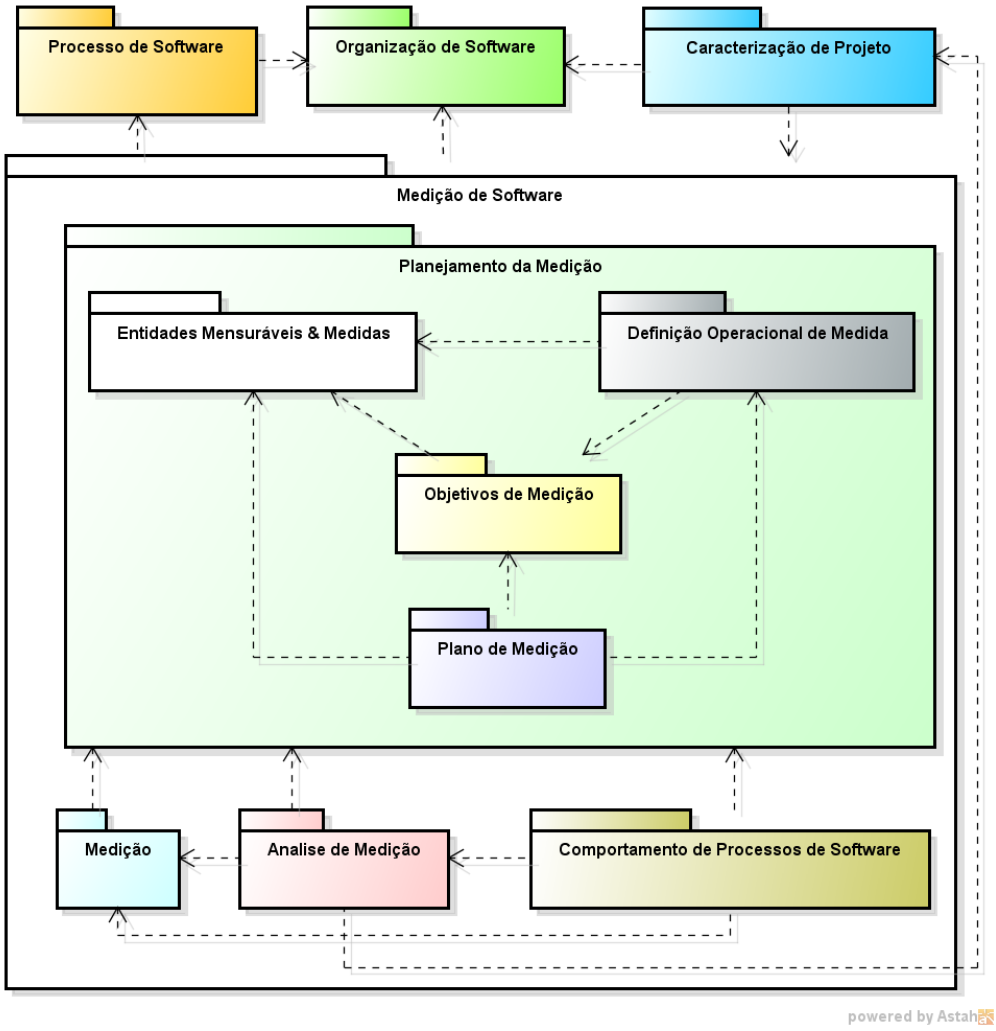

Figura 3.4 - Diagrama de Pacotes do Repositório de Medição.

A versão original da Ontologia de Referência para Medição de Software usada foi integralmente apresentada em (BARCELLOS, 2009), porém em publicações mais recentes (BARCELLOS *et al.*, 2010a), (BARCELLOS *et al.*, 2010b), (BARCELLOS *et al.*, 2010c), (BARCELLOS *et al.*, 2012), (BARCELLOS *et al.*, 2013) foram apresentadas algumas alterações que também foram consideradas neste trabalho. Por exemplo, a decisão de representar em um único pacote as subontologias Entidades Mensuráveis e Medidas de Software, baseou-se nas publicações mais recentes.

A seguir, são apresentados os modelos de classes do repositório de medição. Inicialmente, são apresentados os modelos dos pacotes referentes a Organizações de Software e Processo de Software. Em seguida, são apresentados os modelos e descrições dos pacotes referentes à medição de software. As descrições apresentadas para os modelos foram definidas com base nas descrições dos conceitos e relacionamentos presentes na ontologia e descritos em (BARCELLOS, 2009), preservando-se as particularidades dos modelos de classes do repositório de medição.

No texto, usa-se **negrito** para indicar a primeira ocorrência de nomes de classes ou de papéis, sublinhado para indicar atributos de classes e *itálico* para indicar possíveis instâncias das classes.

Nos modelos, são utilizadas cores diferentes (seguindo as cores apresentadas no diagrama de pacotes da Figura 3.4) para designar os pacotes de origem de cada classe apresentada. Buscando-se evitar que os modelos fiquem muito poluídos visualmente, há relacionamentos apresentados em alguns modelos que são omitidos em modelos posteriores.

No Apêndice B, para cada pacote descrito a seguir, são apresentadas tabelas que mostram o mapeamento entre os conceitos da ontologia e os modelos classes, bem como alguns comentários sobre cada mapeamento realizado.

3.5.1. Pacote Organização de Software

Trata de aspectos relacionados a organizações de software que são relevantes para a medição de software, como, por exemplo, projetos e suas equipes. Os conceitos são provenientes da Ontologia de Organizações de Software (BARCELLOS e FALBO, 2009). A Figura 3.5 apresenta o modelo de classes do pacote Organização de Software. Apesar de não aparecer na figura, a classe Projeto possui, além dos atributos apresentados, os atributos nome e descrição, os quais são definidos em uma relação de generalização/especialização presente no pacote Entidades Mensuráveis e Medidas, apresentado mais adiante.

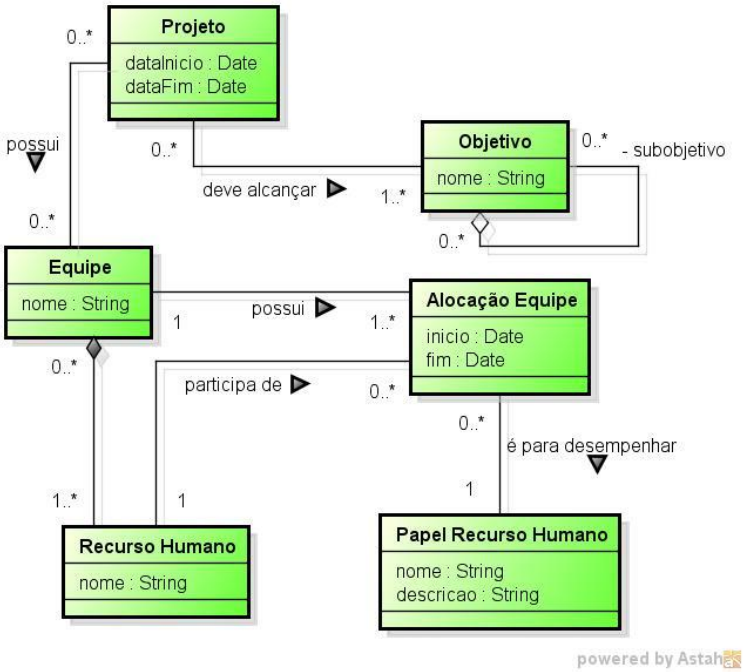

Figura 3.5 - Modelo de Classes do Pacote Organização de Software.

Um **Projeto** (por exemplo, o *Projeto* X) é criado para alcançar determinados **Objetivos.** Um Projeto possui **Equipes** (por exemplo, a *Equipe de Gestão da Qualidade do Projeto X*) formadas por **Recursos Humanos** (por exemplo, *João da Silva, Maria de Souza* e *Pedro dos Santos*). Um Recurso Humano pode estar alocado em um dado momento a uma Equipe (**Alocação Equipe**) desempenhando um determinado **Papel Recurso Humano**. Por exemplo, o recurso humano *João da Silva*, pode ser alocado no papel de *gerente de qualidade* na *Equipe de Gestão da Qualidade do Projeto X* no período de *01/01/2013* a *31/07/2013*. Uma Equipe somente pode possuir alocações para Recursos Humanos que fazem parte dela.

3.5.2. Pacote Processo de Software

Trata de aspectos relacionados aos processos de software e que são necessários para a medição de software. Os conceitos são baseados na Ontologia de Processos de Software (GUIZZARDI *et al.*, 2008a; GUIZZARDI, *et al.*, 2008b; BRINGUENTE *et al.*, 2011), tendo sido feitas adaptações para representar apenas os conceitos e relações indispensáveis à medição de software. Como resultado, apesar de ser suficiente no contexto da medição de software, o modelo desse pacote é limitado e não é capaz de atender as necessidades do domínio de processos de software (definição e gerência). A Figura 3.6 apresenta o modelo de classes do pacote Processo de Software. As classes desse pacote, destacadas em laranja, embora não apresentado na figura, possuem os atributos nome e descrição, os quais são

definidos em uma relação de generalização/especialização presente no pacote Entidades Mensuráveis e Medidas, que será apresentado mais adiante.

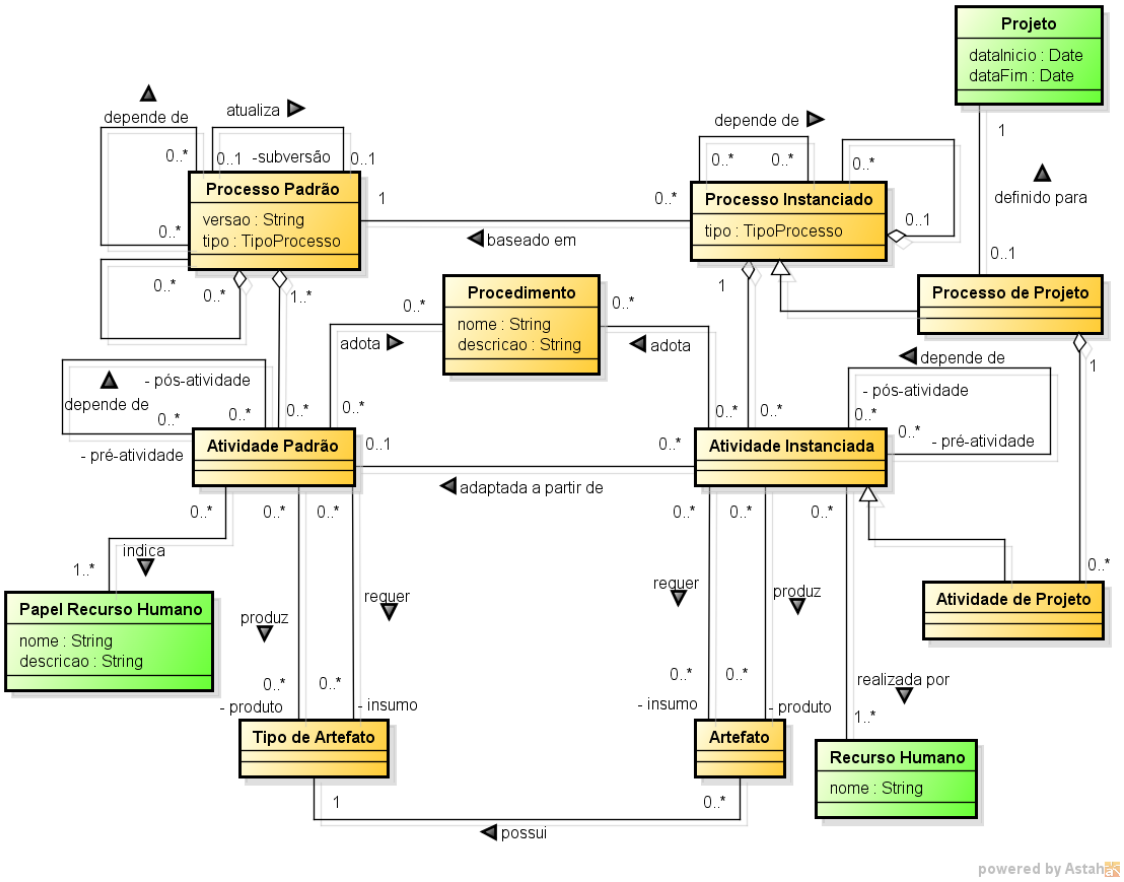

Figura 3.6 - Modelo de Classes do Pacote Processo de Software.

Um **Processo Padrão** é um processo de software que é definido no contexto de uma organização e serve de base para os processos de software a serem usados pela organização (por exemplo, o processo padrão de *Gerência de Requisitos*. Um Processo Padrão pode ser formado por outros processos padrão (tipoProcesso = complexo). Por exemplo, o *Processo Padrão PP1* pode ser formado pelos processos padrão de *Gerência de Requisitos*, *Gerência de Projetos, Implementação* e *Testes*). Um Processo Padrão simples (tipoProcesso = simples) é formado por **Atividades Padrão** (por exemplo, a atividade padrão *Realizar Levantamento de Requisitos*), que podem possuir uma relação de dependência com outras, ditas suas pré-atividades (por exemplo, a atividade padrão *Realizar Levantamento de Requisitos* pode ser pré-atividade da atividade padrão *Registrar Requisitos*). Uma Atividade Padrão pode requerer (**insumo)** ou produzir **(produto) Tipos de Artefatos** (por exemplo, *Especificação de Requisitos*). Uma atividade padrão pode, ainda, adotar um **Procedimento** (por exemplo, a atividade padrão *Realizar Levantamento de Requisitos* pode adotar o procedimento *entrevista*).

Com base nos Processos Padrão da organização, processos podem ser instanciados, a fim de serem executados. **Processos Instanciados** (por exemplo, um *Processo de Gerência de*

Portfólio definido com base no *Processo Padrão de Gerência de Portfólio* da organização), assim como os processos padrão, podem ser complexos ou simples. Um Processo Instanciado tem o mesmo tipo (complexo ou simples) que o processo no qual foi baseado. Processos Instanciados simples são formados por **Atividades Instanciadas**, as quais são adaptadas a partir de Atividades Padrão do Processo Padrão em que o Processo Instanciado foi baseado. Atividades Instanciadas podem requerer ou produzir **Artefatos** (por exemplo, um *Documento de Portfólio de Projetos*), que devem ser do mesmo Tipo de Artefato especificado na Atividade Padrão da qual a atividade foi adaptada. Analogamente, uma Atividade Instanciada pode adotar Procedimentos, os quais são os mesmos especificados na Atividade Padrão da qual a atividade foi adaptada. Quando um processo instanciado é um processo definido para um projeto, tem-se um **Processo de Projeto** (por exemplo, o processo de *Gerência de Requisitos definido para o Projeto X*). Processos de Projeto são compostos por **Atividades de Projeto** (por exemplo, a atividade *Realizar Levantamento de Requisitos definida para o Projeto X*), que são Atividades Instanciadas adaptadas a partir de Atividades Padrão que compõem o Processo Padrão utilizado como base.

O conceito Processo Instanciado, sendo um processo baseado em um Processo Padrão e que não está associado a um Projeto, não existe na Ontologia de Processo de Software, mas é necessário no contexto da medição de software, uma vez que medições e análises podem ser realizadas, também, fora do contexto dos projetos.

3.5.3. Pacote Entidades Mensuráveis & Medidas

Trata das entidades que podem ser medidas, suas propriedades que podem ser quantificadas e as medidas que podem ser usadas. A Figura 3.7 apresenta o modelo de classes do pacote Entidades Mensuráveis & Medidas. Para melhor visualização, alguns relacionamentos apresentados nos modelos anteriores foram omitidos.

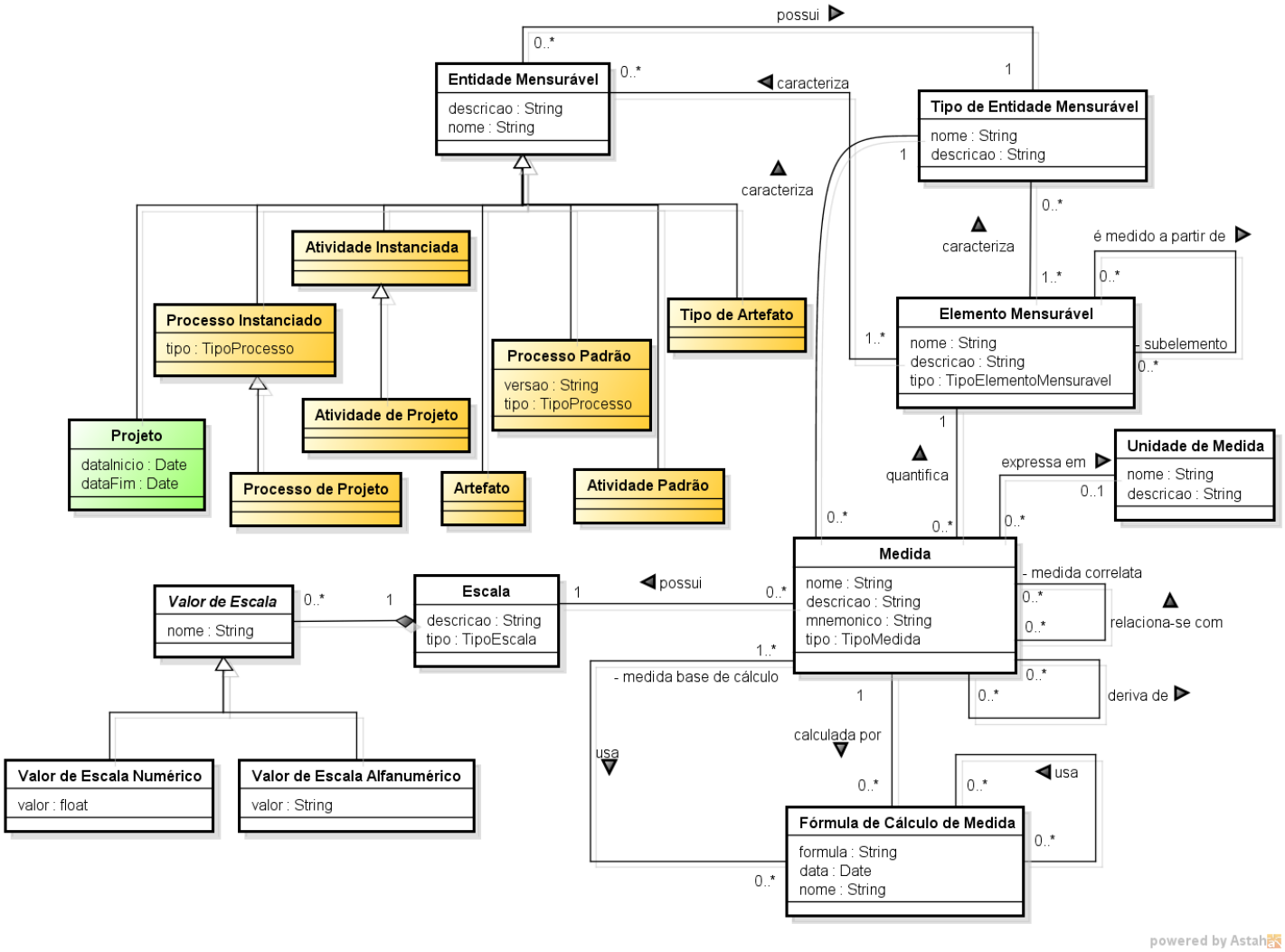

Figura 3.7 - Modelo de Classes do Pacote de Entidades Mensuráveis & Medidas.

Uma **Entidade Mensurável** é algo que pode ser medido, tal como um processo, um artefato e um projeto. Entidades Mensuráveis são classificadas de acordo com seu tipo (**Tipo de Entidade Mensurável**), que pode ser Projeto (por exemplo, o *Projeto X)*, Processo Padrão (por exemplo, o processo padrão de *Gerência de Requisitos*), Atividade Padrão (por exemplo, a atividade padrão *Realizar Levantamento de Requisitos*), Processo de Projeto (por exemplo, o processo de *Gerência de Requisitos definido para o Projeto X*), Atividade de Projeto (por exemplo, a atividade *Realizar Levantamento de Requisitos definida para o Projeto X*), Tipo de Artefato (por exemplo, *Especificação de Requisitos*) e Artefato (por exemplo, a *Especificação de Requisitos do Projeto X*)[8].

Tipos de entidades mensuráveis são caracterizados por **Elementos Mensuráveis,** que são propriedades que podem ser medidas. Por exemplo, *tamanho* é um elemento mensurável que pode ser usado para caracterizar uma entidade mensurável do tipo *Projeto*. Entidades mensuráveis são caracterizadas por elementos mensuráveis que caracterizam entidades do seu tipo. Por exemplo, a entidade mensurável *Projeto X* pode ser caracterizada pelo elemento mensurável *tamanho*, uma vez que entidades mensuráveis do tipo *Projeto* são caracterizadas por esse elemento mensurável.

Elementos Mensuráveis possuem um tipo (tipoElementoMensurável) que pode ser direta (por exemplo, *tamanho*) ou indiretamente mensurável (por exemplo, *produtividade*). Elementos indiretamente mensuráveis são medidos a partir da medição de outros elementos mensuráveis, ditos seus **subelementos.** Por exemplo, o elemento indiretamente mensurável *produtividade* pode ser medido a partir da medição dos subelementos *tamanho* e *esforço*.

Medidas quantificam elementos mensuráveis. Por exemplo, *tamanho* pode ser quantificado pelas medidas *número de linhas de código* e *número de pontos de função*. Medidas podem ser de dois tipos (tipoMedida): medida base, quando quantifica um elemento diretamente mensurável, medida derivada, quando quantifica um elemento indiretamente mensurável, sendo funcionalmente dependente de outras medidas. As medidas *número de linhas de código* e *número de defeitos detectados* são exemplos de medidas base que quantificam, respectivamente, os elementos diretamente mensuráveis *tamanho* e *defeitos detectados*. Já a medida *densidade de defeitos*, dada pela razão entre o número de defeitos detectados e o

[8] Neste trabalho, estão sendo considerados apenas alguns tipos de entidades mensuráveis (foram selecionados aqueles que foram considerados mais significativos à medição tradicional e CEP), mas outros tipos podem ser acrescidos à arquitetura do repositório de medição quando for definida a arquitetura para plataforma específica (nível 4 da abordagem), de acordo com as necessidades da organização para a qual a arquitetura será definida. Por exemplo, caso também se deseje mensurar recursos humanos e equipes, essas classes, já presentes na estrutura proposta para o repositório, seriam representadas como especializações de Entidade Mensurável.

número de linhas de código avaliadas, é um exemplo de medida derivada, que quantifica o elemento indiretamente mensurável *densidade de defeitos*. Medidas derivadas são calculadas por uma **Fórmula de Cálculo de Medida**, que usa outras medidas para determinar a medida derivada. Fórmulas de medida podem usar outras fórmulas. No exemplo apresentado, a fórmula de cálculo de medida é *densidade de defeitos = número de defeitos detectados/número de linhas de código avaliadas*.

Medidas possuem **Unidade de Medida.** *Linhas de código*, *horas* e *reais* são exemplos de unidades de medida. Algumas medidas não possuem unidade de medida. Por exemplo, a medida *taxa de correção de defeitos,* dada pela razão entre o *número de defeitos corrigidos* e o *número de defeitos detectados* não possui unidade de medida.

Medidas possuem uma **Escala**, que determina para quais valores uma medida pode ser mapeada. Escalas possuem um tipo (tipoEscala), que pode ser Nominal, Ordinal, Intervalar, Absoluta e Taxa. Escalas possuem **Valor de Escala**. Por exemplo, a escala da medida *nível de experiência da equipe com as tecnologias do projeto,* é uma escala Nominal cujos valores podem ser *Alto, Médio* e *Baixo.* Algumas escalas podem não ter seus valores de escala determinados. Por exemplo, a medida *número de defeitos detectados* tem uma escala do tipo Absoluta cujos valores são os números inteiros positivos, não havendo uma discriminação limitada para esses valores. Em casos como esse, os valores da escala não são determinados.

Medidas podem se relacionar com outras que, de alguma forma, influenciam em seu valor. Essas medidas são ditas **medidas correlatas**. Medidas com relações de causa e efeito, medidas relacionadas a um mesmo objetivo no plano de medição e medidas utilizadas em uma fórmula de cálculo de medida são exemplos de medidas correlatas.

Para cada modelo de classes dos pacotes que representam a estrutura do repositório de medição foram identificadas as restrições de integridade envolvidas. Os axiomas da Ontologia de Referência para Medição de Software foram utilizados como base para a identificação das restrições. Alguns axiomas foram mapeados em restrições de integridade equivalentes, no entanto, devido às diferenças entre os modelos da ontologia e os modelos do repositório, alguns axiomas deixaram de ser necessários e outros foram adaptados. Além disso, devido às particularidades dos modelos do repositório, algumas restrições não foram identificadas a partir dos axiomas. A seguir são apresentadas as restrições de integridade referentes ao pacote Entidades Mensuráveis & Medidas. Para cada restrição é indicado o axioma da Ontologia de Referência para Medição de Software utilizado como base, quando

for o caso. As restrições de integridade relacionadas aos demais pacotes não são apresentadas neste capítulo. Elas podem ser encontradas no Apêndice B.

3.5.4. Pacote Objetivos de Medição

Trata do alinhamento da medição de software com os objetivos estratégicos da organização, através da relação entre os objetivos, necessidades de informação e medidas. A Figura 3.8 apresenta o modelo de classes do pacote Objetivos de Medição.

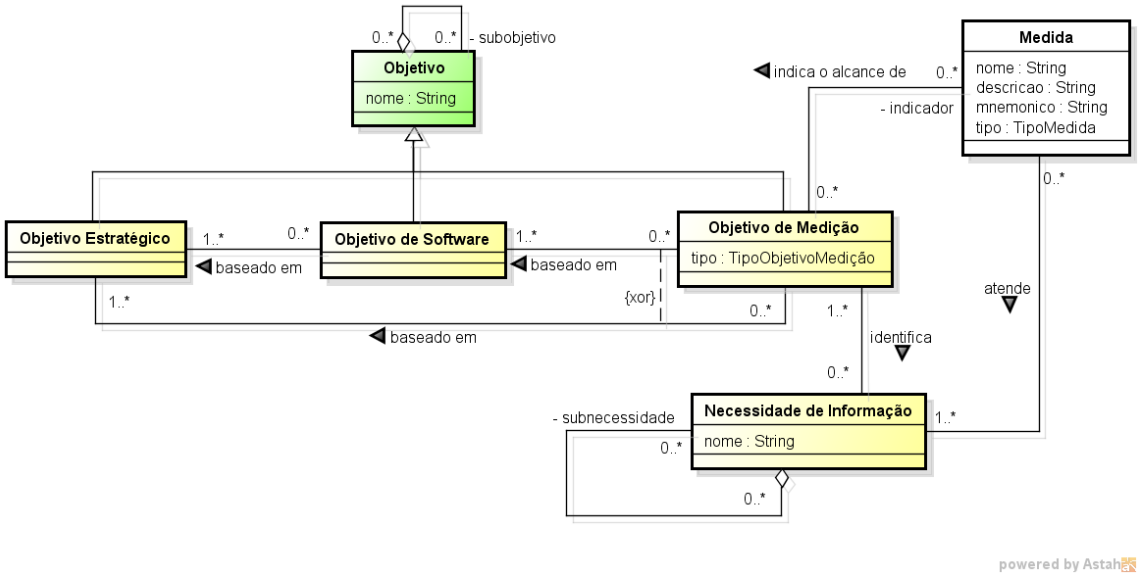

Figura 3.8 - Modelo de Classes do Pacote Objetivos de Medição.

Um **Objetivo** descreve a intenção pela qual ações são planejadas e realizadas. No contexto da medição de software, um objetivo pode ser um **Objetivo Estratégico**, um **Objetivo de Software** ou um **Objetivo de Medição**. Objetivos estratégicos representam os objetivos de negócio de uma organização, como, por exemplo, *aumentar o número de clientes em 10% nos próximos dois anos*. Objetivos de software são objetivos relacionados à área de software de uma organização, como, por exemplo, *alcançar o nível B do MPS.BR até o final do próximo ano*. Objetivos de software devem ser estabelecidos com base em objetivos estratégicos. Dessa forma, os objetivos que devem ser atendidos pela área de software estarão alinhados aos objetivos estratégicos da organização. Objetivos de medição são objetivos relacionados à área de medição de uma organização e devem ser estabelecidos com base em objetivos de software ou em objetivos estratégicos, a fim de garantir o alinhamento da medição com os objetivos da organização. Um exemplo de objetivo de medição, que poderia ser definido com base no objetivo de software apresentado anteriormente, é *monitorar o desempenho dos processos críticos*.

Objetivos de medição possuem um tipo (tipoObjetivoMedicao), que pode ser: objetivo de monitoramento e controle de projetos (por exemplo, *melhorar a aderência dos projetos aos planos*), objetivo de medição de qualidade (por exemplo, *reduzir o número de defeitos dos produtos entregues*) ou objetivo de medição de desempenho (por exemplo, *melhorar o desempenho dos processos críticos*). Ainda, objetivos podem ser subdivididos em outros objetivos.

As medidas atendem a **Necessidades de Informação**, que são identificadas a partir de objetivos. Medidas podem ser utilizadas para indicar o alcance a esses objetivos. Quando isso ocorre, a medida desempenha o papel de **indicador**. Por exemplo, a necessidade de informação *conhecer a estabilidade dos requisitos dos projetos após homologação junto ao cliente*, poderia ser identificada a partir do objetivo de medição *melhorar a aderência dos projetos aos planos* e atendida pela medida *taxa de alteração de requisitos*. Se essa medida é usada para analisar o alcance ao objetivo de medição, ela desempenha o papel de indicador. As Necessidades de Informação podem ser subdivididas. Por exemplo, *conhecer a estabilidade dos requisitos dos projetos após homologação junto ao cliente*, poderia ser decomposta em *conhecer o número de requisitos homologados para o projeto* e *conhecer o número de requisitos alterados ao longo do projeto.*

Uma diferença do modelo de classes do pacote Objetivos de Medição em relação à Ontologia de Referência para Medição de Software, é que, na ontologia, necessidades de informação podem ser identificadas a partir de objetivos e medidas podem ser utilizadas para indicar o alcance de objetivos. Essa é uma representação fidedigna ao mundo real. Mas, no contexto de soluções para medição de software, essas relações podem ser reduzidas para os objetivos de medição de software, uma vez que, seguindo a abordagem GQM (BASILI et al., 1994), a partir de objetivos de medição são identificadas questões (necessidades de informação) cujas respostas são obtidas a partir de medidas.

3.5.5. Pacote Definição Operacional de Medida

Trata do detalhamento operacional das medidas, provendo informações para guiar a coleta e análise. A Figura 3.9 apresenta o modelo de classes do pacote Definição Operacional de Medida.