UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

HENRIQUE NÉSPOLI CASTRO

EVOLUÇÃO DE UMA FERRAMENTA PARA APOIAR O PROCESSO DE MEDIÇÃO SOFTWARE

VITÓRIA
2016

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

HENRIQUE NÉSPOLI CASTRO

EVOLUÇÃO DE UMA FERRAMENTA PARA APOIAR O PROCESSO DE MEDIÇÃO SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.
Orientadora: Monalessa Perini Barcellos

VITÓRIA
2016

HENRIQUE NÉSPOLI CASTRO

EVOLUÇÃO DE UMA FERRAMENTA PARA APOIAR O PROCESSO DE MEDIÇÃO SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovado em_____de_____de 2016.

COMISSÃO EXAMINADORA

_____

Profª. Drª.Monalessa Perini Barcellos, D. Sc.
Universidade Federal do Espírito Santo
Orientadora

_____

Vinícius Soares Fonseca, M. Sc.
Universidade Federal do Espírito Santo

_____

Prof. Dr.Vítor Estêvão Silva Souza, D. Sc.
Universidade Federal do Espírito Santo

A minha irmã Juliana Maria

AGRADECIMENTOS

A Deus, pela força e determinação que me foi concebida nos momentos importantes na realização dessa conquista e pela fé acima de tudo para alcançar todos os objetivos desse trabalho.

Aos meus pais, Paulo Francisco e Elizabete Vitória, que são a minha base e exemplo de vida e que sempre estão ao meu lado, se sacrificando para essa seja possível realizar esse grande desafio de vida, e em especial à minha irmã, Juliana Maria, por todo o suporte e ajuda essencial nessa reta final, principalmente nos momentos de crise, em que minha saúde estava debilitada, para conclusão deste trabalho.

À minha orientadora Monalessa, pela prestatividade e boa vontade sempre em responder as minhas dúvidas e pelo vasto conhecimento transmitido.

Ao meu grande amigo Vinícius, que esteve comigo do início ao fim deste projeto, sempre disposto e paciente para trabalhar ao máximo no melhor.

Aos meus amigos, pela presença, incentivo, compreensão e pelos momentos de distração, quando precisei.

Enfim, agradeço a todos que de alguma forma contribuíram para a realização deste trabalho.

RESUMO

Atualmente, as grandes exigências feitas pelo mercado em relação aos produtos de software têm levado as organizações a buscarem a melhoria de seus processos de software. A medição de software é considerada uma das atividades mais importantes para a gerência e melhoria de processos e produtos de software, uma vez que fornece subsídios para a elaboração de planos realistas para os projetos e possibilita o monitoramento da aderência da execução dos projetos em relação a seus planos (ISO/IEC, 2007). Uma vez que a medição de software é implantada e executada corretamente em uma organização, torna-se possível passar à melhoria de processos. Nesse contexto, em (MARETTO, 2013), a partir de uma arquitetura de referência para medição de software foi desenvolvida a ferramenta Med&CEP (Medição & Controle Estatístico de Processos) visando apoiar o processo de medição de software. Med&CEP foi desenvolvida como prova de conceito para a arquitetura proposta, a qual é baseada na primeira versão da Ontologia de Referência para Medição de Software - ORMS (BARCELLOS, 2009). Devido ao fato de Med&CEP ter sido desenvolvida como prova de conceito, ela apresenta algumas limitações. Visando evoluir a ferramenta para tratrar essas limitações e propiciar seu uso em um ambiente organizacional real, este trabalho evoluiu Med&CEP e chamou a nova versão de SoMeSPC (*Software Measrurement and Statistical Process Control).* SoMeSPC considera a versão mais recente da ORMS e apresenta uma arquitetura de software mais robusta que sua versão anterior. Após a implementação de SoMeSPC, no contexto do trabalho de mestrado de Fonseca (2015), ela foi integrada a duas outras ferramentas e implantada em um organização para apoiar o processo de medição de software.

Palavras-chave: Medição de Software, Integração.

SUMÁRIO

Capítulo 1

Introdução

Este capítulo apresenta uma breve introdução ao tema do trabalho, seus objetivos, histórico do desenvolvimento e a organização deste documento.

1.1 INTRODUÇÃO

Em organizações de software, medição auxilia na identificação de boas práticas, melhoria de processos, análise de tendências, melhoria de estimativas e contribui para o conhecimento sobre a organização, que vai de gerentes a desenvolvedores.

Existem vários *frameworks* de apoio à definição de programas de melhoria de processo de software, nos quais a medição ocupa papel fundamental. Como exemplos, podem ser citados o MR MPS SW – Modelo de Referência para Melhoria de Processo de Software Brasileiro (SOFTEX, 2011), o CMMI – *Capability Maturity Model Integration* (SEI,2010), a ISO/IEC 15504 – *Information Technology – Process Assessment* (ISO/IEC, 2013) e a ISO/IEC12207 – *Systems and Software Engineering – Software Life Cycle Process* (ISO/IEC, 2008).

Medição de software consiste na avaliação quantitativa de qualquer aspecto dos processos e produtos 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). Ela contribui para a melhoria de processos e, consequentemente, para o desenvolvimento de produtos de melhor qualidade. É considerada uma das atividades mais importantes para a gerência de processos e produtos de software, uma vez que fornece subsídios para a elaboração de planos realistas para os projetos e possibilita o monitoramento da aderência da execução dos projetos em relação a seus planos (ISO/IEC, 2007).

Isso é possível pois as medidas, ao serem coletadas e armazenadas, podem ser analisadas através de métodos e fornecem informações importantes para a tomada de decisão, envolvendo a identificação e realização de ações corretivas e preventivas que orientem os projetos e processos a alcançarem os objetivos para eles estabelecidos (BARCELLOS, 2009).

Dentre os grandes 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, 2007 *apud* MARETTO, 2013).

Buscando auxiliar organizações no desenvolvimento de ferramentas para apoiar medição de software, Maretto (2013) propôs uma arquitetura de referência para medição de software, independente de tecnologia, que pode ser utilizada como base para o desenvolvimento de sistemas para apoiar medição de software. A partir da arquitetura proposta, foi desenvolvida uma ferramenta chamada Med&CEP, que provê apoio à medição de software e controle estatístico de processos.

Integrar ferramentas para apoiar o processo de medição de software pode ser a solução adequada para várias organizações, permitindo a integração das diferentes fontes de dados e serviços, a fim de se obter informações relevantes à tomada de decisão. Entretanto, a integração de sistemas como um todo é uma tarefa complexa, que envolve diferentes tipos de preocupações. No âmbito da medição de software, uma das preocupações é garantir que as ferramentas sejam integradas de forma a prover informações realmente relevantes, ou seja, informações alinhadas aos objetivos organizacionais e dos projetos (FONSECA, 2015).

Em (FONSECA, 2015) foi realizada uma iniciativa de integração de ferramentas para apoiar o processo de medição no LEDS (Laboratório de Extensão em Desenvolvimento de Sistemas) do IFES (Instituto Federal do Espírito Santo). A iniciativa foi conduzida utilizando uma abordagem para integração semântica de ferramentas, também proposta em (FONSECA, 2015), que usa a Ontologia de Referência para Medição de Software (ORMS) (Barcellos *et al.*, 2013) como uma de suas bases. Assim, considerando que Med&CEP foi desenvolvida a partir de uma arquitetura baseada em ORMS, ela foi selecionada como uma das ferramentas a serem integradas. Contudo, Med&CEP havia sido desenvolvida como uma prova de conceito, não estando preparada para implantação e utilização em um ambiente de produção devido a problemas de usabilidade e de implementação. Além disso, a integração de ferramentas para o LEDS deveria ser feita por meio de serviços e Med&CEP não estava preparada para tal. Por fim, uma vez que Med&CEP foi desenvolvida com base na primeira versão de ORMS (BARCELLOS, 2009), evoluções realizadas nessa ontologia não eram contempladas por Med&CEP.

1.2 OBJETIVOS

O *objetivo geral* deste trabalho é **evoluir a ferramenta Med&CEP para ser utilizada em uma iniciativa de integração para apoiar o processo de medição de software no contexto de (FONSECA, 2015).** Esse objetivo geral pode ser detalhado nos seguintes *objetivos específicos*:

i. Alterar tecnologias utilizadas a fim de se obter melhorias de desempenho e usabilidade;
ii. Adequar a conceituação à nova versão da Ontologia de Referência para Medição de Software;
iii. Desenvolver um mediador na ferramenta para permitir integração com as ferramentas SonarQube[1] e Taiga[2];
iv. Desenvolver uma API (*Application Programming Interface*) de serviços para expor funcionalidades de medição necessárias para a integração;

1.3 HISTÓRICO DE DESENVOLVIMENTO DO TRABALHO

A metodologia utilizada no desenvolvimento deste trabalho foi composta pelas seguintes atividades:

(i) *Revisão Bibliográfica*: O trabalho teve início com uma revisão bibliográfica sobre alguns assuntos, principalmente sobre medição de software. Foram analisados artigos científicos e trabalhos acadêmicos, sendo o principal deles (FONSECA, 2015), dissertação de mestrado no contexto da qual este trabalho foi conduzido.

(ii) *Estudo de Tecnologias*: Nesta etapa, foi necessário o estudo de tecnologias utilizadas para o desenvolvimento da ferramenta, tais como: Linguagem de Programação JAVA; *Framework* OpenXava; Ambiente de Desenvolvimento Eclipse; Servidor *Web Apache Tomcat*; Banco de Dados *MySQL*; *JSP (JavaServer Pages); AngularJS;* Twiiter *Bootstrap;* e *Quartz Scheduler*.

(iii) *Elaboração da Documentação da Ferramenta:* Nesta etapa, foi definida a documentação da ferramenta. A princípio foi apresentada uma descrição geral do minimundo do sistema e seu principal propósito. Em seguida, foram elaborados a composição do sistema, baseada em pacotes, os casos de uso e os diagramas de classes.

[1] *http://www.sonarqube.org/*

[2] *https://taiga.io/*

(iv) *Implementação e Testes da Ferramenta:* Nesta etapa, a ferramenta foi implementada e testada, tendo sido realizados testes ao longo do desenvolvimento, bem como após a conclusão de funcionalidades presentes na ferramenta.

(v) *Redação da Monografia:* Nesta etapa, foi realizada a escrita desta monografia.

1.4 ORGANIZAÇÃO DO TEXTO

Neste capítulo, foi apresentada uma introdução, contendo uma breve descrição do domínio do problema, o contexto em que ele está inserido e a motivação que levou a esse trabalho. Além deste capítulo, a monografia possui mais 4 capítulos, a saber:

Capítulo 2 – Medição de Software e Tecnologias para Aplicações Web: Apresenta uma breve fundamentação teórica sobre medição de software e o conjunto de tecnologias usados no contexto deste trabalho.

Capítulo 3 – Especificação e Análise de Requisitos: Apresenta a descrição do minimundo contemplado pela ferramenta, seus casos de uso e diagramas de classes.

Capítulo 4 – Implementação da Ferramenta SoMeSPC: Apresenta a arquitetura definida para a ferramenta, detalha alguns dos componentes dessa arquitetura e apresenta algumas telas da ferramenta.

Capítulo 5 – Considerações Finais: Apresenta as considerações finais do trabalho, incluindo algumas dificuldades encontradas, contribuições e experiências adquiridas no desenvolvimento dessa ferramenta. Além disso, são identificados alguns possíveis trabalhos futuros que poderão surgir a partir deste.

Capítulo 2

Medição de Software e Tecnologias para Aplicações Web

Este capítulo apresenta os principais aspectos teóricos que fundamentaram este trabalho e está organizado em 3 seções. A seção 2.1 aborda medição de software, destacando os principais conceitos e o processo de medição. A seção 2.2 apresenta brevemente a Ontologia de Referência para Medição de Software (ORMS). A seção 2.3 trata das tecnologias usadas na evolução da ferramenta.

2.1 MEDIÇÃO DE SOFTWARE

Segundo Dumke (2007), 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 a 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; DEMIRORS, 2006).

Dumke e Ebert (2007) 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).

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. Os resultados do planejamento da medição são registrados no Plano de Medição. 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, e analisá-los. A análise dos dados fornece informações para a tomada de decisão, favorecendo a identificação e execução de ações apropriadas. Por fim, o processo de medição e seus resultados devem ser avaliados buscando-se identificar oportunidades de melhoria (BARCELLOS *et al.*, 2010).

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).

2.2 ONTOLOGIA DE REFERÊNCIA PARA MEDIÇÃO DE SOFTWARE (ORMS)

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 (GUARINO, 1998).

A Ontologia de Referência para Medição de Software (ORMS) foi desenvolvida em (BARCELLOS, 2009), tendo sido evoluída desde então. ORMS é uma ontologia de domínio (i.e., estabelece uma conceituação para o domínio de medição) e é uma ontologia de referência, ou seja, uma ontologia construída com o objetivo de fazer a melhor descrição possível de um domínio na realidade, considerando um certo nível de granularidade e certo ponto de vista (GUIZZARDI, 2005). Buscando fidedignidade à realidade, ORMS foi desenvolvida à luz da Ontologia de Fundamentação Unificada (*Unified Foundational Ontology – UFO*) (GUIZZARDI, 2005).

ORMS é subdividida em sete subontologias e reutiliza conceitos das ontologias de Organização de Software (FALBO *et al.*, 2014) e Processos de Software (BRINGUENTE; FALBO; GUIZZARDI, 2011). A Figura 2.3 apresenta um diagrama de pacotes representando as subontologias de ORMS relevantes para este trabalho, as ontologias integradas e as relações entre elas. Em seguida, é feita uma breve descrição das subontologias. Considerando-se que a documentação de ORMS e das ontologias a ela integradas foi elaborada em inglês, optou-se por manter esse idioma nos diagramas aqui apresentados.

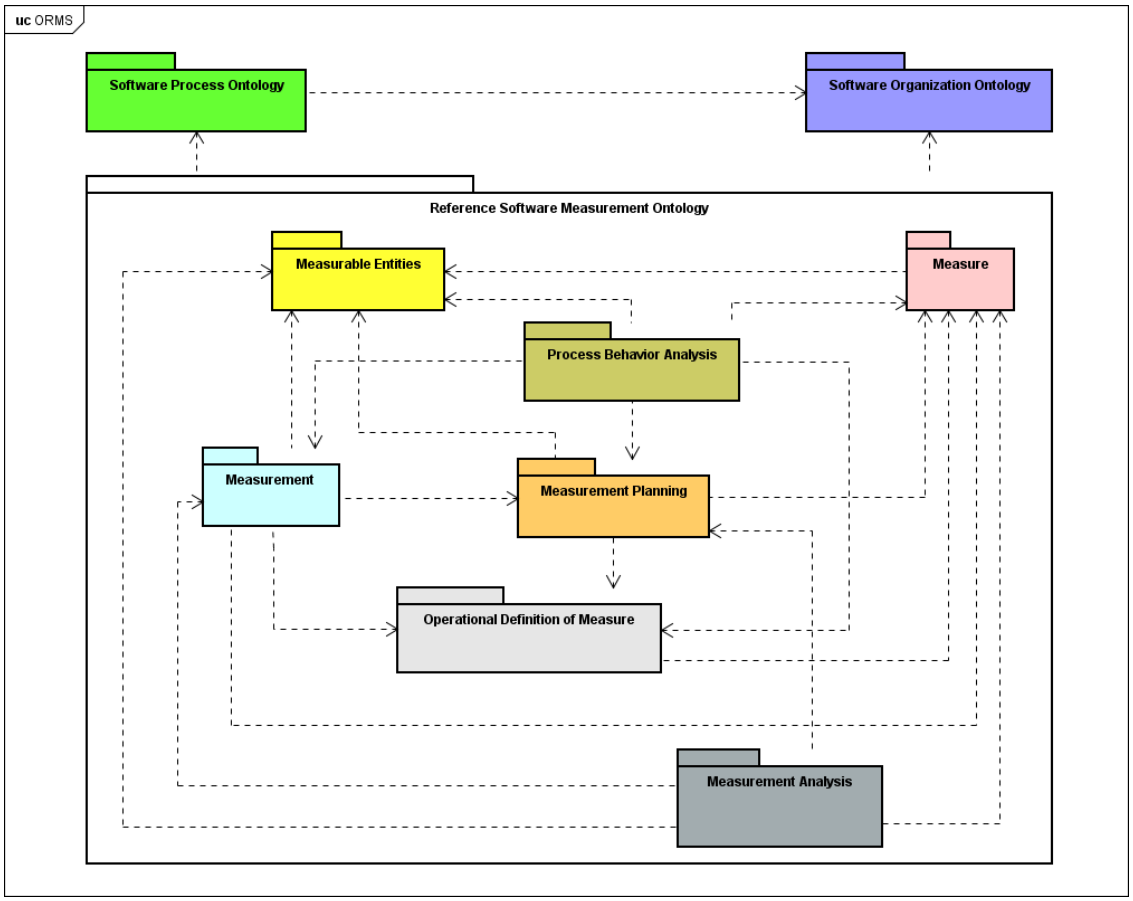

Figura 2.1 - Visão geral da Ontologia de Referência para Medição de Software. Fonte: (BARCELLOS, 2009; BARCELLOS *et al.*, 2013)

- Subontologia Entidades Mensuráveis (*Measurable Entities*): trata das entidades que podem ser submetidas à medição e das suas propriedades que podem ser medidas.
- Subontologia Medida (*Measure*): trata da definição de medidas de software e do relacionamento das medidas com as entidades e elementos mensuráveis.
- Subontologia Definição Operacional de Medidas (*Operational Definition of Measure*): trata do detalhamento de aspectos relacionados à coleta e análise de medidas.
- Subontologia Planejamento de Medição (*Measurement Planning*): trata de aspectos relacionados ao planejamento da medição de software.
- Subontologia Medição (*Measurement*): trata da medição propriamente dita, ou seja, a coleta e armazenamento dos dados para as medidas.

- Subontologia Análise de Medição (*Measurement Analysis*): trata da análise dos dados coletados para as medidas para obtenção das informações de apoio às decisões.
- Subontologia Análise de Comportamento de Processos (*Process Behavior Analysis*): trata da aplicação da análise dos dados coletados na análise de comportamento de processos utilizando controle estatístico de processos.

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).

2.3 TECNOLOGIAS PARA DESENVOLVIMENTO DE APLICAÇÕES WEB

Os usuários de sistemas ou sites na web estão mais exigentes quanto à rapidez, eficácia e qualidade. Felizmente, podem-se utilizar *frameworks* que apoiam o desenvolvimento de sistemas (ou aplicações) web. Neste trabalho foram utilizadas diversas tecnologias, sendo que grande parte delas envolve *frameworks*. A seguir são apresentadas algumas das tecnologias usadas neste trabalho.

- **OpenXava**[3]: é um *framework* Java com AJAX (*Asynchronous Javascript* e XML) voltado para desenvolvimento ágil de aplicações web, com módulos CRUD e geração de relatórios. A essência desse *framework* é que ele permite o desenvolvedor definir em vez de programar. Ele fornece automaticamente a interface com o usuário, o acesso aos dados e muitas outras caracteristicas. Desta forma, todos os problemas comuns são resolvidos com facilidade e o desenvolvedor tem sempre a possibilidade de programar manualmente qualquer parte da aplicação, sendo flexível suficiente para resolver os casos particulares.
- **Apache Tomcat**[4]: é um servidor web Java, mais especificamente, um *container* de *servlets*. Ele cobre parte da especificação Java EE com tecnologias como Servlet e JSP (JavaServerPage). O Tomcat tem a capacidade de atuar como servidor web/HTTP, puramente Java, ou pode funcionar integrado a um servidor web dedicado como o Apache ou o Microsoft IIS. O servidor inclui ferramentas para configuração e gerenciamento, o que também pode ser feito

[3] http://www.openxava.org/

[4] http://tomcat.apache.org/

editando-se manualmente arquivos de configuração formatados em XML. Em resumo o Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção.

- **Google AngularJS**[5]: é um *framework* JavaScript mantido pelo Google e que vem ganhando cada vez mais adeptos para o desenvolvimento de aplicações WEB. Foi construído sob o padrão *Model View Controller* (MVC) e possui suporte a data *binding*, com ligação direta e bidirecional dos dados (two-way data *binding*), que permite sincronização automática de *models* e *views*, injeção de dependência, carga de módulos com gerenciamento de dependências, roteamento e gestão de histórico, serviços e promessas. Ele se adapta e estende o HTML (*Hyper Text Markup Language*), que consiste em uma linguagem de marcação utilizada para produção de páginas na web, que permite a criação de documentos que podem ser lidos em praticamente qualquer tipo de computador e transmitidos pela internet. Ele é simples e intuitivo, guia os desenvolvedores através da construção de todo o aplicativo de forma robusta e organizada, escrevendo código muito mais expressivo e de fácil manutenção, desde o design de interface, passando pela escrita das regras de negócio, até chegar aos testes da aplicação.
- **Twitter Bootstrap**[6]: é um *framework* ideal para melhorar a usabilidade do sistema, contendo um conjunto de HTML , CSS (*Cascading Style Sheets*) e JavaScript que proporciona uma maneira ultra-rápida, eficiente e profissional de planejar e desenvolver o front-end de web sites, aplicações, sistemas e produtos web através de seus elementos de UI (*User Interface*) e interações. Ele possui uma diversidade de componentes (plugins) escritos em JavaScript e que fazem uso da biblioteca JQuery, auxiliando o designer a implementar: *tootlip*, menu-*dropdown*, *modal*, *carousel*, *slideshow*, entre outros recursos.
- **Quartz Scheduler**[7]: O Quartz é um serviço de agendamento de tarefas que pode ser integrado, ou utilizado virtualmente, em qualquer aplicação Java SE ou Java EE. A ferramenta pode ser utilizada para criar agendas que executam milhares de tarefas, que são definidas utilizando componentes padrão da plataforma Java, que são codificados para suprir as necessidades da aplicação. O Quartz Scheduler fornece diversos recursos corporativos,

[5] *https://angularjs.org/*

[6] *http://getbootstrap.com/*

[7] *https://quartz-scheduler.org/*

como suporte a transações JTA ou clusterização. O Scheduler é o componente principal do Quartz e é o responsável por gerenciar a execução de *jobs*. A partir dele, o desenvolvedor pode agendar, iniciar e parar as execuções.

- **Taiga**: é uma plataforma de gerenciamento de projetos para desenvolvedores ágeis e designers. Com foco no desenvolvimento ágil de projetos baseados no Scrum, Taiga permite o controle das atividades do projeto e utiliza conceitos do Scrum como *sprint*, estória e tarefas, Kanban, entre outros.
- **SonarQube**: SonarQube é uma ferramenta de análise estática, específica para o controle de qualidade do código fonte. A partir de um conjunto de regras e padrões, o SonarQube coleta diversos dados sobre o código fonte de um software.

Capítulo 3

Especificação e Análise de Requisitos

Este capítulo aborda os resultados da Engenharia de Requisitos da ferramenta SoMeSPC. Na seção 3.1, são apresentados o objetivo da ferramenta e seus requisitos funcionais, na seção 3.2, é apresentado o sistema da SoMeSPC, organizado em pacotes, na seção 3.3, são apresentados diagramas de casos de uso, na seção 3.4, são apresentados os diagramas de classes.

3.1 PROPÓSITO DO SISTEMA

A ferramenta SoMeSPC tem como propósito apoiar as atividades do processo de medição de software, incluindo o planejamento da medição, a coleta e armazenamento de dados para as medidas definidas, bem como a análise dos dados coletados.

Para atender a esse propósito a ferramenta deve atender aos requisitos funcionais gerais definidos em (MARETTO, 2013), conforme mostra a Tabela 3.1.

Tabela 3.1 – Requisitos Funcionais de SoMeSPC.

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

Tabela 3.1 – Requisitos Funcionais de SoMeSPC (cont).

Id	Requisito
R20	Deve ser possível apresentar resultados da medição por meio de relatórios, gráficos ou consultas.
R21	Deve ser possível apresentar resultados da análise de comportamento de processos por meio de relatórios, gráficos ou consultas.

Conforme mencionado do Capítulo 1 desta monografia, a principal motivação para evoluir Med&CEP e transformá-la em SoMeSPC foi a necessidade de utilizá-la em uma iniciativa de integração para apoiar o processo de medição de software no contexto de (FONSECA, 2015). Para isso, foram identificados outros requisitos, focando nas funcionalidades necessárias para mediar a integração de SoMeSPC com SonarQube e Taiga, conforme definido em (FONSECA, 2015).

Tabela 3.2 – Requisitos Funcionais para Medição de Integração.

Id	Requisito
R1	Deve ser possível gerar planos de medição considerando medidas presentes das ferramentas SonarQube e Taiga.
R2	Deve ser possível registrar o agendamento de medições automáticas.
R3	Deve ser possível realizar medições automaticamente.
R4	Deve ser possível apresentar gráficos para apoiar a análise das medições realizadas por meio da integração.

A descrição completa dos requisitos funcionais gerais de SoMeSPC e dos requisitos funcionais para Medição de Integração podem ser encontradas em (MARETTO, 2013) e (FONSECA, 2015), respectivamente.

3.2 SUBSISTEMAS

A Figura 3.1 apresenta o diagrama de pacotes de SoMeSPC, onde são identificados os subsistemas e as relações entre eles. Uma vez que a ferramenta é baseada em ORMS, há similaridade entre o diagrama de pacotes de SoMeSPC e de ORMS. Informações sobre a definição dos subsistemas a partir de ORMS encontram-se em (MARETTO, 2013). Além dos subsistemas definidos a partir da ORMS, foram definidos outros dois. O subsistema *Caracterização de Projeto* lida com aspectos necessários para atribuir características a projetos (por exemplo, nível de experiência da equipe, tamanho, complexidade, tipo de cliente, domínio, etc.) a fim de que seja possível identificar projetos similares. Identificar projetos similares é muito importante no contexto de medição de software, uma vez que as análises dos dados coletados para as medidas devem considerar as

características dos projetos. O subsitema *Mediador,* trata dos aspectos necessários para mediar a integração da ferramenta com SonarQube e Taiga.

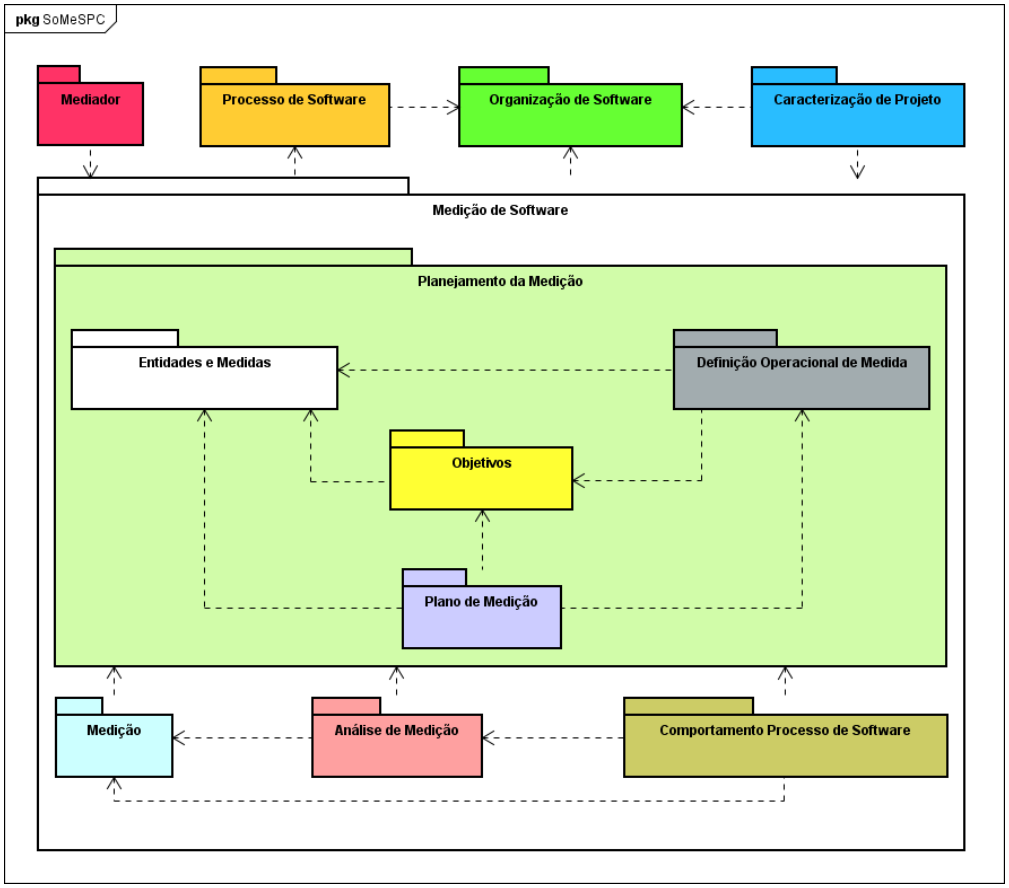

Figura 3.1 - Diagrama de Pacotes da SoMeSPC.

3.3 CASOS DE USO

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. Os atores identificados no contexto deste projeto são:

- **Gerente de Medição**: Papel responsável pelo cadastro de dados necessários para a realização do processo de medição (tais como dados sobre a organização, recursos humanos e processos) e pelo planejamento da medição.
- **Executor da Medição**: Papel responsável por realizar a medição.
- **Executor da Análise de Medição**: Papel responsável por realizar a análise da medição.

A seguir são apresentados os diagramas de casos de uso de SoMeSPC para cada subsistema. São usadas as mesmas cores do diagrama de pacotes e o *namespace* do diagrama para identificar o subsistema de origem de cada caso de uso. A maioria dos casos de uso dos pacotes Organização, Processo de Software, Entidades e Medidas, Definição Operacional de Medida e Objetivos são cadastrais e são responsáveis pelo registro dos dados necessários para a execução do processo de medição propriamente dito. Vale ressaltar que, embora em (MARETTO, 2013) tenham sido definidos os requisitos, o diagrama de pacotes e os modelos conceituais da versão anterior da ferramenta, os casos de uso não haviam sido documentados, além disso existem relacionamentos entre casos de uso de um subsistema para outro, também identificados no contexto deste trabalho. A Figura 3.2 apresenta o diagrama de casos de uso do subsistema Organização de Software.

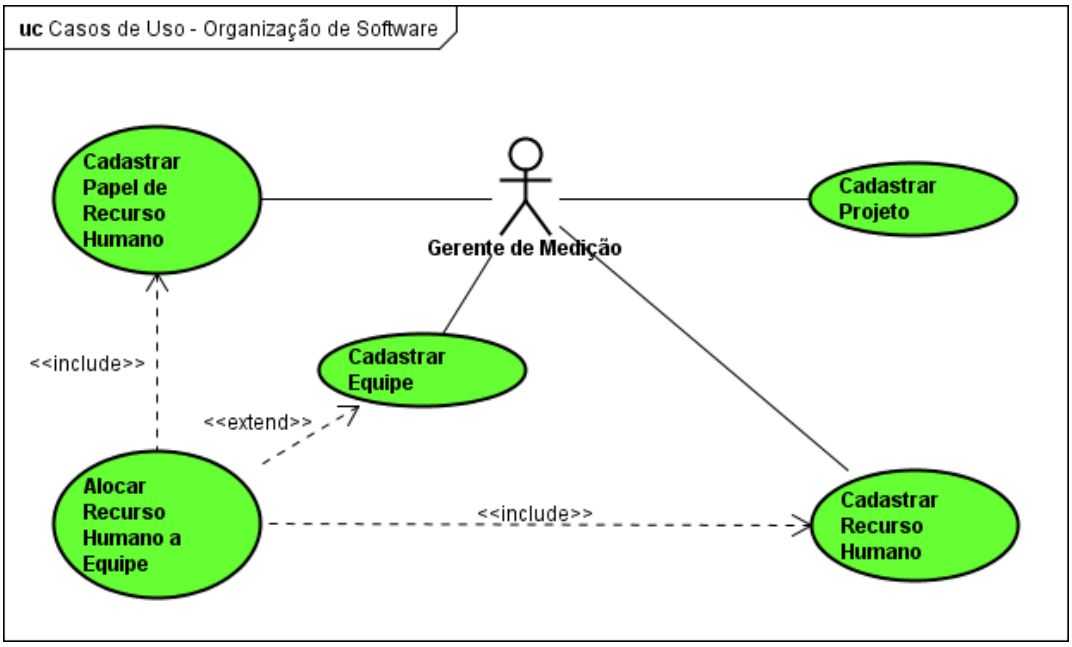

Figura 3.2 - Diagrama de Casos de Uso do Subsistema Organização de Software.

O caso de uso *Cadastrar Projeto* permite que o gerente de medição crie um projeto, formado por uma equipe (caso de uso *Cadastrar Equipe*), que contém recursos humanos (caso de uso *Cadastrar Recurso Humano*), cada um tendo sua alocação e seu papel definido (casos de uso *Alocar Recurso Humano a Equipe* e *Cadastrar Papel de Recurso Humano*, respectivamente).

A Figura 3.3 apresenta o diagrama de casos de uso do subsistema Caracterização de Projetos.

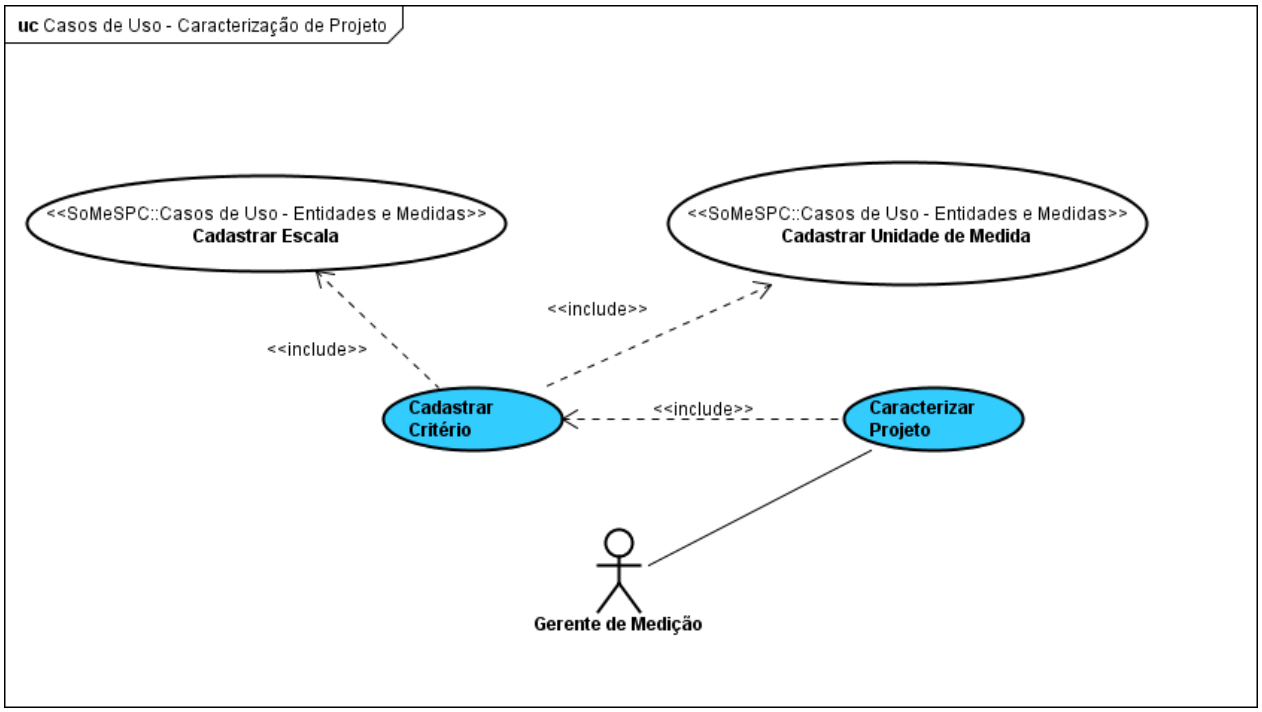

Figura 3.3 - Diagrama de Casos de Uso do Subsistema Caracterização de Projeto.

O caso de uso *Caracterizar Projeto* permite que o gerente de medição informe características do projeto, que no contexto de medição de software são úteis para identificar projetos similares. As características são determinadas a partir de critérios de caracterização como, por exemplo, nível de experiência da equipe, que poderia receber os valores muito alto, alto, médio, baixo, muito baixo, os quais formam a escala do critério (i.e., conjunto de valores que podem ser atribuídos a ele).

A Figura 3.4 apresenta o diagrama de casos de uso do subsistema Processo de Software.

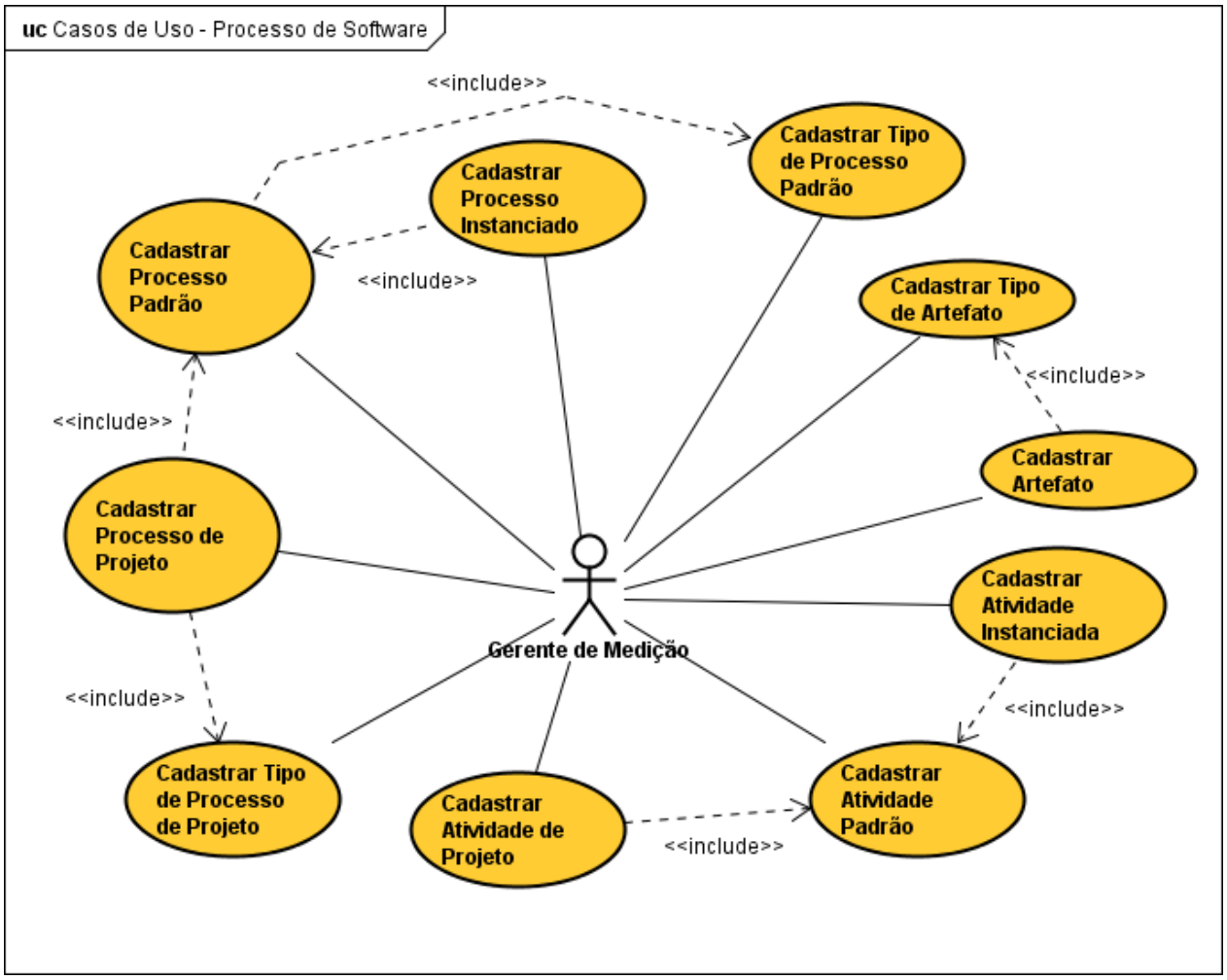

Figura 3.4 - Diagrama de Casos de Uso do Subsistema Processo de Software.

Os casos de uso *Cadastrar Processo Padrão* e *Cadastrar Processo de Projeto* permitem que o gerente de medição, através dessas funcionalidades, tenha conhecimento das atividades, métodos, ferramentas e práticas que são utilizadas para construir um produto de software. Na definição de um processo devem ser consideradas as seguintes informações: atividades a serem realizadas (casos de uso *Cadastrar Atividade Padrão* e *Cadastrar Atividade de Projeto*), recursos necessários, artefatos requeridos e produzidos (caso de uso *Cadastrar Artefato*), e procedimentos adotados.

A Figura 3.5 apresenta o diagrama de casos de uso do subsistema Entidades e Medidas.

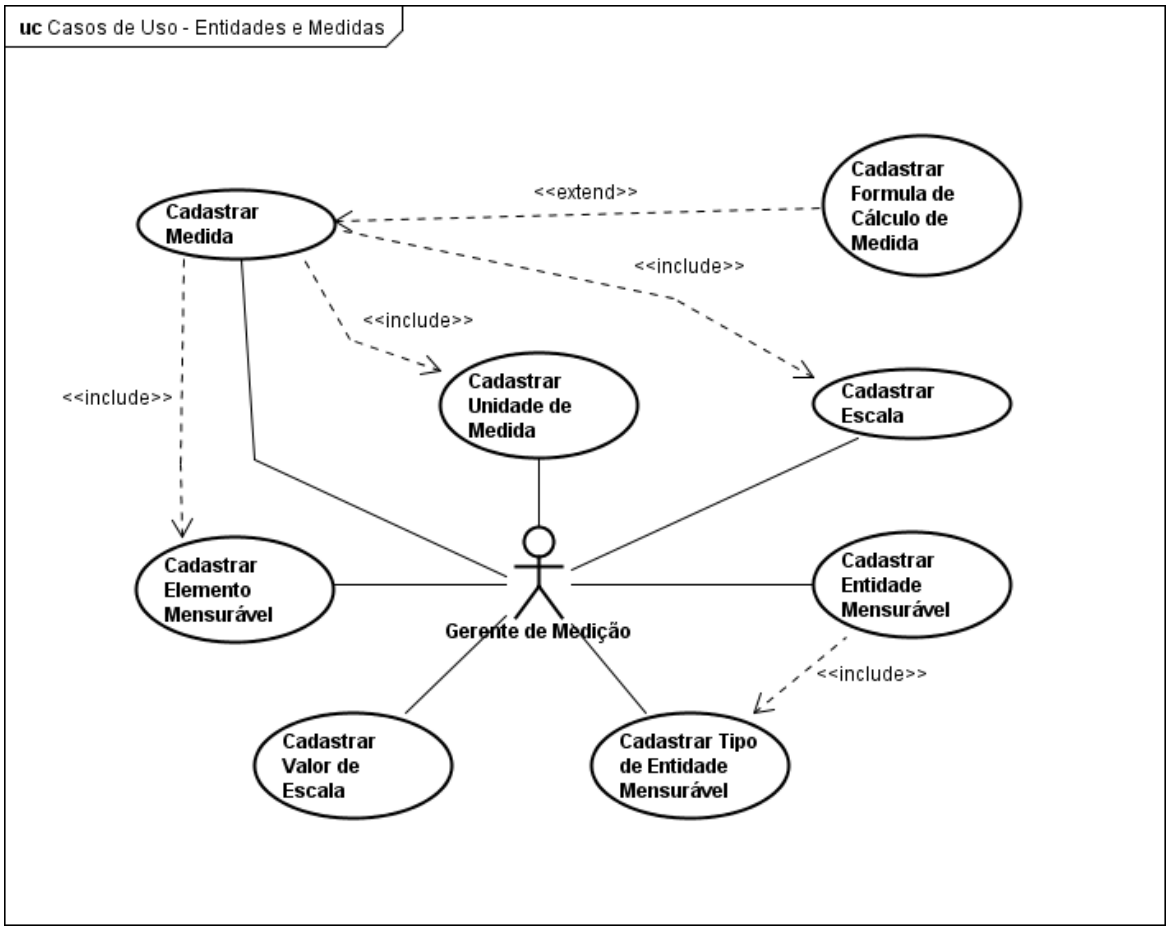

Figura 3.5 - Diagrama de Casos de Uso do Subsistema Entidades e Medidas.

O caso de uso *Cadastrar Medida* é usado pelo gerente de medição para controlar as medidas, responsáveis por quantificar (medir) uma propriedade (elemento mensurável) de um tipo de entidade mensurável. A medida deve ser: válida, para quantificar o que se quer; confiável, para produzir o mesmos resultados dadas pelas mesmas condições; e prática, fácil de se computar e interpretar. Uma medida pode ser medida base, quando depender de nenhuma outra medida, ou medida derivada, quando for obtida através de outras medidas base ou derivadas.

A Figura 3.6 apresenta o diagrama de casos de uso do subsistema Objetivos.

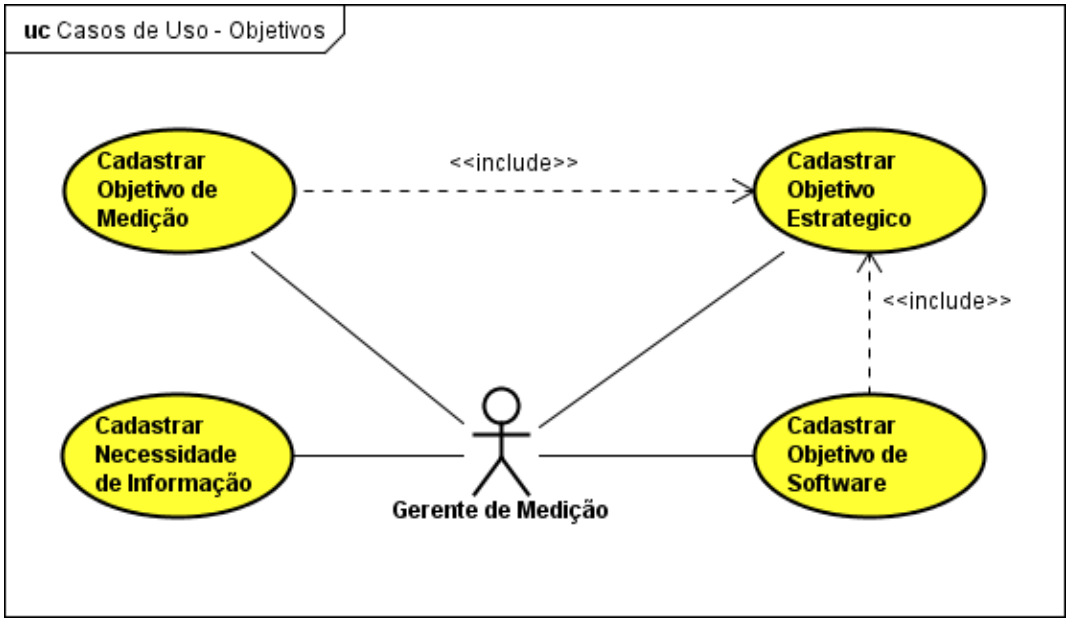

Figura 3.6 - Diagrama de Casos de Uso do Subsistema Objetivos.

O caso de uso *Cadastrar Objetivo de Medição* permite que o gerente de medição expresse as intenções pelas quais ações relacionadas à medição são planejadas e realizadas. Objetivos de medição são definidos com base em objetivos estratégicos, que expressam a intenção pela qual ações estratégicas são planejadas e realizadas, e a partir deles pode-se derivar objetivos de software, importantes para mensurar produtos ou processos de software.

A Figura 3.7 apresenta o diagrama de casos de uso do subsistema Definição Operacional de Medida.

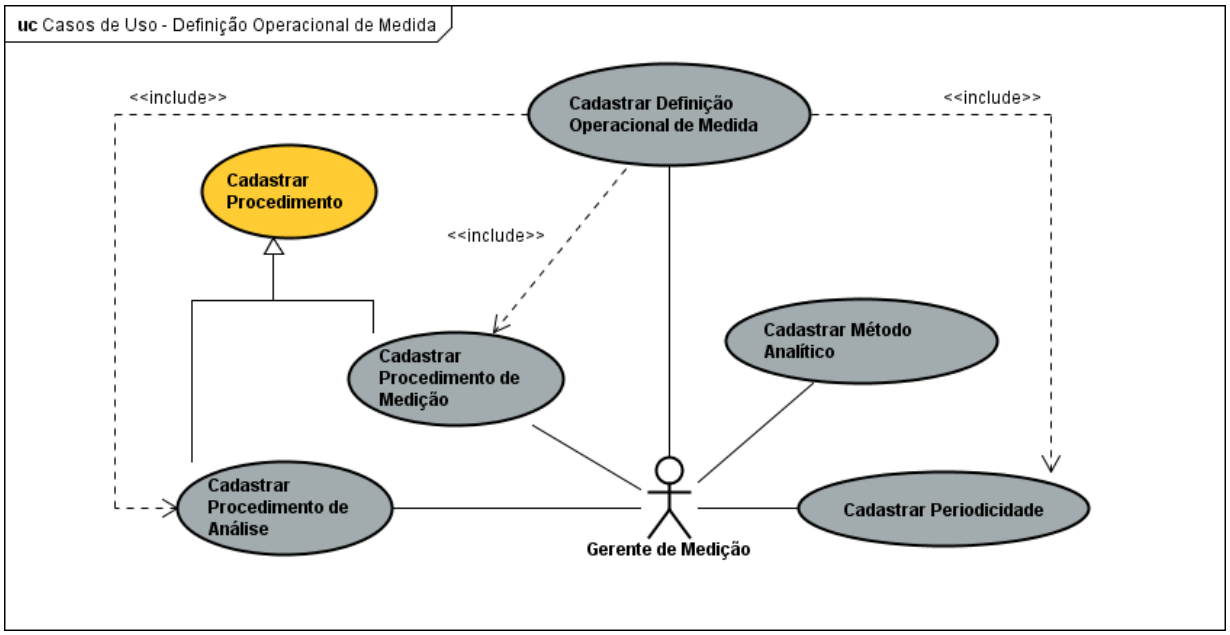

Figura 3.7 - Diagrama de Casos de Uso do Subsistema Definição Operacional de Medida.

O caso de uso *Cadastrar Definição Operacional de Medida* permite ao gerente de medição detalhar uma medida com informações sobre sua coleta e análise. Uma definição operacional de medida inclui procedimentos de medição e análise, papéis dos responsáveis pela medição e análise da medida, momento em que a medição e análise devem ser realizadas e a periodicidade.

A Figura 3.8 apresenta o diagrama de casos de uso do subsistema Plano de Medição. Os casos de uso desse subsistema são responsáveis pela elaboração do Plano do Projeto, que é o principal resultado do planejamento da medição. *Elaborar Plano de Medição da Organização* é responsável pela criação do plano de medição organizacional, que serve como base para a definição dos Planos de Medição de Projeto, tratados no caso de uso *Elaborar Plano de Medição do Projeto.*

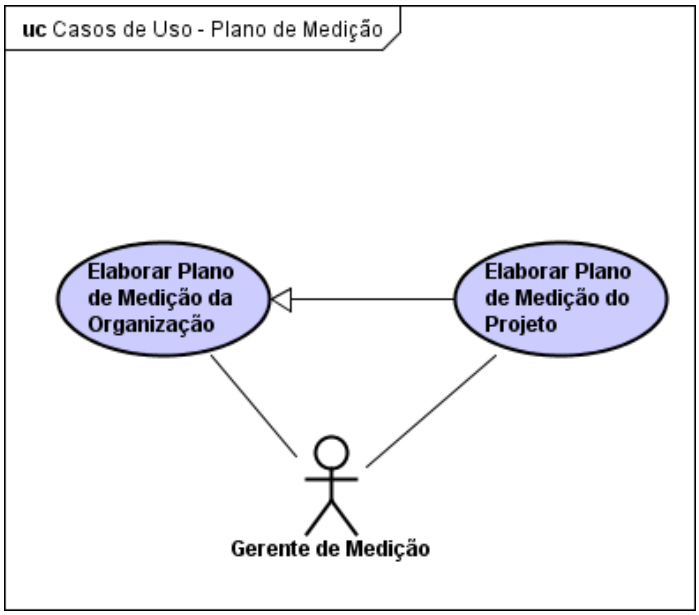

Figura 3.8 - Diagrama de Casos de Uso do Pacote Subsistema de Medição.

A Figura 3.9 (a) apresenta o diagrama de casos de uso do subsistema Medição. O caso de uso *Registrar Medição* é responsável pelo registro dos valores medidos nas medições. Na Figura 3.9 (b) é apresentado o diagrama de casos de uso do subsistema Análise de Medição. O caso de uso desse pacote é responsável pela seleção, representação e análise dos dados coletados para as medidas. A Figura 3.9 (c) apresenta o diagrama de casos de uso do subsistema Comportamento Processso de Software. *Registrar Desempenho Especificado para o Processo* trata do registro dos valores estabelecidos pela organização para o desempenho do processo, considerando uma medida específica. *Analisar Comportamento de Processo* trata da análise dos dados das medições utilizando técnicas do controle estatístico de processos. Por fim, *Registrar Modelo de Desempenho do Processo* permite que os resultados na análise de comportamento dos processos sejam utilizados para estabelecer modelos de desempenho para os processos.

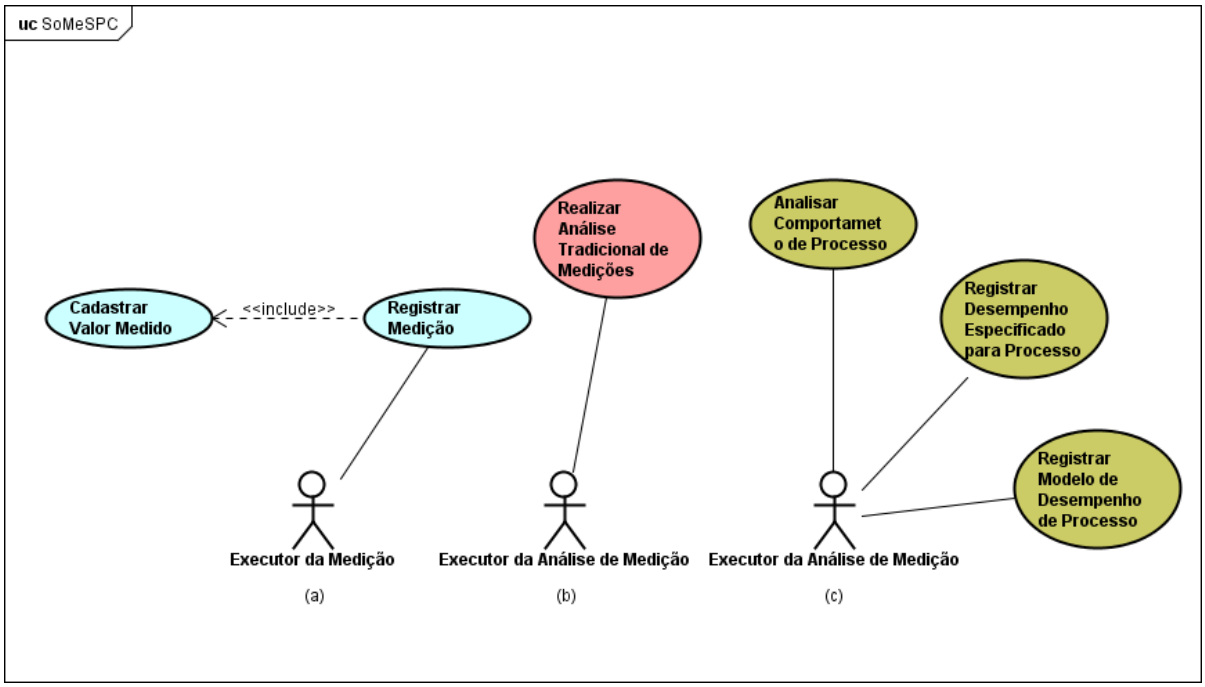

Figura 3.9 - Diagramas de Casos de Uso dos Subsistemas (a) Medição, (b) Análise de Medição e (c) Comportamento Processo de Software.

A Figura 3.10 apresenta o diagrama de casos de uso do subsistema Mediador.

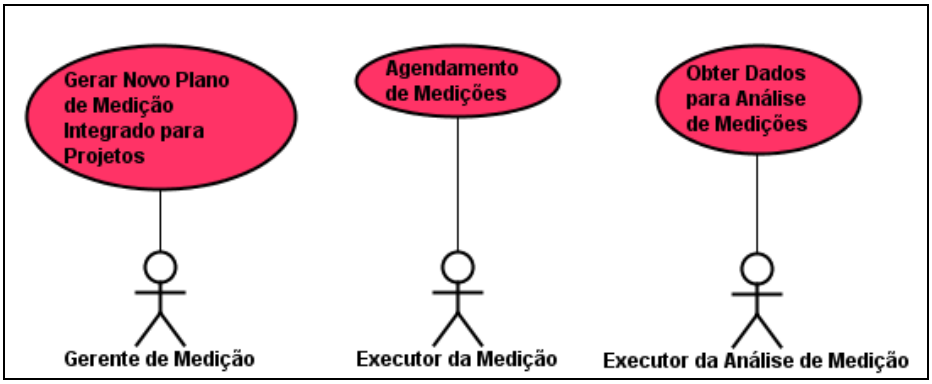

Figura 3.10 - Diagrama de Casos de Uso do Subsistema Mediador.

Gerar Novo Plano de Medição Integrado para Projetos dá suporte à criação de plano de medição para projetos considerando medidas disponíveis nas ferramentas integradas. *Agendamento de Medições* é responsável pelo agendamento de coleta de dados (i.e., medições) a ser realizada automaticamente a partir das ferramentas integradas. Por fim, *Obter Dados para Análise de Medições* permite seleção de *dados* obtidos a partir da integração e representação destes em gráficos para serem analisados.

3.4 DIAGRAMAS DE CLASSES

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades descritas na seção anterior. A seguir, são apresentados os diagramas de classes de SoMeSPC. Os diagramas de classes de alguns subsistemas não foram alterados, permanecendo os mesmos originalmente definidos em (MARETTO, 2013). Para os diagramas que foram alterados, as modificações realizadas são citadas. As alterações foram realizadas visando principalmente a adequação da nova versão da ferramenta à versão atual de ORMS. Decidiu-se por apresentar todos os diagramas (mesmo os não alterados) para que esta monografia contenha a documentação completa da nova versão da ferramenta. Demais informações sobre restrições de integridade e tipo dos atributos estão definidos em (MARETTO, 2013). Nos diagramas, são utilizadas cores diferentes, seguindo as cores apresentadas no diagrama de pacotes da Figura 3.1, para designar os subsistemas de origem de cada classe apresentada.

3.4.1 Diagrama de Classes do Subsistema Organização de Software

A Figura 3.11 apresenta o diagrama de classes do subsistema Organização de Software. Não foram feitas alterações nesse modelo de classes, tendo sido mantido o modelo definido em (MARETTO, 2013).

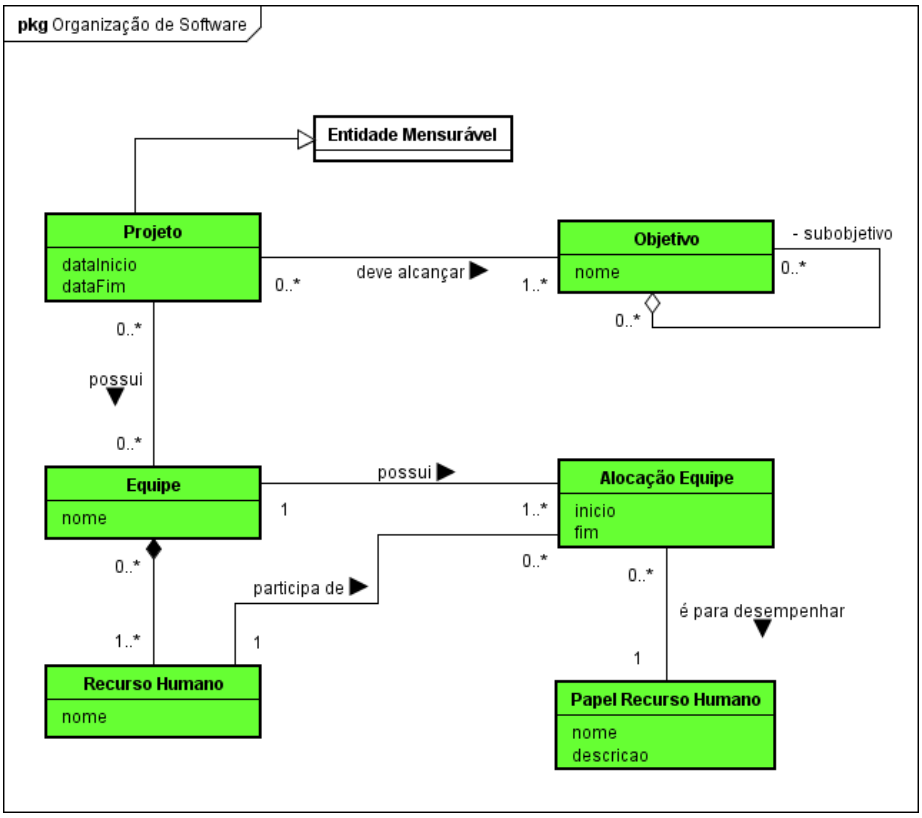

Figura 3.11 - Diagrama de Classes do Subsistema Organização de Software.

Um **Projeto** é criado para alcançar determinados **Objetivos**. Um Projeto possui **Equipes** formadas por **Recursos Humanos**. Um Recurso Humano pode estar alocado em um dado momento a uma Equipe (**Alocação Equipe**) desempenhando um determinado **Papel Recurso Humano** (p. ex., analista, programador). A classe Projeto possui, além dos atributos apresentados, os atributos nome e descrição, os quais são definidos em uma relação de generalização/especialização com a classe Entidade Mensurável.

3.4.2 Diagrama de Classes do Subsistema Processo de Software

A Figura 3.12 apresenta o diagrama de classes do subsistema Processo de Software.

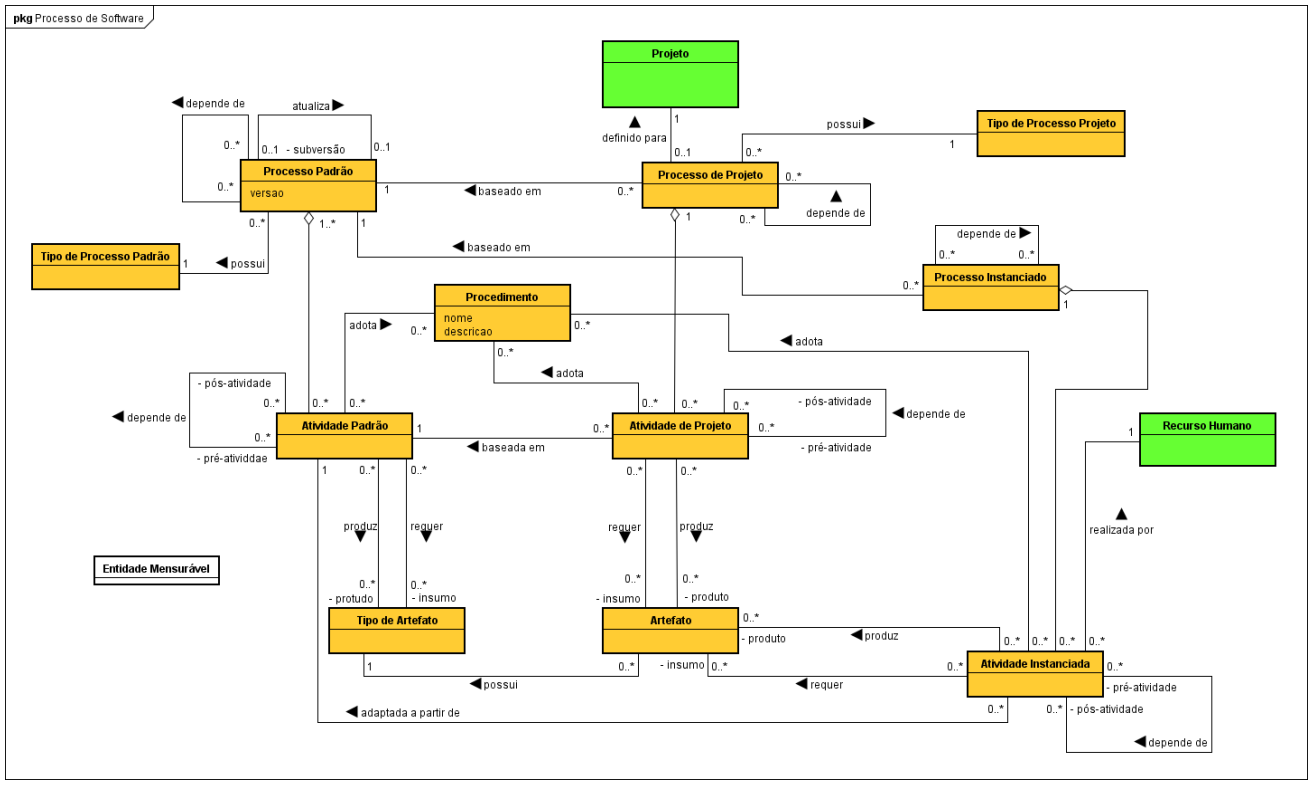

Figura 3.12 - Diagrama de Classes do Subsistema Processo de Software.

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. Um Processo Padrão possui um **Tipo de Processo Padrão**. Um Processo Padrão é formado por **Atividades Padrão**, que podem possuir uma relação de dependência com outras, ditas suas pré-atividades. Uma Atividade Padrão pode requerer (*insumo*) ou produzir (*produto*) **Tipos de Artefatos** e pode, ainda, adotar um **Procedimento**.

Com base nos Processos Padrão da organização, processos podem ser instanciados, a fim de serem executados. **Processos Instanciados** 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**, 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. Atividade Instanciada é realizada por um **Recurso Humano**.

Processo de Projeto é um processo definido para um projeto, composto por **Atividades de Projeto**, que são baseadas em Atividades Padrão que compõem o Processo Padrão utilizado como base. Um Processo de Projeto possui um **Tipo de Processo Projeto**. 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.

As classes destacadas em laranja na Figura 3.12, exceto **Procedimento**, **Tipo de Processo Padrão** e **Tipo de Processo Projeto**, possuem os atributos nome e descrição, os quais são definidos em uma relação de generalização/especialização com a classe Entidade Mensurável, mas devido a complexidade do diagrama, esse relacionamento não pode ser representado, pois a imagem ficaria poluída, tornando sua leitura e interpretação inviável.

3.4.3 Diagrama de Classes do Subsistema Entidades e Medidas

A Figura 3.13 apresenta o diagrama de classes do subsistema Entidades e Medidas. Com exceção das alterações oriundas do modelo de classes do pacote Processo de Software, não foram feitas alterações nesse modelo de classes, tendo sido mantido o modelo definido em (MARETTO, 2013).

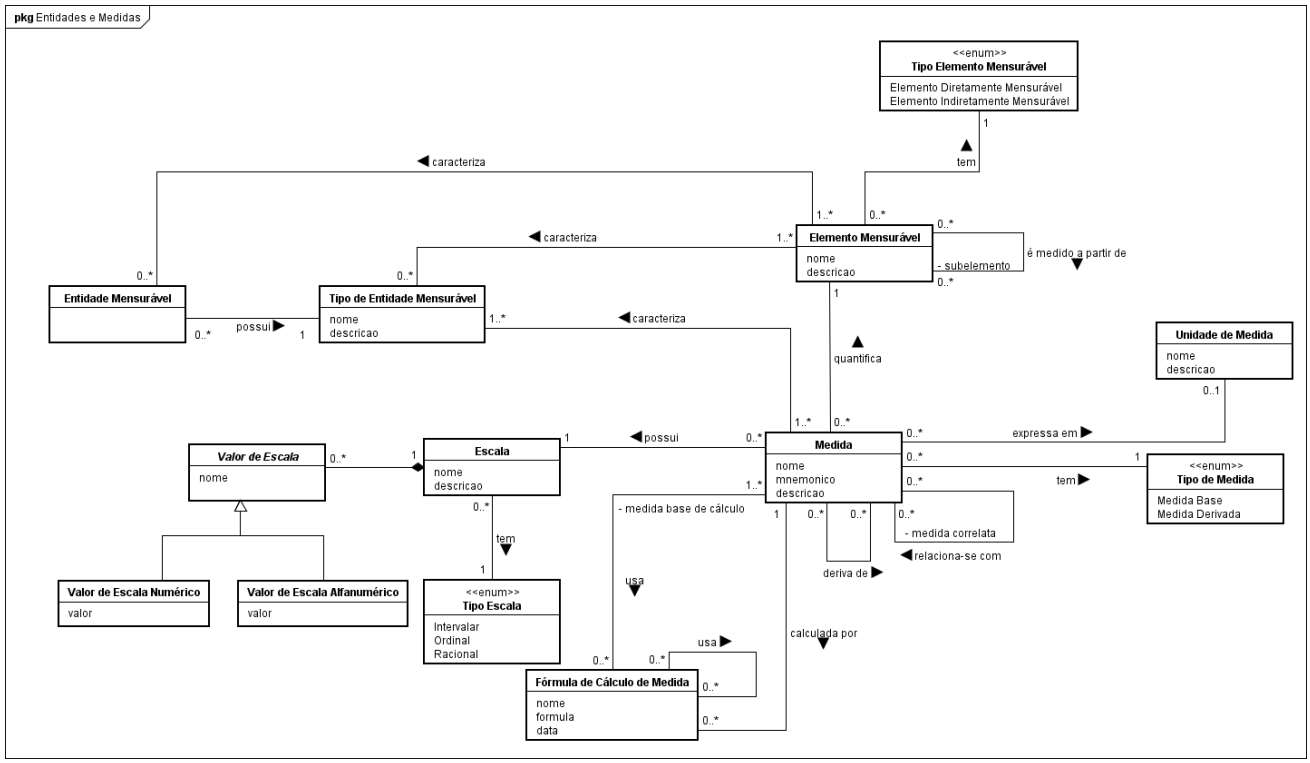

Figura 3.13 - Diagrama de Classes do Subsistema Entidades e Medidas.

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, por exemplo, **Projeto, Processo Padrão**, **Atividade Padrão**, **Atividade Instanciada**, **Processo de Projeto**, **Processo Instanciado**, **Atividade de Projeto**, **Tipo de Artefato** e **Artefato**. Tipos de entidades mensuráveis são caracterizados por **Elementos Mensuráveis**, que são propriedades que podem ser medidas. Entidades mensuráveis são caracterizadas por elementos mensuráveis que caracterizam entidades do seu tipo.

Elementos Mensuráveis tem **Tipo Elemento Mensurável** que pode ser direta ou indiretamente mensurável. Elementos indiretamente mensuráveis são medidos a partir da medição de outros elementos mensuráveis, ditos seus subelementos.

Medidas quantificam elementos mensuráveis, elas tem **Tipo de Medida** que pode ser: medida base, quando quantifica um elemento diretamente mensurável ou medida derivada, quando quantifica um elemento indiretamente mensurável, sendo funcionalmente dependente de outras medidas. 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.

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 tem um **Tipo Escala**, que pode ser, Intervalar, Ordinal e Racional. Escalas possuem **Valor de Escala**. Algumas escalas podem não ter seus valores de escala determinados. Medidas podem se relacionar com outras que, de alguma forma, influenciam em seu valor. Essas medidas são ditas **medidas correlatas**.

3.4.4 Diagrama de Classes do Subsistema Objetivos

A Figura 3.14 apresenta o diagrama de classes do subsistema Objetivos. Não foram feitas alterações nesse modelo de classes, tendo sido mantido o modelo definido em (MARETTO, 2013).

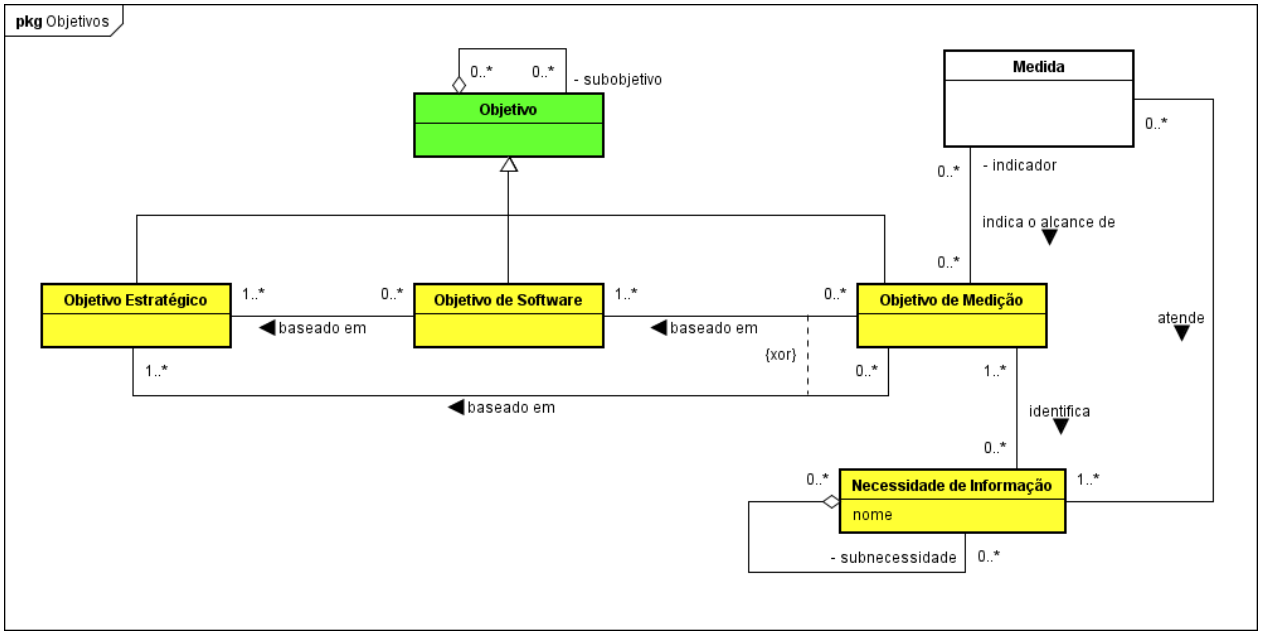

Figura 3.14 - Diagrama de Classes do Subsistema Objetivos.

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. Objetivos de software são objetivos relacionados à área de software de uma organização e 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.

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**.

3.4.5 Diagrama de Classes do Subsistema Definição Operacional de Medida

A Figura 3.15 apresenta o diagrama de classes do subsistema Definição Operacional de Medida.

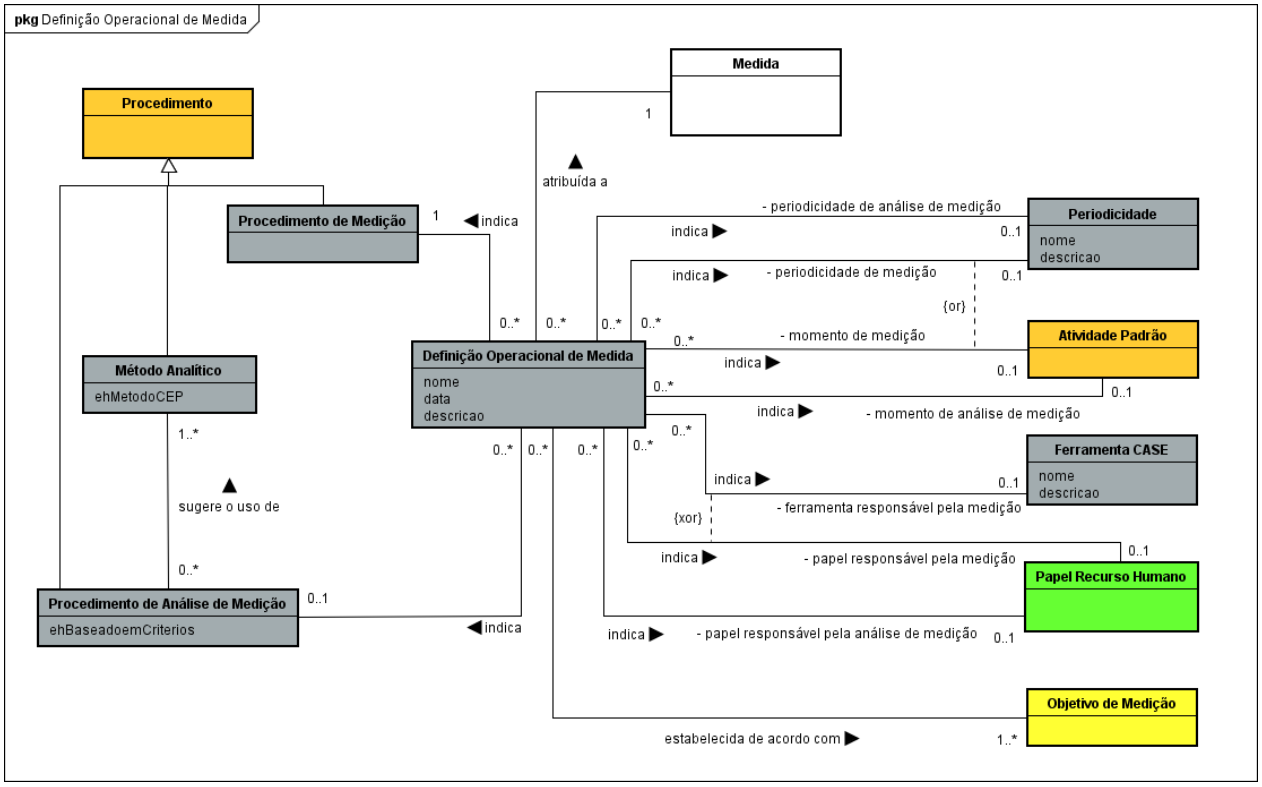

Figura 3.15 - Diagrama de Classes do Subsistema de Definição Operacional de Medida.

Medidas possuem **Definições Operacionais de Medida**, que fornecem informações relacionadas à coleta e à análise da medida. Definições operacionais de medidas são estabelecidas de acordo com um ou mais objetivos de medição. Uma definição operacional de medida deve indicar:

- **Procedimento de Medição**: um procedimento de medição é um tipo de procedimento (Procedimento) que define uma sequência lógica de operações que devem ser executadas para atribuir um valor a uma medida. Ou seja, define o que deve ser feito para que seja possível coletar dados para uma medida. Caso a medida seja medida derivada, o procedimento de medição deve incluir uma ou mais fórmulas de cálculo de medida.
- **Momento de Medição**: indica em que atividade (Atividade Padrão) a medição de uma medida deve ser realizada.
- **Periodicidade de Medição**: indica a periodicidade (Periodicidade) em que a medição da medida deve ser realizada.
- **Papel Responsável pela Medição**: indica o papel do recurso humano (Papel Recurso Humano) que é responsável pela execução da medição.
- **Ferramenta CASE Responsável pela Medição**: indica a ferramenta CASE (Ferramenta CASE) responsável pela execução da medição, caso a coleta de dados para a medida seja feita automaticamente.
- **Procedimento de Análise de Medição**: um procedimento de análise de medição é um procedimento que define uma sequência lógica de operações utilizada para representar e analisar valores medidos para uma medida.
- **Momento de Análise de Medição**: indica em que atividade (Atividade Padrão) a análise dos dados coletados para uma medida deve ser realizada.
- **Periodicidade de Análise de Medição**: indica a periodicidade (Periodicidade) em que a análise de medição deve ser realizada.
- **Papel Responsável pela Análise de Medição**: indica o papel do recurso humano (Papel Recurso Humano) que é responsável pela execução da análise medição.

Procedimentos de análise de medição podem sugerir o uso de **Métodos Analíticos** para representar e analisar os valores medidos. Histograma e gráfico de barras são exemplos de métodos analíticos (FLORAC; CARLETON, 1999). Um método analítico pode ser um método do controle

estatístico de processos (*ehMetodoCEP*) quando utiliza os princípios do controle estatístico para representar e analisar valores.

Em relação ao modelo de classes definido em (MARETTO, 2013), neste trabalho foram feitas as seguintes alterações: inclusão da classe Ferramenta CASE, para permitir que ferramentas possam ser identificadas como responsáveis pela coleta de dados para medidas onde as medições são feitas automaticamente. Além disso, foi eliminada a obrigatoriedade de se definir o momento da medição e a periodicidade de medição, sendo permitido informar apenas um deles. Assim, organizações que não possuem um processo padrão definido podem estabelecer definições operacionais para as medidas informando apenas a periodicidade da medição.

3.4.6 Diagrama de Classes do Subsistema Plano de Medição

A Figura 3.16 apresenta o diagrama de classes do subsistema Plano de Medição.

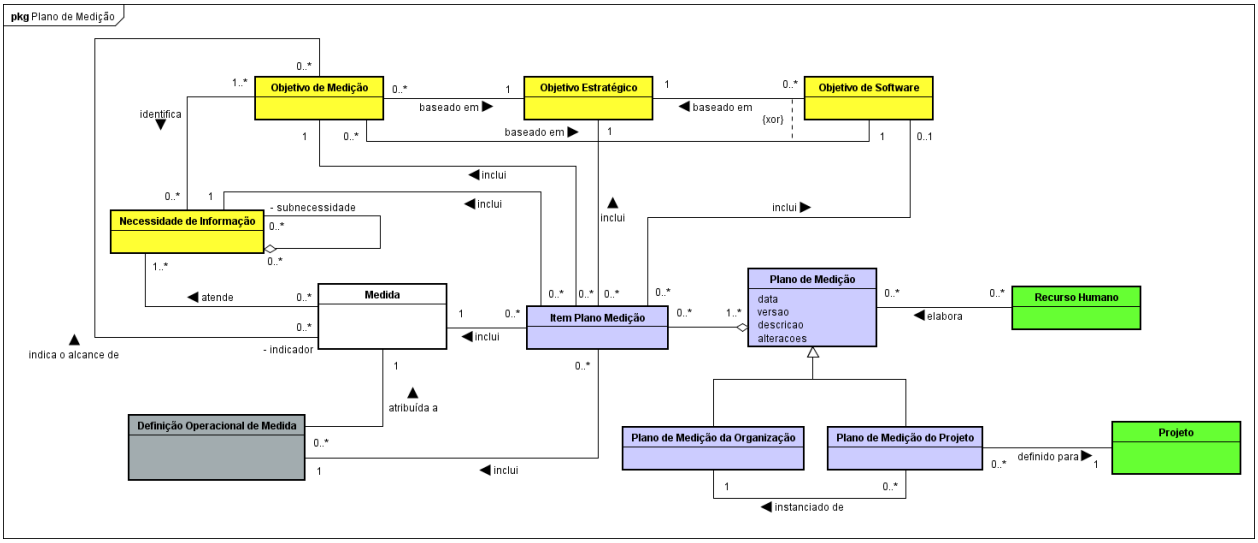

Figura 3.16 - Modelo de Classes do Subsistema Plano de Medição.

Um **Plano de Medição** é elaborado por um ou mais recursos humanos (**Recurso Humano**) e é o resultado da atividade de planejamento da medição. Ele reúne todas as informações relevantes para que a medição possa ser realizada. Um **Plano de Medição da Organização** é um plano que define o padrão para a realização da medição em uma organização. A partir desse plano são instanciados **Planos de Medição do Projeto**, que são planos estabelecidos para cada projeto da organização.

Um Plano de Medição é composto por **Itens Plano Medição.** Um Item Plano Medição inclui um **Objetivo Estratégico**, um **Objetivo de Software** (opicional, definido com base no objetivos estratégico), um **Objetivo de Medição** (definido com base nos anteriores), uma **Necessidade de Informação** (identificada a partir do objetivo de medição), uma medida (para atender a necessidade de informação) e uma **Definição Operacional de Medida** (a ser utilizada para a coleta e análise de dados da medida).

Em relação ao modelo de classes definido em (MARETTO, 2013), neste trabalho foram feitas as seguintes alterações: inclusão da classe Item de Plano de Medição e seus relacionamentos em substituição à classe Medida Plano Medição, que não apresentava todos os relacionamentos necessários à definição de um Plano de Medição.

3.4.7 Diagrama de Classes do Subsistema Medição

A Figura 3.17 apresenta o diagrama de classes do subsistema Medição.

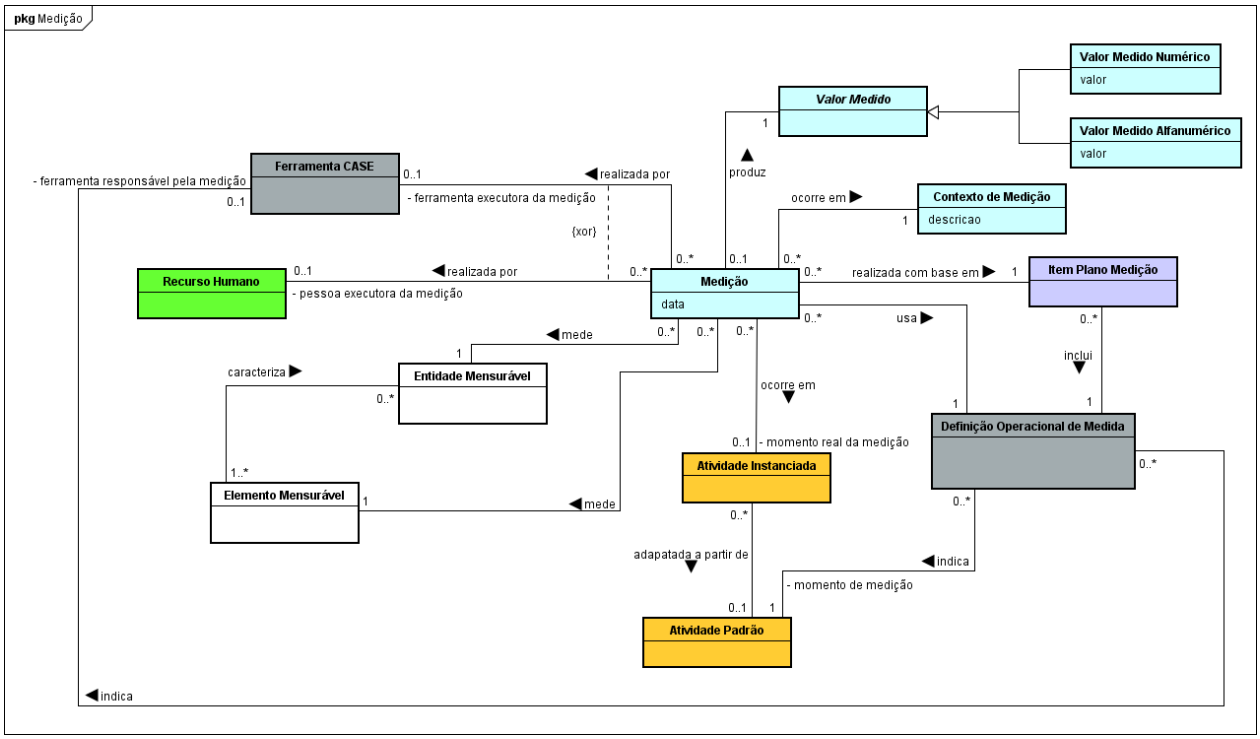

Figura 3.17 - Diagrama de Classes do Subsistema Medição.

Uma **Medição** é o ato de usar uma **Medida** para medir um **Elemento Mensurável** de uma **Entidade Mensurável**, produzindo-se um valor (**Valor Medido**), podendo ele ser numérico (**Valor Medido Numérico**) ou alfanumérico (**Valor Medido Alfanumérico**). Uma Medição ocorre em um determinado **Contexto de Medição**, que descreve as condições sob as quais a medição foi realizada.

Uma **Medição** é realizada seguindo um determinado **Item de Plano de Medição** e, dessa forma, se a Medição aplica uma determinada **Medida**, a **Definição Operacional** utilizada na medição deve ser aquela que consta no Plano de Medição considerado.

Uma Medição é executada por um **Recurso Humano** (**pessoa executora da medição**) ou, quando é realizada automaticamente, por uma **Ferramenta CASE** (**ferramenta executora da medição**). Uma Medição pode ser realizada em uma Atividade **Instanciada**, que representa o **momento real da medição**. Idealmente, a **pessoa executora da medição** deve desempenhar o mesmo **Papel Recurso Humano** indicado na definição operacional da medida e a Atividade Instanciada deve ser adaptada da Atividade Padrão que descreve o momento de medição na definição operacional da medida.

Em relação ao modelo de classes definido em (MARETTO, 2013), neste trabalho foram feitas as seguintes alterações: inclusão da relação entre Medição e Ferramenta CASE, para permitir que ferramentas possam ser identificadas como executoras de medições feitas automaticamente, e eliminação da obrigatoriedade de se identificar o momento real da medição, para permitir que organizações que não possuem um processo padrão definido possam registrar suas medições.

3.4.8 Diagrama de Classes do Subsistema Análise de Medição

A Figura 3.18 apresenta o diagrama de classes do subsistema Análise de Medição.

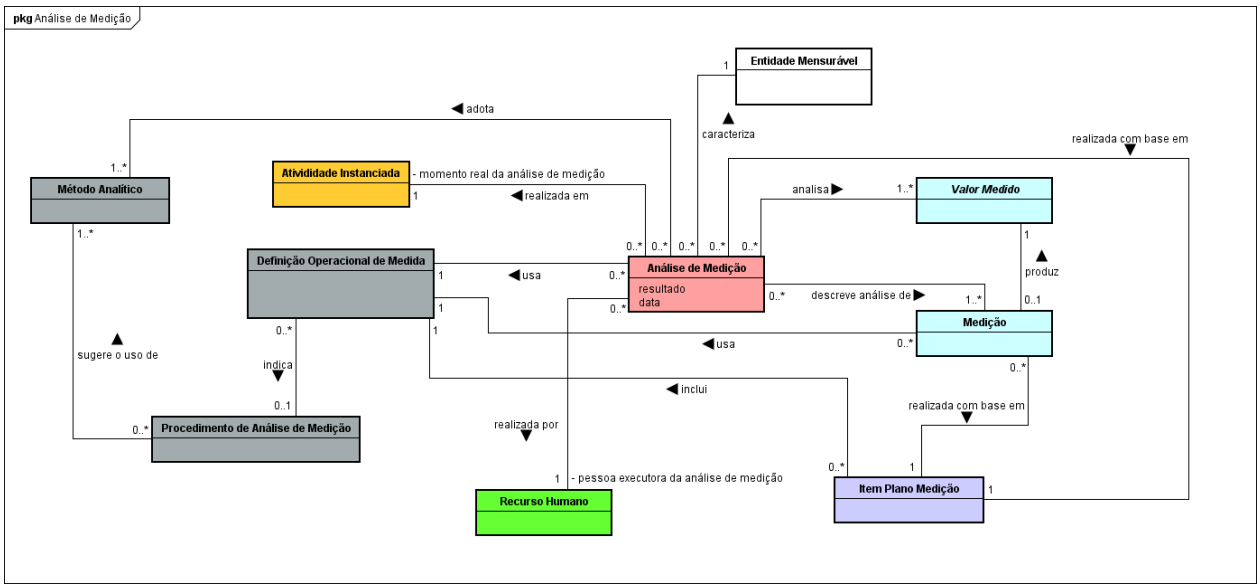

Figura 3.18 - Diagrama de Classes do Subsistema Análise de Medição.

Uma **Análise de Medição** analisa **Valores Medidos** para uma Medida a fim de obter um **resultado** que, de alguma forma, caracterize a **Entidade Mensurável** que foi medida.

Uma Análise de Medição é realizada seguindo um determinado **Item Plano Medição** e, dessa forma, se ela aplica uma determinada **Medida** e a **Definição Operacional** considerada na análise de medição deve ser aquela que consta no Plano de Medição utilizado.

Uma Análise de Medição pode adotar um **Procedimento de Análise de Medição** que deve ser o mesmo indicado na Definição Operacional de Medida utilizada. Além disso, uma Análise de Medição usa **Métodos Analíticos**, os quais devem ser métodos sugeridos no Procedimento de Análise de Medição utilizado.

De maneira similar à Medição, uma Análise de Medição é executada por um **Recurso Humano**, que atua como **executor da análise de medição** e, idealmente, deve desempenhar o mesmo **Papel Recurso Humano** indicado pela Definição Operacional de Medida para o responsável pela análise de medição. Uma Análise de Medição pode ser realizada em uma **Atividade Instanciada**, que representa o momento real da análise de medição, e que, idealmente, deve ser

adaptada da **Atividade Padrão** indicada pela definição operacional de medida como o momento da análise de medição.

As alterações realizadas em relação ao modelo definido em (MARETTO, 2013) são apenas reflexo das alterações realizadas nos modelos apresentados anteriormente (por exemplo, substituição da classe Medida Plano de Medição por Item de Plano de Medição e adequação dos relacionamentos).

3.4.9 Diagrama de Classes do Subsistema Comportamento Processo de Software

A Figura 3.19 apresenta o diagrama de classes do subsistema Comprortamento Processo de Software. Não foram feitas alterações nesse modelo de classes, tendo sido mantido o modelo definido em (MARETTO, 2013).

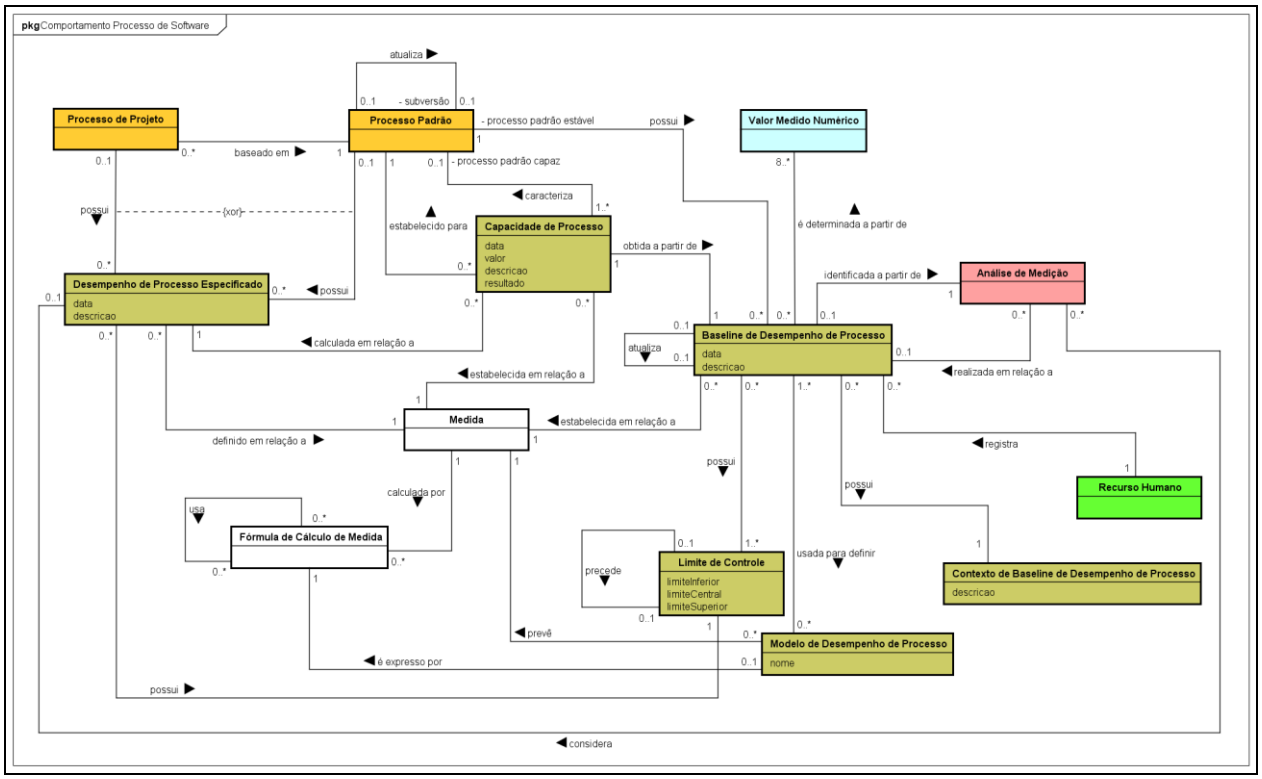

Figura 3.19 - Diagrama de Classes do Subsistema Comportamento Processo de Software.

Em uma Análise de Medição que adota um Método Analítico do controle estatístico pode-se identificar uma **Baseline de Desempenho de Processo** para um **Processo Padrão**, relativa a uma

Medida. Uma Baseline de Desempenho de Processo é obtida a partir de um conjunto de **Valores Medidos Numéricos**.

Quando um processo tem uma baseline de desempenho estabelecida ele é um processo estável (processo padrão estável). Uma baseline é definida por **Limites de Controle**. Uma baseline de desempenho de processo é estabelecida por um **Recurso Humano** em um **Contexto de Baseline de Desempenho de Processo**.

A Baseline de Desempenho de Processo de um Processo Padrão pode ser utilizada em uma Análise de Medição que analisa o comportamento de um Processo de Projeto que foi baseado naquele Processo Padrão. Para analisar o desempenho de um processo de projeto em relação a uma baseline de desempenho, os limites dessa baseline devem ser utilizados como limites de controle do método analítico do controle estatístico utilizado para representar os valores medidos que estão sendo analisados.

Baselines de desempenho de processo podem ser usadas na definição de **Modelos de Desempenho de Processo**. Um Modelo de Desempenho de Processo é expresso por uma fórmula que prevê o valor de uma medida derivada a partir de outras. A quantificação da relação entre essas medidas é obtida a partir dos valores medidos para essas medidas.

Desempenho de Processo Especificado é o intervalo de resultados que se espera que um processo (Processo Padrão ou Processo de Projeto) alcance considerando uma Medida específica, sendo definido por limites de controle. Quando é esperado que um Processo de Projeto apresente desempenho diferente daquele especificado para seu Processo Padrão, é estabelecido um Desempenho de Processo Especificado para o Processo de Projeto. Uma análise de medição que analisa o comportamento de um Processo de Projeto pode ser realizada considerando um Desempenho de Processo Especificado estabelecido para o Processo Padrão no qual o Processo de Projeto se baseou ou para o Processo de Projeto sendo analisado.

A **Capacidade de Processo** determina a habilidade de um processo padrão estável atender a um desempenho de processo para ele especificado. Ela é obtida a partir de uma Baseline de Desempenho de Processo e calculada em relação a um Desempenho de Processo Especificado. Uma Capacidade de Processo é estabelecida em relação a uma Medida, que deve ser a mesma medida considerada pela Baseline de Desempenho de Processo e pelo Desempenho de Processo Especificado. Quando a capacidade de um processo mostra que ele é capaz de atender ao desempenho de processo considerado, tem-se um processo padrão capaz.

3.4.10 Diagrama de Classes do Subsistema Caracterização de Projeto

A Figura 3.20 apresenta o diagrama de classes do subsistema Caracterização de Projeto. Não foram feitas alterações nesse modelo de classes, tendo sido mantido o modelo definido em (MARETTO, 2013).

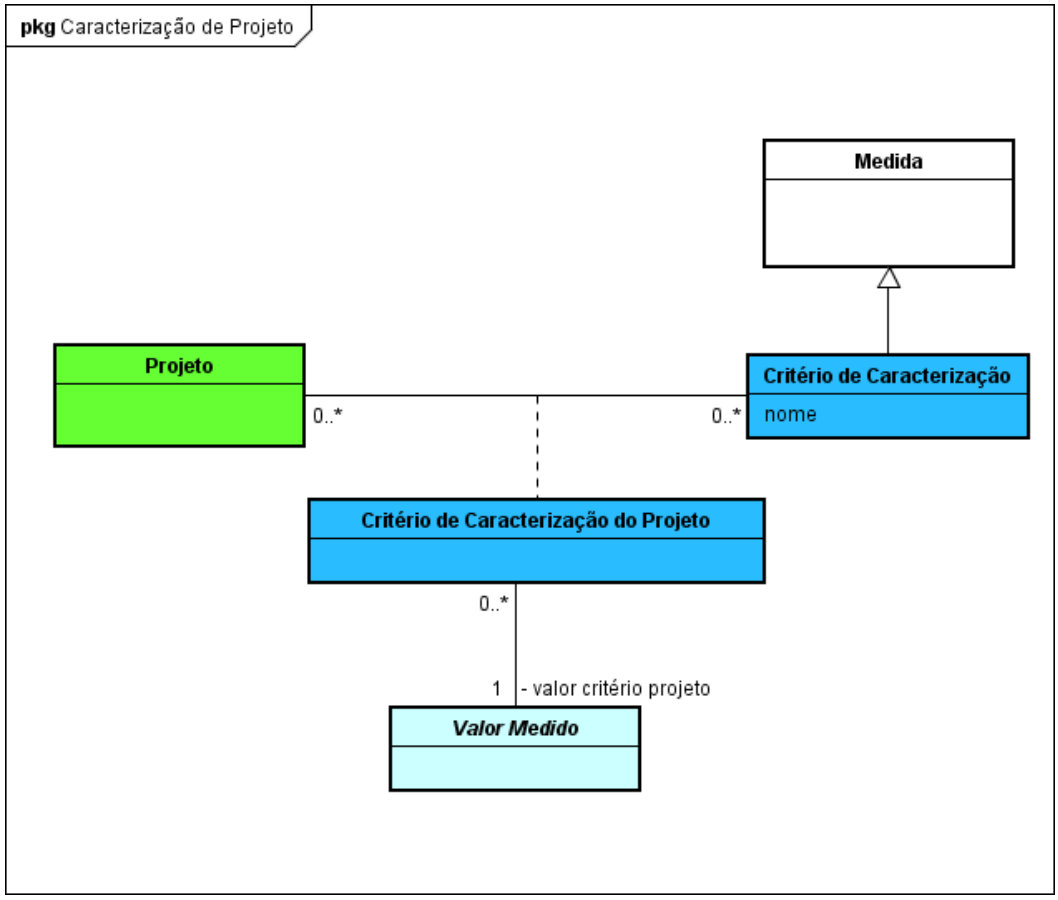

Figura 3.20 - Diagrama de Classes do Subsistema de Caracterização de Projeto.

Um **Critério de Caracterização** é uma **Medida** que é utilizada para caracterizar **Projetos**. A principal diferença entre Medida e Critério de Caracterização é que a atribuição de um valor medido a um critério em um projeto não requer que esse critério tenha sido definido em um plano de medição, nem que seja registrada uma medição, como descrito no pacote Medição. Além disso, não é necessário estabelecer Definição Operacional de Medida para um Critério de Medição. O valor atribuído ao critério de caracterização no contexto de um determinado Projeto (**Critério de Caracterização do Projeto**) é chamado **valor critério projeto**.

3.4.11 Diagrama de Classes do Subsistema Mediador

Para facilitar o entendimento e a visualização, o diagrama de classes do subsistema Mediador foi dividido em duas figuras. A Figura 3.21 apresenta o diagrama de classes referente ao caso de uso *Gerar Novo Plano de Medição Integrado para Projetos*. O diagrama apresenta as classes envolvidas na execução desse caso de uso. Algumas relações apresentadas nos modelos anteriores foram omitidas para diminuir a poluição visual. Apenas **Item do Plano de Medição Integrado** é classe oriunda do subsistema Mediador. Ela é reponsável por armazenar os itens (objetivos, necessidades de informação, medidas e definições operacionais de medidas) dos planos de medição gerados a partir da integração.

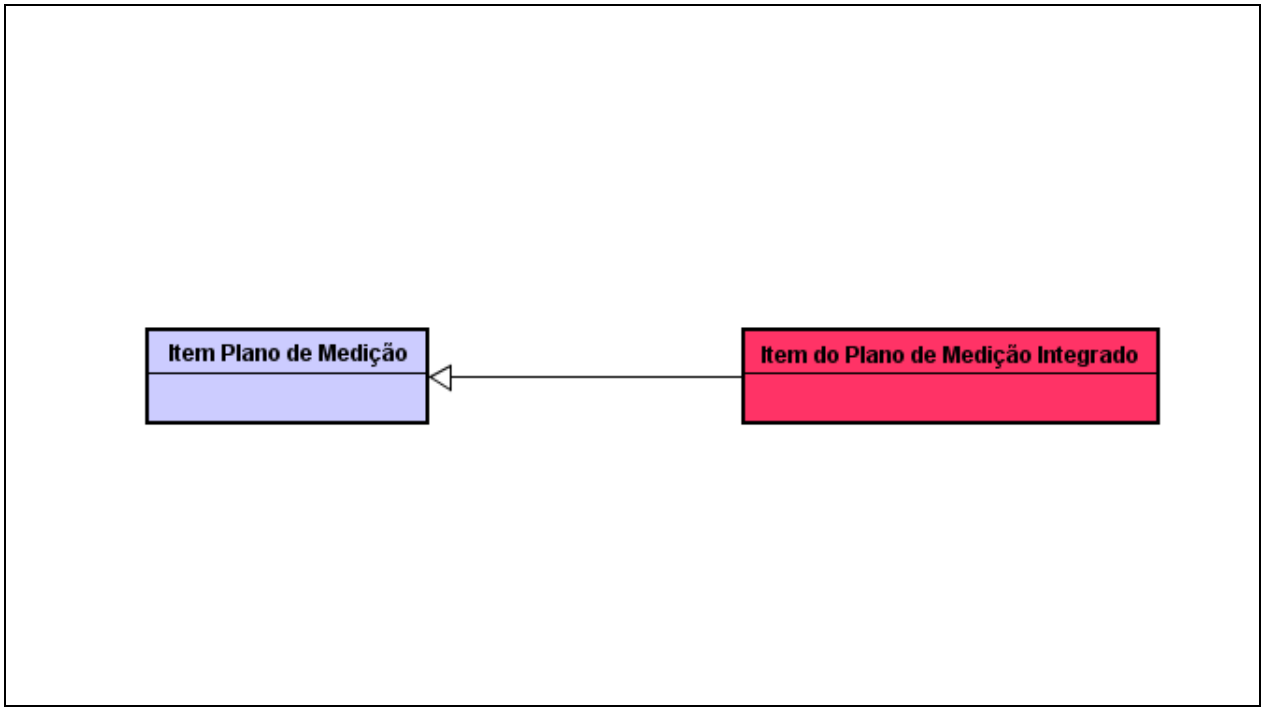

Figura 3.21 - Diagrama de Classes Novo Plano de Medição Integrado.

A Figura 3.22 apresenta o diagrama de classes referente ao caso de uso *Agendamento de Medições*. As classes **Taiga Medição Job** e **SonarQube Medição Job** são responsáveis pelos *jobs* de medição que permitirão a realização das medições automáticas pelas **Ferramentas CASE associadas**. Na figura, algumas relações apresentadas em modelos anteriores são omitidas para diminuir a poluição visual.

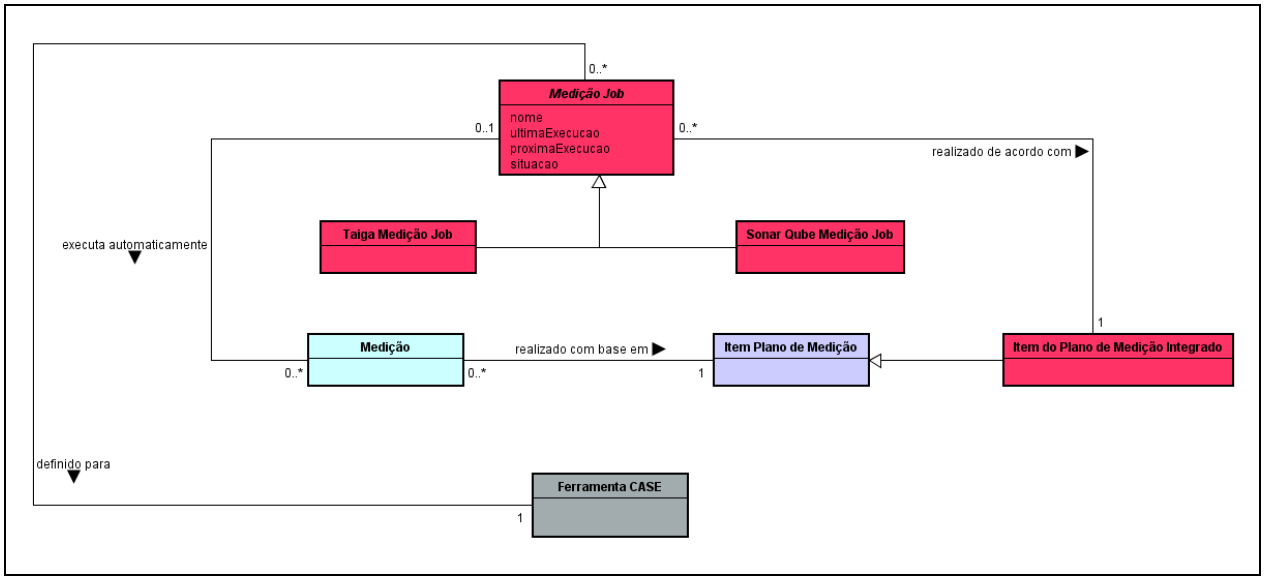

Figura 3.22 - Diagrama de Classes do Agendamento de Medições.

Capítulo 4
Implementação da Ferramenta SoMeSPC

Neste capítulo são apresentadas algumas informações sobre a implementação de SoMeSPC, incluindo tecnologias utilizadas (seção 4.1) e algumas telas do sistema (seção 4.2).

4.1 TECNOLOGIAS ENVOLVIDAS

A versão inicial da ferramenta, desenvolvida em (MARETTO, 2013), utilizou diversas tecnologias descritas nos parágrafos a seguir.

O *framework* principal da ferramenta é o OpenXava, responsável por disponibilizar o sistema em forma de *portlets* Java. Os *portlets* Java são pequenas aplicações web independentes que interagem entre si dentro de uma aplicação maior, um portal. Com base nas classes do domínio anotadas com os *annotations* do *framework*, o OpenXava produz o CRUD (*Create*, *Read*, *Update* e *Delete*) e gera o banco de dados da ferramenta, que é suportado pelo PostgreSQL[8], um sistema de gestão de banco de dados relacionais que, além de ser um software livre, possui bibliotecas para diversos tipos de linguagens, inclusive Java.

A lógica de negócio é contemplada pelo *framework* por meio de *Actions* (permite customizar e tratar ações disparadas pelo usuário, por exemplo, clique no botão delete), *Validators* (permite validar uma entidade antes do salvamento) e *Views* (define como e quais dados serão exibidos). O OpenXava não disponibiliza recursos de geração de gráficos. Portanto, para preencher essa lacuna do *framework*, foi adotado o RGraph[9], uma biblioteca JavaScript para geração de gráficos. Os *portlets* resultantes são adicionados ao portal LifeRay[10]. O LifeRay é um portal que dispõe aplicações integradas a um sistema para trabalho colaborativo. Algumas dessas aplicações são: gestor de conteúdo, fórum e armazenamento de documentos. O portal é responsável por estruturar a aplicação, definindo o menu e navegabilidade para cada *portlet*.

[8] *http://www.postgresql.org*

[9] *http://www.rgraph.net/*

[10] *http://www.liferay.com*

Para o desenvolvimento de SoMeSPC decidiu-se por abandonar o uso da tecnologia de *portlets* devido à inviabilidade de uso do Portal LifeRay no contexto de (FONSECA, 2015). A plataforma da ferramenta foi, então, adaptada para utilizar o servidor de aplicação web Apache Tomcat e outras tecnologias de aplicações web em Java.

Para a atualização da plataforma, foi necessário atualizar a versão do *framework* OpenXava para permitir a reescrita dos *portlets* para serem exibidos no formato de páginas JSP, as quais são tratadas e exibidas pelo Apache Tomcat. Também foi necessário fazer a portabilidade do sistema gerenciador de banco de dados originalmente usado (PostgreSQL) para MySQL[11], uma vez que esse SGBD estava configurado e disponível no servidor web do LEDS e do NEMO.

Um dos requsitos da inicitativa de integração de ferramentas conduzida em (FONSECA, 2015) era realizar a integração na camada de serviços. Dessa forma, SoMeSPC deveria ser capaz de disponibilizar serviços. Então, foi desenvolvida uma API de serviços seguindo os princípios do padrão arquitetural REST (*Representational State Transfer*) (FIELDING, 2000), incluindo serviços relacionados à criação e obtenção de entidades mensuráveis, projetos, recursos humanos, equipes, medidas, medições, plano de medição, entre outros. Foi utilizado o *framework Jersey*[12], desenvolvido para permitir a criação de serviços web baseados na arquitetura REST. Para acesso aos serviços, o cliente REST deve usar um dos verbos do protocolo HTTP (*HyperText Transfer Protocol*) para a URI (*Uniform Resource Identification*) do serviço, sendo mais comuns os verbos GET, para obter dados, e POST, para enviar dados. A notação utilizada para a comunicação com os serviços de SoMeSPC é a JSON (*JavaScript Object Notation*). A Tabela 4.1 exibe as APIs que foram desenvolvidas.

[11] *http://www.mysql.com*

[12] *http://jersey.java.net*

Tabela 4.1 – APIs REST SoMeSPC.

APIs	Serviço	URL
obterItensPlanoDeMedicao()	Do tipo GET, essa API obtém os itens essenciais para a composição de um plano de medição, sendo eles: Medidas, Objetivos (Estratégico e Medição) e Necessidade de Informação.	*SoMeSPC/api/ItensPlanoDeMedicao*
obterPeriodicidades()	Do tipo GET, essa API obtém as periodicidades.	*SoMeSPC/api/Periodicidade*
criarPlanoMedicao(PlanoDTO)	Do tipo POST, essa API recebe um PlanoDTO, composto por um conjunto de dados que compõe a criação de um plano de medição do projeto	*SoMeSPC/api/Plano*
obterEntidadesComMedicoes(String)	Do tipo GET, essa API recebe o nome da medida e retorna todas as entidades com medições que contém aquele nome.	*SoMeSPC/api/Medicao/Entidade*
obterMedicoes(String, int, int, int, String, String)	Do tipo GET, essa API recebe o nome da medida, o id da entidade mensurável, índice atual dá pagina, tamanho da página e as datas de início e fim da medida e retorna todas as medições que estão relacionadas a aquela medida dentro do intervalo das datas e contendo o id da entidade mensurável especificado.	*SoMeSPC/api/Medicao*
obterTotalMedicoes(String, int, String, String)	Do tipo GET, essa API recebe o nome da medida, o id da entidade mensurável e as datas de início e fim da medida e retorna a quantidade das medições relacionadas a essa medida dentro do intervalo das datas e contendo o id da entidade mensurável especificado.	*SoMeSPC/api/Medicao/Total*

As APIs criadas permitiram ao Mediador intermediar a integração de SoMeSPC com as demais ferramentas envolvidas na iniciativa de integração descrita em (FONSECA, 2015). A Figura 4.1 ilustra a interação do Mediador com as APIs de serviços das ferramentas integradas.

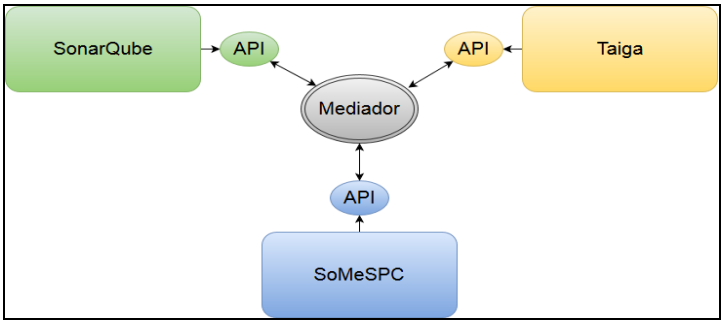

Figura 4.1 - Interação do Mediador com as APIs de serviços.

O desenvolvimento do Mediador se deu através de uma extensão de SoMeSPC. Foram adicionados pacotes de classes em SoMeSPC para permitir que o mediador seja executado sempre que SoMeSPC esteja ativa.

Para a construção da interface gráfica do Mediador, foram utilizados os *frameworks Google AngularJS* e *Twitter Bootstrap*. A combinação desses *frameworks* possibilita a criação de interfaces web ricas e responsivas, ou seja, permiti que a programação da interface tenha os elementos que a compõem se adaptando automaticamente à largura de tela do dispositivo que está sendo visualizado, facilitando o acesso por dispositivos móveis como *tablets* e *smartphones*. O *Google AngularJS* é executado pelo navegador web e é usado pelo mediador para a comunicação entre o cliente e as APIs de serviços das ferramentas. O *Twitter Bootstrap*, no mediador, foi utilizado para estilização das interfaces gráficas.

4.1 ALGUMAS TELAS DA FERRAMENTA

A evolução de Med&CEP em SoMeSPC preservou todas as funcionalidades originalmente definidas em Med&CEP, tendo sido feitas melhorias de interface e ajustes para favorecer a usabilidade. Por exemplo, a estrutura do menu foi alterada, sendo eliminadas algumas subcategorias para torná-lo mais simples e facilitar o acesso aos itens do menu. Além disso, passou-se a possibilitar o acesso a algumas telas a partir de outras, evitando que o usuário precise retornar ao menu principal com frequência. A Figura 4.2 ilustra o menu principal de SoMeSPC.

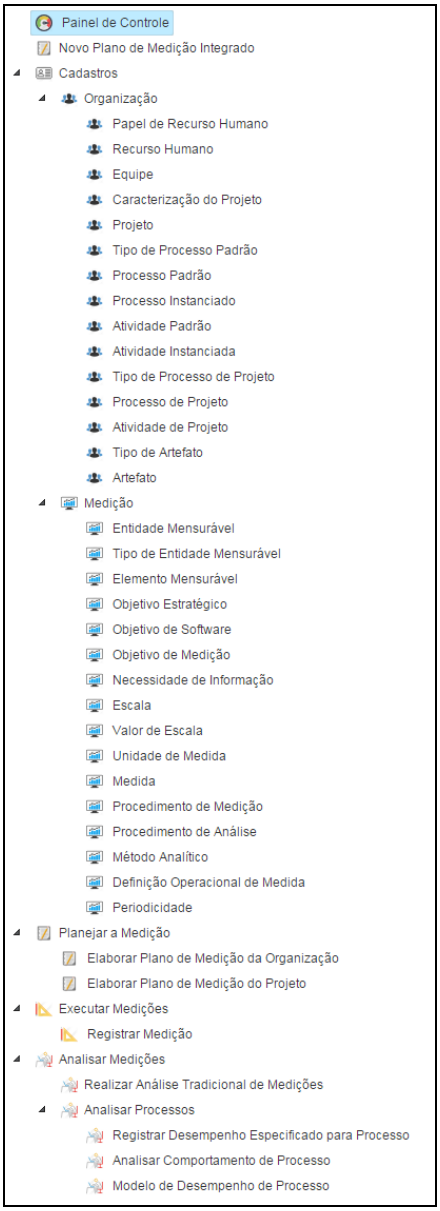

Figura 4.2 - Menu em SoMeSPC.

Além das funcionalidades definidas na versão anterior da ferramenta, foram desenvolvidas as funcionalidades do Mediador, para propiciar a integração com SonarQube e Taiga. Uma vez que essas são novas funcionalidades providas pela ferramenta, a seguir são apresentadas as telas a elas relacionadas.

Conforme apresentado no capítulo anterior, as funcionalidades providas pelo mediador são: (i) criação de Plano de Medição Integrado para projetos; (ii) agendamento de medições automáticas; (iii) apresentação de dados para análise das medições. As funcionalidades (i) e (iii) foram

desenvolvidas a partir de adaptações nas funcionalidades *Elaborar Plano de Medição do Projeto* e *Analisar Medições*, presentes respectivamente nos subsistemas Plano de Medição e Análise de Medições.

As funcionalidades providas pelo Mediador são disponibilizadas a partir do menu de SoMeSPC, por meio das opções *Novo Plano de Medição Integrado*, que apoia a criação de planos de medição, e *Painel de Controle*, que permite controlar o agendamento de medições e realizar a análise de medições. A Figura 4.3 apresenta o menu de SoMeSPC, incluindo as opções para acessar as funcionalidades do Mediador.

Figura 4.3 - Menu de SoMeSPC incluindo itens para acesso às funcionalidades do mediador.

O Mediador disponibiliza um *wizard* para guiar o usuário. No seu desenvolvimento foi utilizada uma extensão do *framework AngularJS*, o *Angular Wizard*[13], para permitir a adaptação dinâmica dos próximos passos a serem realizados de acordo com os dados selecionados pelo usuário. As Figuras 4.4 e 4.5 apresentam algumas das telas do *wizard* para criação de Plano de Medição. Na Figura 4.4 é apresentada a tela onde são selecionados os objetivos, necessidades de informação e medidas para o Plano de Medição e é informada a periodicidade de coleta das medidas. Na Figura 4.5 é apresentada a tela com o resumo do Plano de Medição criado, mostrando as medidas incluídas no plano e os projetos para os quais foram definidos planos de medição.

[13] http://mgonto.github.io/angular-wizard

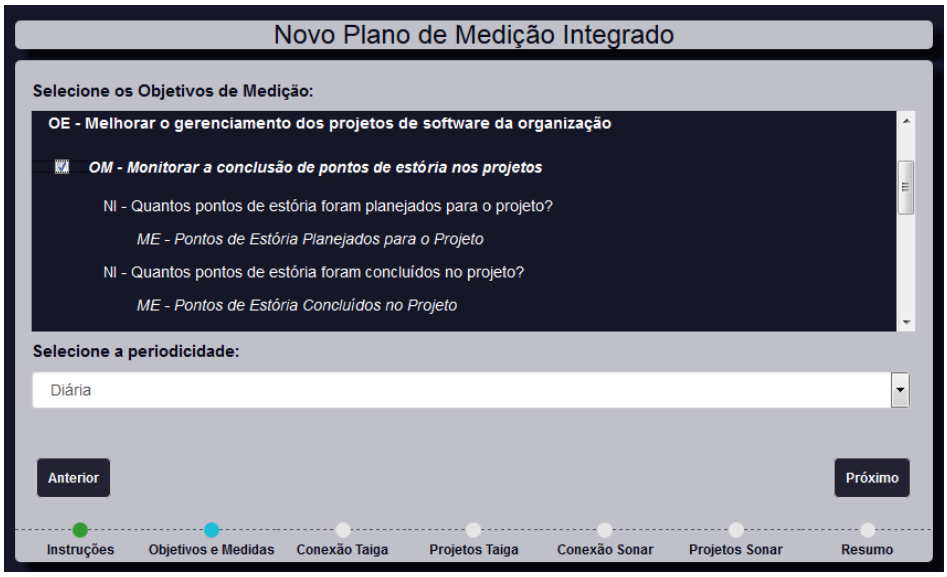

Figura 4.4 - Tela para seleção de objetivos e medidas para o Novo Plano de Medição Integrado.

Figura 4.5 - Tela com o resumo do Novo Plano de Medição Integrado.

Uma vez definido o Plano de Medição para um projeto, são feitos agendamentos para a realização de medições automáticas das medidas presentes no Plano de Medição, considerando a periodicidade para elas determinada. Os agendamentos são realizados como *jobs* de medição e foram implementados com auxílio do *framework Quartz Scheduler*. Foram criadas duas classes representando os *jobs* para cada ferramenta: *TaigaMedicaoJob* e *SonarQubeMedicaoJob*. Cada uma das classes implementa o método *execute()* do *framework*. Esse método é acionado sempre que a periodicidade de medição

ocorre. Por exemplo, se a periodicidade da medição é diária, a cada 24 horas o método execute do respectivo *job* é acionado. Ao final da criação de um Plano de Medição, para cada medida incluída no plano, uma instância do *job* da ferramenta que provê os dados para a medida é criada. Dessa forma, o método *execute* dos *jobs* de medição é o responsável pelas medições e possui o conhecimento de qual serviço chamar, como obter os valores da medição e como armazená-los. A Figura 4.6 é apresenta a tela com o painel de agendamento de medições.

Figura 4.6 - Agendamento de Medições do Mediador.

O mediador permite que o usuário identifique o objetivo de medição a ser monitorado, a medida a ser analisada e os dados a serem analisados informando a entidade mensurável, o período ao qual os dados se referem ou a quantidade de dados a ser considerada. A partir dessas informações são apresentados gráficos com os valores medidos selecionados para que o usuário realize a análise das medições. A Figura 4.7 apresenta a tela com a representação de dados para análise de medições.

Figura 4.7 – Apresentação de Dados para Análise de Medições.

Capítulo 5
Considerações Finais

Neste capítulo são realizadas as considerações finais deste trabalho, sendo apresentadas suas principais contribuições e perspectivas de trabalhos futuros.

5.1 CONCLUSÕES

A eficiência da medição de software está fortemente relacionada ao apoio de ferramentas associadas a todo o processo de medição (DUMKE; EBERT, 2007). Mesmo que as ferramentas utilizadas em uma organização não sejam específicas para apoiar o processo de medição de software, muitas vezes elas fornecem dados relacionados à medição, que são úteis para a tomada de decisão. Entretanto, essas ferramentas precisam ser integradas para que possam apoiar o processo de medição como um todo.

Este trabalho teve como objetivo evoluir a ferramenta Med&CEP (MARETTO, 2013) para ser utilizada em uma iniciativa de integração para apoiar o processo de medição de software no contexto de (FONSECA, 2015). O objetivo geral deste trabalho foi detalhado em 4 objetivos específicos, sendo que todos foram alcançados neste trabalho. A Tabela 5.1 apresenta os objetivos específicos do trabalho e o principal produto que serve como evidência do alcance de cada objetivo.

Tabela 5.1 – Objetivos Específicos do Trabalho.

Objetivos	**Resultado**
Adaptar as tecnologias da ferramenta, contemplando melhorias de desempenho e usabilidade.	Utilização no desenvolvimento de SoMeSPC das Tecnologias para Aplicações Web apresentadas no Capítulo 2.
Adequar a conceituação à nova versão da Ontologia de Referência para Medição de Software.	Modificações realizadas nos modelos de classes (Vide Capítulo 3)
Desenvolver uma API (*Application Programming Interface*) de serviços para expor funcionalidades de medição necessárias para a integração.	Implementação da API de serviços. (Vide Capítulo 4)
Desenvolver um mediador na ferramenta para integrar com as ferramentas SonarQube e Taiga.	Definição dos casos de uso, modelos de classes (vide Capítulo 3) e implementação do Mediador (Vide Capítulo 4)

Pode-se dizer que a experiência adquirida com a realização desse trabalho foi muito grande. Vale a pena destacar a integração das diversas disciplinas e conceitos vistos durante toda a graduação, como Programação III, Engenharia de Software, Engenharia de Requisitos de Software, Banco de Dados e Projeto de Sistemas, que serviram como principal base de conhecimento para a construção desse projeto. Para a execução deste trabalho passou-se por todas as fases do processo de desenvolvimento de software (especificação de requisitos, análise, projeto do sistema, implementação e testes). Além disso, foi muito importante a utilização de várias ferramentas e *frameworks*, até então desconhecidos para o autor deste trabalho. Não menos importante, tem-se o conhecimento obtido através da revisão de literatura acerca de medição de software e integração.

5.2 TRABALHOS FUTUROS

No final do desenvolvimento de um software, tipicamente novas necessidades são identificadas. A manutenção e a evolução de software devem ser um trabalho constante, de forma que o ciclo de vida não finalize na homologação, mas permaneça ao longo de toda a vida do software.

Uma evolução seria o mecanismo de autenticação, que hoje é gerenciado diretamente pelo desenvolvedor, cadastrando usuário e senha em arquivo dentro do projeto da ferramenta. O ideal seria a criação de um módulo para esse mecanismo, onde pudessem ser cadastrados usuários e suas respectivas permissões. Além disso a ferramenta pode incorporar geração de relatórios, o que facilita a gestão dos resultados de medições e análises. O *framework* OpenXava possui recursos que permitem esse tipo funcionalidade. Por fim, outras ferramentas podem ser integradas a SoMeSPC, como:

- Tuleap[14], uma ferramenta de gerenciamento de ciclo de vida de aplicativos, incluindo suporte para o desenvolvimento ágil e gerenciamento de projetos.
- OrangeScrum[15], uma ferramenta de gerenciamento de projeto moderno para freelancers, agências, e pequenas e médias empresas. As características incluem quadro de tarefas, planejamento de recursos, acompanhamento de progresso, e muito mais.
- Agilefant[16], uma ferramenta baseada nas metodologias ágeis de gerenciamento de projetos, possui uma plataforma *open source* que permite gerir portfólios de projetos, *sprints*, tarefas e cronogramas com facilidade.

[14] *https://www.tuleap.org/*

[15] *http://www.orangescrum.com/*

Referências

BASS, L.; BELADY, L.; BROWN, A.; FREEMAN, P.; ISENSEE, S.; KAZMAN, R.; KRASNER, H.; MUSA, J.; PFLEEGER, S.; VREDENBURG, K.; WASSERMAN, T. **Constructing Superior Software.** Software Quality Institute Series, Macmillan Technical Publishing. 1999.

BARCELLOS, M. P.; FALBO, R. A.; DALMORO, R. **A Well-Founded Software Measurement Ontology**. In: PROCEEDINGS OF THE 6TH INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY IN INFORMATION SYSTEMS (FOIS 2010), Toronto, Canada, 2010, v. 209, pp. 213-226.

BARCELLOS, M. P. **Estratégia para Medição de Software e Avaliação de Bases de Medidas para Controle Estatístico de Processos de Software em Organizações de Alta Maturidade**. 2009. Tese de Doutorado. Sc. PESC, COPPE/UFRJ, Rio de Janeiro, 2009.

BARCELLOS, M.P.; FALBO, R.A. **A Software Measurement Task Ontology**. In: PROCEEDINGS OF THE ACM SAC 2013 28TH SYMPOSIUM ON APPLIED COMPUTING, Coimbra, Portugal, 2013.

BRINGUENTE, A. C. O; FALBO, R. A.; GUIZZARDI, G. **Using a foundational ontology for reengineering a software process ontology**. In: PROCEEDINGS OF THE XXVI BRAZILIAN SYMPOSIUM ON DATA BASE, Florianópolis, Brasil, 2011, pp. 1-16.

DE LUCIA, A.; POMPELLA, E.; STEFANUCCI, S. **Assessing the Maintenance Process of a Software Organization: an Empirical Analysis of a Large Industrial Project**. The Journal of Systems and Software, v. 65, 2013, pp. 87-103.

DUMKE, R.; EBERT, C. **Software Measurement: Establish - Extract - Evaluate - Execute**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

FALBO, R. A.; RUY, F.; GUIZZARD, G.; BARCELLOS, M. P.; ALMEIDA, J. P. **Towards an Enterprise Ontology Pattern Language**. Paper presented at the 29th ACM Symposium On Applied Computing (SAC 2014), Gyeongju, Korea, 2014.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**, 2000.

FLORAC, W. A.; CARLETON, A. D. **Measuring the Software Process: Statistical Process Control for Software Process Improvement**, Addison Wesley, 1999

FONSECA, V.S. **Uma Abordagem Baseada em Ontologias para Integração Semântica de Ferramentas de Apoio à Medição de Software**, 2015. Dissertação de Mestrado, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, 2015.

---

[16] *http://www.agilefant.com/*

GUARINO, N. **Formal Ontology and Information Systems**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE IN FORMAL ONTOLOGY AND INFORMATION SYSTEMS - FOIS'98. Trento, Itália, 1998, pp. 3- 15.

GUIZZARDI, G. **Ontological Foundations for Structural Conceptual Models**, 2005.

IEEE. **Std 1061 – IEEE Standard for a Software Quality Metrics Methodology.**1998.

ISO/IEC. **ISO/IEC 15504-2 - Information Technology – Software Process Assessment**. International Organization for Standardization and the International Electrotechnical Commission, Geneva, Suiça, 2013.

ISO/IEC. **ISO/IEC 15939 (E) Software Engineering – Software Measurement Process**. International Organization for Standardization and the International Electrotechnical Commission, Geneva, Suiça, 2007.

ISO/IEC. **ISO/IEC 12207:2008 - Systems and Software Engineering - Software Life Cycle Process**. International Organization for Standardization and the International Electrotechnical Commission, Geneva, Suiça, 2008.

MARETTO, C. X. **Uma Arquitetura de Referência para Medição de Software**, 2013. Dissertação de Mestrado, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, 2013.

MCGARRY, J.; CARD, D.; JONES, C.; LAYMAN, B.; CLARK, E.; DEAN, J.; HALL, F. **Practical Software Measurement: Objective information for decision makers**. Boston, Estados Unidos: Addison Wesley, 2002.

SEI. **CMMI® for Development, Version 1.3**. Pittsburg, 2010.

SOFTEX. **MPS.BR: Melhoria de Processo do Software Brasileiro – Guia Geral: 2009.** 2011.

SOFTEX. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software**. 2012.

TARHAN, A.; DEMIRÖRS, O. **Investigating Suitability of Software Process and Metrics for Statistical Process Control**. 2006, p. 88–99.