UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PAULO ROBERTO GARCIA JÚNIOR

APSEE-Metrics: um Modelo para Mensuração em Processos de *Software*

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Daltro José Nunes
Orientador

Porto Alegre, setembro de 2006

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Garcia Júnior, Paulo Roberto

APSEE-Metrics: um Modelo para Mensuração em Processos de *Software* / Paulo Roberto Garcia Júnior – Porto Alegre: Programa de Pós-Graduação em Computação, 2006.

165 f. : il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2006. Orientador: Daltro José Nunes

1. Processos de *software*. 2. Mensuração. 3. Métricas. 4. Ambientes de engenharia de *software* centrados no processo. I. Nunes, Daltro José. II. Título.

AGRADECIMENTOS

Meu agradecimento especial a Deus que me deu inspiração e coragem para superar todas as dificuldades neste período. À minha esposa Patrícia pela paciência e apoio em todos os momentos desta jornada, e por ter me ajudado a persistir na conclusão deste trabalho. Aos meus pais, familiares e amigos que me incentivaram e me apoiaram em todos os momentos.

Meu agradecimento especial ao meu orientador, Prof. Daltro Nunes, por ter acreditado no meu potencial e me ajudado com toda sua experiência e sabedoria, e também compreendido as dificuldades que tiveram de ser superadas neste período.

Agradeço também aos colegas do grupo Prosoft (Abraham Lincoln, Alessandra Dahmer, Antonio Terceiro, Ana Vitória, Anderson Baia, Luiza Pagliari, Marcos Henker), pelas contribuições a este trabalho e em todo o Mestrado. Em especial aos amigos Lincoln e Duda por terem me apoiado sempre de forma tão especial e me ajudado a chegar ao final desta jornada.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADM	Ambiente de desenvolvimento de *software*
ATO	Ambiente de Tratamento de Objetos
CASE	*Computer-Aided Software Engineering* – Engenharia de *Software* Auxiliada por Computador
CMMI	*Capability Maturity Model Integration* – Integração de Modelos de Maturidade da Capacidade
GQM	*Goal-Question-Metric*
GQ(i)M	*Goal-Question-Indicator-Metric*
HTML	*Hypertext Markup Language*
ICM	*Issue - Category - Measure*
ICS	Interface de Comunicação do Sistema
PHP	*Hypertext Preprocessor*
PML	*Process Modeling Language* - Linguagem de modelagem de processos
PSM	*Practical Software & System Measurement*
PSEE	*Process-centered Software Engineering Environment* – Ambiente de Engenharia de *Software* Centrado no Processo
RUP	*Rational Unified Process* – Processo Unificado Rational
SEI	*Software Engeneering Institute*
SGBD	Sistema Gerenciador de Banco de Dados
TAD	Tipo Abstrato de Dado
UML	*Unified Modelling Language* – Linguagem de Modelagem Unificada

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A avaliação e melhoria de um processo de *software* requer um grande esforço por parte de gerentes e desenvolvedores, uma vez que a obtenção de dados para tomada de decisão, acerca da evolução do processo nem sempre é trivial. O desenvolvimento de *software*, como qualquer disciplina de Engenharia, necessita de mecanismos de mensuração, para que através de uma avaliação de seus resultados o processo possa ser melhorado. Mas a mensuração pode ser uma tarefa bastante trabalhosa sem o auxílio de metodologias ou ferramentas de apoio. Neste sentido, este trabalho apresenta um modelo para mensuração em processos de *software*, no contexto do projeto APSEE-Prosoft. O objetivo do modelo proposto é prover suporte a realização de mensuração em processos de *software*, através do desenvolvimento e implantação de programas de mensuração, visando auxiliar aos gerentes na realização de atividades como planejamento, coleta e análise de métricas em um processo de *software*. Neste trabalho são combinados aspectos relacionados a importantes áreas da Engenharia de *Software*, como mensuração, processos de *software*, ambientes de engenharia de *software* centrados em processos (PSEE), bem como a utilização de métodos formais para a definição e especificação de um modelo de mensuração integrado a um PSEE.

Palavras-chave: Processos de *software*, Mensuração de *Software*, Métricas, Ambientes de engenharia de *software* centrados no processo.

APSEE-*Metrics* : *A Model for Software Process Measurement*

ABSTRACT

The software process evaluation and improvement requests great effort of managers and developers, especially because obtaining data for decision support (about process evolution) is not a simple task. The software development, as any discipline of Engineering, needs measurement mechanisms, so that through an evaluation of its results, the process can be improved. But measurement can be a difficult task without the aid of methodologies or support tools. In this sense, this paper presents a model for software processes measurement, on the context of APSEE-Prosoft project. The model's objective is to provide support for the measurement in software processes, through the development and implantation of measurement programs, to aid managers in the accomplishment of activities such as planning, collecting and analysis of metrics in a software process. This paper combines aspects related to important areas of Software Engineering, as measurement, software process, process-centered engineering environment (PSEE), besides the use of formal methods for a definition and specification of a measurement model integrated to a PSEE.

Keywords: *Software Process, Software Measurement, Metrics, Process-Centered Software Engineering Environment*.

1 INTRODUÇÃO

Aumentar a qualidade, performance e produtividade é o objetivo chave de qualquer organização que desenvolve *software* (MENS, 2001). Mas a obtenção de produtos de qualidade e o aumento da produtividade e performance não dependem somente de uma equipe técnica competente, mas também de pessoal especializado em questões gerenciais. Tradicionalmente, os profissionais de *software* dão extrema importância a questões técnicas, e algumas vezes acabam deixando um pouco de lado os aspectos gerenciais, tais como o controle sistemático dos projetos em andamento. Sem a sistematização do processo de desenvolvimento é muito difícil melhorar continuamente a qualidade do *software*, bem como deixar o resultado (o produto) mais previsível (ANACLETO, 2003). Para que possamos obter esta sistematização do processo de desenvolvimento, a utilização de técnicas, métodos e ferramentas da engenharia de *software* em um processo de desenvolvimento de *software* é um fator importante no estabelecimento do custo e da qualidade do produto.

Para que um projeto de desenvolvimento de *software* tenha sucesso, é necessário analisar alguns fatores durante todo o ciclo de vida do *software*, tais como o escopo do *software*, riscos, recursos, tarefas, custos, entre outros. Mas a análise destes parâmetros pode ser uma tarefa bastante trabalhosa sem o auxílio de metodologias ou ferramentas de apoio, uma vez que a coleta de dados sobre o processo de desenvolvimento, quase sempre requer um grande esforço por parte dos envolvidos. Além disto, muitas vezes pode não garantir a coleta das informações mais relevantes e consistentes para as análises, ou ainda, não proporcionar os resultados esperados (VAVASSORI, 2002).

Segundo Queiroz (1999), desafios como a utilização da engenharia de *software* como diferencial no desenvolvimento de *software* e o atendimento das constantes demandas dos usuários com qualidade e produtividade, tem levado os pesquisadores à busca de novas soluções em várias áreas de pesquisa, tais como: gerenciamento de projetos e de processos, modelos de qualidade, métodos formais para validação e verificação da qualidade, utilização de métricas, ferramentas, entre outras.

Entre estas várias áreas de pesquisas relacionadas, a literatura especializada aponta um crescente interesse sobre a utilização e desenvolvimento de métodos e técnicas para avaliação da qualidade e maturidade do processo de desenvolvimento de *software*, sendo que diversos especialistas da área, como por exemplo, Fuggetta (1998) e Mens (2001), consideram que um dos mais importantes fatores para

habilitar e suportar qualquer iniciativa de melhoria é a avaliação efetiva de métricas de *software*, as quais podem possibilitar elementos para a avaliação quantitativa do nível da qualidade dos produtos e do processo de desenvolvimento.

O uso de métricas está diretamente ligado à melhoria do processo de produção de *software*. A mensuração de *software* está se consolidando como uma prática importante entre as empresas, visando suportar suas iniciativas de melhoria no processo de *software*, pois pode servir tanto como fonte de informação para o monitoramento da situação atual do processo de *software*, como para a identificação de desvios na execução do processo (FRANÇA, 1999).

> Em qualquer campo científico, as medições fornecem descrições quantitativas dos processos e dos produtos, possibilitando a compreensão de comportamentos e de resultados. Este aumento de entendimento permite a melhor seleção de técnicas e ferramentas para controlar e melhorar os processos, produtos e recursos. Como a Engenharia envolve a análise de medições, a Engenharia de Software somente será uma verdadeira engenharia, quando estiver sedimentada numa sólida fundação de teorias de medição França (1998, p.4 apud PFLEEGER, 1993, p.2).

Diante da grande importância desta área para o desenvolvimento de melhores produtos e processos relacionados ao desenvolvimento de *software*, este trabalho visa abordar estudos referentes à área de mensuração em processos de *software*, de forma a contribuir ainda mais para o estado da arte das atuais pesquisas desta área.

Como veremos a seguir, são muitas as necessidades existentes e que merecem atenção de pesquisadores e profissionais da área na busca de melhores teorias e soluções para consolidar a mensuração de *software* no contexto das melhores práticas em desenvolvimento de *software*. A próxima seção apresenta alguns dos aspectos que motivaram o desenvolvimento deste trabalho.

1.1 Motivação

Nesta seção iremos discutir alguns pontos importantes, apresentados na literatura, acerca das dificuldades existentes na realização de mensuração em processos de desenvolvimento de *software*.

Como vimos na seção anterior, existe uma grande importância na coleta de informações a cerca de um processo de desenvolvimento de *software*. Para que isto ocorra, uma das necessidades principais é a obtenção de dados confiáveis e consistentes para tomada de decisão. Mas a simples coleta de dados normalmente não produz os resultados necessários em termos da melhoria dos processos, sem que tenhamos uma organização e metodologias para isto. Neste sentido, um questionamento importante a ser investigado seria: ***como gerenciar a coleta de dados relevantes acerca do processo de desenvolvimento de software de forma consistente e organizada?***

A literatura apresenta ainda alguns outros questionamentos que são de importante valia para uma reflexão sobre as atuais práticas nesta área, como por exemplo, os apresentados por Munson (2004) e Fuentes (1997):

- Medimos de forma adequada?

- Estamos medindo conceitos claros e bem definidos, ou simplesmente estamos medindo pseudométricas e conceitos incoerentes?
- Como obter dados de medições do processo de desenvolvimento de *software*?
- Como converter estes dados em informações que irão permitir o gerenciamento do processo de *software*?
- Como gerenciar todos estes dados?

Segundo Tanaka (1998), em muitos casos, os departamentos de desenvolvimento de *software* também enfrentam algumas dificuldades no uso de ferramentas e técnicas para análise e medições. Segundo ele, os principais problemas são relacionados a dificuldade de entender e utilizar os dados e resultados obtidos pelas ferramentas e técnicas de medição. Além disto, freqüentemente não existe tempo suficiente para avaliar ferramentas e preparar o ambiente necessário para aplicá-las na prática, bem como incorporar o processo de mensuração de forma intrínseca ao processo de desenvolvimento de *software*.

Borges (2003), ainda aponta que, devido à quantidade e complexidade dos fatores envolvidos no desenvolvimento de um *software*, a seleção das métricas a serem coletadas deve ser cuidadosamente estudada, planejada e fortemente fundamentada, pois interfere diretamente na qualidade e relevância dos resultados obtidos com a mensuração. Sem uma política bem definida para a definição desse subconjunto de métricas, a probabilidade de se escolher arbitrariamente as mais adequadas, é bastante difícil.

Adicionalmente, Basili (1994), um dos principais pesquisadores desta área, ainda aponta que um dos maiores problemas em projetos de *software*, é a falta de habilidade dos gerentes para buscar critérios para escolha de processos apropriados (modelos globais de processo, métodos e ferramentas para suportar estes processos), além das dificuldades para avaliação e melhoria dos processos.

Outra dificuldade refere-se a dificuldade em se definir e padronizar o processo de coleta e análise dos dados, sendo necessário criar e documentar procedimentos, sistematizar a coleta e análise, e estabelecer critérios claros de medição, evitando assim diferentes interpretações das medições pelas pessoas envolvidas, o que reduz sua confiabilidade.

Diante deste cenário, acredita-se que novas pesquisas que auxiliem nestes aspectos apresentados, no intuito de aprimorar ou identificar novas abordagens para a realização de mensuração em processos de *software*, constituem uma grande contribuição para uma área que tem despertado grande interesse tanto da indústria quanto da comunidade cientifica nos últimos anos, uma vez que cada vez mais, busca-se elevar os níveis de qualidade tanto do processo de desenvolvimento quanto do produto deste processo.

Neste contexto, este trabalho apresenta um estudo relacionado a mensuração em processos de *software*, procurando identificar e apresentar as principais abordagens desta área e viabilizando a definição de um modelo para implementação de programas de mensuração de processos de *software* em organizações desenvolvedoras de *software*, com o intuito de disponibilizar para a comunidade científica um referencial para novos estudos, além de sua aplicação prática em

contextos empresariais. Os objetivos mais específicos do trabalho proposto são discutidos na próxima seção, onde são apresentados os limites deste trabalho em relação à área de pesquisa a qual está inserida.

1.2 Objetivos

Nas seções anteriores, discutimos acerca dos aspectos relacionados à importância da aplicação da mensuração em processos de desenvolvimento de *software*, bem como algumas das principais dificuldades existentes nesta área. No intuito de contribuir para as pesquisas desta área, o presente trabalho visa apoiar algumas estratégias para mensuração, sendo descritos nesta seção de forma sintética, os objetivos deste trabalho.

De maneira geral, o objetivo principal deste trabalho é propor um modelo para mensuração em processos de *software*, proporcionando uma infra-estrutura de apoio para implementação de programas de mensuração em processos de *software*. Este modelo está relacionado com a evolução do projeto APSEE-Prosoft (NUNES, 1994), descrito em maiores detalhes nas seções seguintes. Para isso, são objetivos gerais deste trabalho:

- Identificar um conjunto de requisitos necessários para realização de mensuração em processos de *software*;
- Elaborar um modelo para mensuração em processos de *software*, com base nos requisitos identificados e em técnicas consagradas na literatura e na indústria;
- Especificação do modelo para suporte a mensuração em processos de *software* em ambiente de engenharia de software centrado em processos (PSEE);
- Verificar a viabilidade do modelo proposto através de um protótipo, isto é, uma implementação limitada que forneça as funcionalidades básicas descritas pelo modelo; e
- Mostrar um exemplo de utilização do modelo, usando um cenário real.

1.3 Abordagem utilizada

O presente trabalho faz parte de uma atividade de pesquisa do Projeto PROSOFT, referente à tecnologia de processos de *software* estudadas no desenvolvimento do ambiente de automação e gerência de processos APSEE. A seguir, é apresentado um breve histórico do projeto, segundo Souza (2004).

O projeto PROSOFT iniciou suas atividades há mais de dez anos, envolvendo vários estudantes de graduação e pós-graduação, pesquisadores e professores do Instituto de Informática da UFRGS (Universidade Federal do Rio Grande do Sul), assim como instituições coligadas no país e exterior, visando a construção de um ambiente integrado que vem servindo de laboratório para a experimentação de

tecnologias que apoiam o desenvolvimento de *software* com alta qualidade e produtividade.

O PROSOFT, enquanto ambiente para construção de *software*, visa auxiliar o desenvolvimento de *software* desde as etapas iniciais (concepção e especificação de requisitos do produto) até a implementação do sistema, seguindo rigorosos padrões de qualidade.

A utilização de métodos formais é enfatizada na construção das ferramentas do ambiente a partir de um paradigma próprio, criado pelo próprio projeto. Sob este paradigma, ferramentas CASE (*Computer Aided Software Engineering*, ou Engenharia de *Software* Apoiada por Computador) foram desenvolvidas para apoiar a utilização dos métodos formais e semi-formais para a construção de *software*, sendo então integradas ao ambiente. A popularização da Internet e o crescente aperfeiçoamento e uso das redes de alta velocidade influenciou decisivamente a evolução do ambiente.

Desse modo, nos últimos anos, diversos aperfeiçoamentos foram especificados sob o paradigma do PROSOFT, a maioria desses diretamente relacionados à evolução do ambiente para apoiar o envolvimento cooperativo e distribuído de profissionais que atuam no desenvolvimento de *software*, assim como a automação da gerência do processo de desenvolvimento de *software* (REIS, 1998).

Atualmente, sua arquitetura evoluiu para uma estrutura integrada de gerência de processos de *software* que apresenta o meta-modelo a ser utilizado por todas as ferramentas do ambiente. Este aperfeiçoamento é chamado de APSEE, no qual o presente trabalho está inserido.

Assim, este trabalho contempla o desenvolvimento de um modelo para mensuração em processos de software no ambiente APSEE, com a finalidade de proporcionar a infra-estrutura necessária para implementação de programas de mensuração em organizações desenvolvedoras de software, não sendo o objetivo do mesmo a proposição de novas métricas.

1.4 Organização do texto

O presente trabalho tem seu texto está organizado da seguinte forma. O **capítulo 2** discute os aspectos relativos a mensuração de *software*, abrangendo os principais conceitos relativos ao domínio do estudo, as abordagens existentes, bem como um conjunto de requisitos para mensuração em processos de *software*.

O **capítulo 3** apresenta uma visão geral de alguns trabalhos relacionados, presentes na literatura.

O **capítulo 4** apresenta o modelo proposta para mensuração em processos de *software*.

O **capítulo 5** apresenta a especificação algébrica do modelo.

O **capítulo 6** apresenta o protótipo de uma ferramenta desenvolvida para apoiar o processo de mensuração, integrada ao ambiente PROSOFT-APSEE.

O **capítulo 7** ilustra um exemplo de utilização do modelo em um cenário real.

Finalmente, o **capítulo 8** apresenta as considerações finais, conclusões e limitações do trabalho e relaciona os trabalhos futuros vislumbrados a partir do presente trabalho.

2 MENSURAÇÃO EM PROCESSOS DE *SOFTWARE*

Este capítulo apresenta uma breve introdução sobre os principais conceitos apresentados na literatura, relativos a mensuração em processos de *software*, de forma a contextualizar o domínio do estudo realizado.

2.1 Conceitos e objetivos

Medir é fundamental em qualquer disciplina de engenharia, e a engenharia de *software* não é exceção. As métricas corretas, usadas da maneira certa, são absolutamente essenciais para o sucesso de um projeto. Cada métrica precisa ter um propósito, provendo suporte a um processo de tomada de decisão (PERKINS, 2003).

A obtenção de métricas e medidas é realizada através de atividades de medição (ou mensuração). Uma definição de medição ou mensuração é dada por (FENTON, 1994) onde ele define como sendo o processo pelo qual números ou símbolos são associados a atributos de entidades no mundo real, com o objetivo de descrevê-la de acordo com um conjunto de regras claramente definidas.

A medição de *software* está diretamente ligada à garantia de qualidade de *software*. Ao longo do desenvolvimento do *software*, três pontos devem ser focos da medição: produto, processo e recurso (FRANÇA, 1998).

São vários os benefícios obtidos através da mensuração de *software*. No contexto de um projeto de desenvolvimento, dados quantitativos podem fornecer informações precisas quanto às características do produto em desenvolvimento, além de possibilitar mecanismos adequados de acompanhamento e controle da evolução do projeto. Na melhoria dos processos, a mensuração permite que se tenha um melhor conhecimento da eficácia do processo de desenvolvimento, através do acompanhamento quantitativo de diversos aspectos como produtividade, qualidade dos produtos desenvolvidos, acurácia de estimativas, entre outros (BORGES, 2004).

Em termos de planejamento e estimativa, cujo interesse é histórico, a mensuração pode nos ajudar a analisar responder questionamentos comuns entre os gerentes, tais como os apresentados por Pressman (2006):

- Qual era a produtividade no desenvolvimento de *software* de projetos passados?
- Qual a qualidade do *software* que era produzido?

- Como os dados passados de produtividade e qualidade podem ser extrapolados para o presente?
- Como isso pode ajudar-nos a planejar e a estimar com maior precisão?

As métricas obtidas com a mensuração podem nos ajudar a entender o processo técnico usado para se desenvolver um produto, como também o próprio produto. O processo é medido, num esforço para melhorá-lo. O produto é medido, num esforço para aumentar sua qualidade. A mensuração capacita-nos a quantificar e, consequentemente, a administrar mais efetivamente (PRESSMAN, 2006).

Informações de diferentes aspectos do desenvolvimento pode nos ajudar a determinar o progresso em direção aos objetivos do projeto. A mensuração pode ser usada extensivamente como ferramenta para prover base para tomada de decisão (PERKINS, 2003). Além disto, umas das razões chave de se utilizar métricas no desenvolvimento de *software* é a previsibilidade. O processo de se tentar prever problemas, ou simplesmente permitir gerenciar o processo de desenvolvimento de *software* em todas as fases é fundamental.

Segundo Pressman (2006), outras razões importantes para a realização de mensuração são:

- indicar a qualidade do produto;
- avaliar a produtividade das pessoas que produzem o produto;
- avaliar o benefícios (em termos de produtividade e qualidade) derivados de novos métodos e ferramentas de *software*;
- formar uma linha básica para estimativas;
- ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional.

Tendo em vista estes conceitos em relação a mensuração, é fundamental que possamos explorar alguns dos principais tipos de medição existentes, visando contextualizar ainda mais a abrangência da mensuração em termos práticos. Para tanto, a seção seguinte apresenta alguns dos principais tipos de medição existentes.

2.2 Tipos de medição

A literatura corrente enumera uma extensa lista de métricas que podem ser utilizadas na mensuração de um processo de *software*. Essas métricas podem ser classificadas quanto às propriedades ou artefatos que estão medindo (produto, processo, recurso), ou quanto à forma da medição (direta ou indireta) (PRESSMAN, 2006).

Entre as medidas diretas do processo de *software* incluem-se, por exemplo, o custo e o esforço aplicados. As medidas diretas do produto incluem-se, por exemplo, as linhas de código (LOC) produzidas, velocidade de execução, tamanho de memória e defeitos registrados ao longo de certo espaço de tempo. As medidas indiretas do produto são, por exemplo, funcionalidade, qualidade, complexidade, eficiência, confiabilidade, manutenabilidade e muitas outras.

O custo e o esforço exigidos para se construir o *software*, o número de linhas de código produzido e outras medidas diretas, são relativamente fáceis de ser reunidas, desde que convenções específicas para medição sejam estabelecidas antecipadamente. Porém, a qualidade e a funcionalidade do *software*, ou sua eficiência e capacidade de manutenção, são mais difíceis de serem avaliadas e somente podem ser medidas indiretamente.

Dado a grande diversidade existente a grande quantidade de métricas existentes, um conjunto de categorias relativas ao domínio de métricas de *software* é apresentado por (PRESSMAN, 2006):

- Métricas da produtividade: concentram-se na saída do processo de engenharia de *software*;
- Métricas da qualidade: oferecem uma indicação de quão estreitamente o *software* conforma-se às exigências implícitas e explícitas do cliente (adequação ao uso do *software*);
- Métricas técnicas: concentram-se na característica do *software* (por exemplo, complexidade lógica, grau de modularidade);

Uma outra divisão usada pelo autor para compilar as medições diretas da saída e da qualidade da engenharia de *software*, é apresentada a seguir:

- Métricas orientadas ao tamanho: são usadas para compilar as medições diretas da saída e da qualidade de engenharia de *software*;
- Métricas orientadas para a função: oferecem medidas indiretas;
- Métricas orientadas às pessoas: compilam informações sobre a maneira segundo a qual as pessoas desenvolvem *software* e percepções humanas sobre a efetividade das ferramentas e métodos.

Assim, algumas métricas frequentemente referidas e relativas ao acompanhamento dos projetos de desenvolvimento podem incluir:

- cronogramas de projeto;
- custos de projeto;
- esforço para execução do projeto;
- progresso do trabalho;
- características do projeto (tamanho, complexidade, natureza, custo, etc.);
- características do produto (tamanho, complexidade, natureza, etc.);
- medições variadas, selecionadas ou definidas pelos gerentes de *software*;
- falhas, defeitos e faltas encontradas, catalogadas por sua natureza;
- entre outras.

Mas esta grande variedade de métricas, se de um lado permite que vários aspectos do desenvolvimento de *software* sejam mensurados, por outro pode gerar a falta de objetividade e, por conseguinte, o fracasso das iniciativas de mensuração em uma empresa (FRANÇA, 1998). Desta forma, a necessidade de um processo

organizado e bem definido para a mensuração é fundamental. Na seção seguinte veremos a importância da realização de um programa de mensuração, a qual pode nos auxiliar em cada uma das etapas da mensuração.

2.3 Programas de mensuração

As empresas necessitam de dados para tomada de decisão. E estes dados, são freqüentemente baseados em resultados de medições, através de algum instrumento ou meios de medição (CANOSSA, 2005).

Mas no desenvolvimento de *software*, a maioria dos desenvolvedores de *software* não realiza medições, e infelizmente, a maioria não tem muita vontade de começar a fazê-lo. O problema é cultural. Tentar compilar medidas quando ninguém as compilava no passado freqüentemente provoca resistências (PRESSMAN, 2006). Além disto, a dificuldade existente na condução adequada das atividades de mensuração requer um gerenciamento adequado destas práticas, bem como o uso de metodologias e técnicas apropriadas par que estas medições sejam realmente efetivas.

Alguns autores como (PERKINS, 2003), (BASILI, 1994) entre outros, enfatizam que a utilização de métricas por si só, não apresentam resultados significativos, e defendem a criação de um programa de mensuração, ou seja, uma seqüência de ações coordenadas para realizar as medições em processos de *software*. Além disto, este tipo de programa deve ser baseado nos objetivos da organização e cuidadosamente planejado, implementado e regularmente avaliado para ser efetivo.

No entanto, a implementação de um programa de mensuração não constitui uma tarefa simples. Quando uma organização opta por iniciar um programa de mensuração, várias providências devem ser tomadas: as medidas a serem coletadas devem ser selecionadas, procedimentos de medição devem ser padronizados e documentados, a forma de armazenamento das medidas em uma base histórica precisa ser definida, e diretrizes para a análise desses dados devem ser estabelecidas (BORGES, 2004).

Além disto, um programa de mensuração envolve a geração de um grande volume de dados, que somente pode ser manipulado e analisado de forma eficaz, se devidamente automatizado. Através da mensuração, os laboratórios de Engenharia de *Software* poderão obter dados quantitativos para contrapor ou confirmar os supostos benefícios advogados sem a devida comprovação experimental (FRANÇA, 1999). Na seção 2.6 serão apresentadas algumas das principais abordagens existentes para realização de programas de mensuração, onde serão descritas em maiores detalhes as principais etapas existentes neste tipo de programa.

Nas seções seguintes serão apresentados os principais benefícios e dificuldades existentes com a realização de mensuração de *software*.

2.4 Benefícios com a mensuração

A utilização de métricas pode prover importante suporte ao planejamento, controle, monitoramento e avaliação do processo de *software*. A necessidade de utilização de métricas que permitam fornecer indicadores de esforço de desenvolvimento e também o acompanhamento das atividades inerentes aos projetos de desenvolvimento, bem como seus níveis de qualidade, tem sido apontada de forma bastante intensa na literatura por diversos autores como Abreu (1992), Basili (2002), Borges (2003), Fuggetta (1998), Perkins (2003), Tanaka (1998), Vavassori (2001) e muitos outros.

Neste sentido, inúmeras pesquisas tem sido realizadas à respeito, as quais apontam o uso de medições e métricas de *software* como fundamental para a avaliação de diferentes aspectos relativos as fases do ciclo de vida do *software*, visando a otimização de esforços e custos para manter a qualidade do produto e a competitividade no mercado. Sua utilização também pode servir também como base para identificação de forças e fraquezas em um processo, além de permitir a visualização de oportunidades de melhoria, auxiliando também na realização de análises inerentes a gerência de projeto.

Segundo Fuggetta (1998), métricas confiáveis provêem a evidência da melhoria, permitindo análises de custo-benefício e provendo base para tomada de decisão.

Os resultados diretos e/ou indiretos habitualmente referidos na literatura sobre a aplicação de um programa de mensuração em processos de *software* são, entre outros:

- melhor monitoramento dos projetos e do processo de desenvolvimento;
- um controle mais efetivo da qualidade do processo e do produto;
- identificação mais precoce de problemas, permitindo ações corretivas em tempo útil, com menor impacto nos prazos globais dos projetos;
- melhoria da compreensão da atividade de gestão, uma vez que a obtenção de dados quantitativos pode diminuir um pouco a imprevisibilidade associada ao desenvolvimento de *software*;
- melhoria da imagem da entidade, dado que a confiança dos clientes aumenta ao sentirem uma preocupação (e ação) no sentido de melhorar a qualidade dos produtos e serviços fornecidos;
- maior produtividade;
- melhor identificação das fases do ciclo de vida, definidas no processo de desenvolvimento adotado, onde um maior investimento em melhorias provocará uma maior rentabilidade;
- melhoria das estimativas e planeamento dos projetos;
- avaliação da eficácia e eficiência da introdução de novas ferramentas;
- avaliação da influência da linguagem e ambiente utilizado na produtividade e qualidade dos sistemas desenvolvidos;

- maior facilidade dos gestores de projetos na comprovação, perante a direção, da melhoria da proficiência conseguida;

Mas apesar destes inúmeros benefícios, a realização de medições em processos de desenvolvimento de *software*, não constitui uma tarefa trivial. Na próxima seção iremos discutir alguns aspectos importantes relativos as principais dificuldades existentes com relação a isto.

2.5 Dificuldades na mensuração de processos de *software*

Segundo França (1998), de maneira geral a introdução de um programa de mensuração nas organizações é marcado por dificuldades, inicialmente na definição do programa e, posteriormente em sua execução.

Em relação a definição, muitas vezes, a empresa não consegue identificar as métricas mais relevantes para sua realidade. Para a execução do programa, o suporte computacional é fundamental, pois a medição de *software* envolve a geração e a manipulação de um grande volume de dados. Contudo, existe muita dificuldade de se conseguir ferramentas adequadas para suportar o programa de mensuração requerido pela organização e muitas vezes o programa acaba sendo definido para adequar-se a capacidade das ferramentas selecionadas.

Em termos gerais, a iniciativa de implementar um programa de mensuração eficiente e eficaz, depara-se muitas vezes com problemas do tipo:

- Da extensa lista de métricas, quais são as adequadas para o meu caso?
- Quais são as ferramentas e os processos necessários para realizar as medições desejadas?
- Quais são as ferramentas e os processos disponíveis?
- Quais são os recursos financeiros e humanos necessários para implementar e manter o programa de medição?
- Como analisar os dados coletados?
- Como apresentar as informações relacionadas aos dados coletados?
- Como atuar sobre ferramentas, métodos e processos no sentido de automatizar a medição?

Além destes problemas, alguns questionamentos importantes se fazem presentes em iniciativas de mensuração, como as apresentadas em (PRESSMAN, 2006).

- Quais são as métricas apropriadas para o processo e para o produto?
- Como os dados que são compilados podem ser utilizados?
- É justo utilizar medições para se comparar pessoas, processos e produtos?

Essas perguntas e muitas outras sempre vêm à tona quando é feita uma tentativa de se medir alguma coisa que não tenha sido medida no passado.

Para suprirem muitas destas dificuldades, algumas abordagens existentes procuram definir conceitos e metodologias para orientar a aplicação de mensuração de *software*. Na seção seguinte procuramos apresentar uma breve introdução às principais abordagens existentes para mensuração, que visam justamente minimizar estes problemas e orientar a correta realização destas atividades.

2.6 Abordagens para mensuração de *software*

Esta seção apresenta algumas das abordagens existentes na literatura especializada da área relativas a realização de mensuração de *software*, onde é apresentado uma visão geral e as principais características de três das principais abordagens existentes, o *Goal-Question-Metric* (BASILI, 1994), *Goal-driven Software Measurement* ou GQ(i)M (PARK, 1996) e o PSM - *Practical Software & System Measurement* (USA, 2003).

2.6.1 Goal-Question-Metric (GQM)

Uma das principais abordagens existente e considerada como grande referencial para várias pesquisas da área é apresentada nesta sub-seção. Trata-se do GQM, desenvolvido por Victor Basili (BASILI, 1994).

O Goal/Question/Metric (GQM) é um paradigma para sistematizar a definição, estabelecimento e exploração de programas de medição, suportando a avaliação quantitativa tanto de produtos quanto de processos de *software* (FUGGETTA, 1998). O principal propósito do GQM é servir como suporte na elaboração e implementação de programas de mensuração, definidos para avaliar aspectos de qualidade para processos e produtos de *software* (ABIB, 1999).

GQM foi proposto por Basili para a caracterização e avaliação de defeitos em projetos desenvolvidos por um dos laboratórios de Engenharia de *Software* da NASA (*Goddard Space Flight Center*). Posteriormente, o uso do GQM foi expandido e tem sido adotado para medir e melhorar a qualidade em organizações de desenvolvimento de *software*.

A abordagem GQM parte do princípio que para uma organização medir de maneira eficiente, ela precisa antes identificar quais objetivos devem ser alcançados para ela e seus projetos. Com os objetivos identificados, ela precisa então associá-los com os dados que definem esses objetivos operacionalmente e finalmente prover uma estrutura para interpretação desses dados em relação aos objetivos. Como resultado da aplicação do GQM, temos a especificação de um sistema de medição objetivando um conjunto particular de questões pertinentes e um conjunto de regras de interpretação para os dados medidos (ANDRADE, 2003).

O ponto de partida do GQM consiste em estabelecer-se uma quantidade de objetivos de melhoria e refinar estes objetivos em um número de questões e métricas, conforme ilustrado pela figura 2.1. Objetivos são definidos em termos de propósitos, perspectivas e ambientes, e são subseqüentemente refinados em um conjunto de questões que definem estes objetivos. Elas devem satisfazer a noção intuitiva de que os objetivos são completos e consistentes. As questões por sua

vez, são refinadas em métricas, que por sua vez refletem os dados necessários para responder as questões.

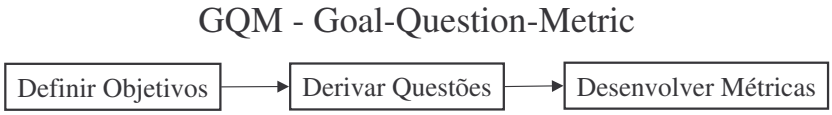


Figura 2.1: Paradigma GQM

De maneira mais específica, apresentamos à seguir uma breve descrição dos principais conceitos chave do GQM, segundo Perkins (2003):

- Os processos (incluindo desenvolvimento de *software*, programa de gerenciamento, etc.), têm seus alvos associados;
- Cada alvo indica uma ou mais questões que correspondem ao atingimento do alvo;
- Cada questão indica um ou mais métricas necessárias para responder a estas questões;
- Cada métrica requer duas ou mais medidas para produzir a métrica (ex.: milhas por hora, despesas gastas x despesas planejadas, temperatura x limites operacionais, tempo de execução atual x estimado, etc.);
- Medidas são selecionadas para proverem dados que irão produzir a métrica de forma precisa.

O GQM é composto por estágios, cada qual composto de fases que especificam atividades, de acordo com a figura 2.2.

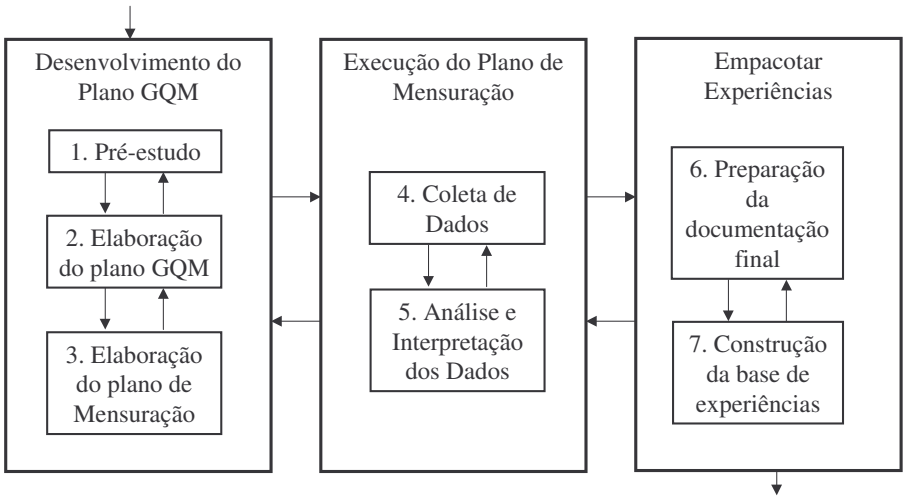


Figura 2.2: Estágios e fases da abordagem GQM

Os estágios e fases do GQM podem ser resumidos em:

a) Desenvolvimento do plano GQM: o objetivo deste estágio é detalhar as intenções do plano de medição. Esta atividade é atingida pelas seguintes fases:

- Pré-estudo: esta fase ajuda a caracterizar o contexto completo para o problema considerado, obtendo a informação necessária para implementar o programa de medição.

- Elaboração do plano GQM: o propósito desta fase é definir os alvos, questões, e métricas, para o programa de medição. Para isto, os passos a serem seguidos são

  ⇒ *Identificação dos alvos de Medição*, onde os alvos do plano de medição são identificados. Um alvo é composto por *objeto, propósito, foco da qualidade, ponto de vista e ambiente*.

  ⇒ *Produção do plano GQM*, onde as questões e métricas que irão guiar o plano de medição são definidas.

- *Elaboração do plano de medição:* nesta fase, o plano GQM e os alvos de medição são analisados, ajudando a definir em detalhes as estratégias e técnicas que irão fazer parte da implementação do programa de medição. Esta fase compreende as seguintes atividades:

  ⇒ *Preparar a Abstraction Sheet:* consiste na construção de um documento onde os alvos de medição são colocados no cabeçalho e o foco de qualidade, os fatores de variação, a hipótese base, e os impactos da hipótese são colocados nos quadrantes.

  ⇒ *Definir as questões*: inclui a definição de questões compatíveis com os alvos e com a hipótese base, e concentra no foco de qualidade e os fatores de variação;

  ⇒ *Definir as métricas*: As métricas ajudam a quantificar as questões formuladas. Cada métrica precisa ser relacionada com pelo menos uma questão, e cada questão, a pelo menos uma métrica.

b) Execução do plano de medição: o objetivo deste estágio é implementar a intenção do plano de medição. Envolve principalmente a coleta de dados, onde os dados são coletados e validados, e a análise de dados e interpretação dos dados coletados.

c) Empacotar experiências: o objetivo deste estágio é preparar os resultados obtidos pelas medições, de forma que eles possam ser de eficiente utilização. Este estágio é dividido em duas fases, que são:

- A *preparação da documentação final*: onde o programa de mensuração é completamente documentado, incluindo documentação de todos os materiais produzidos durante o processo de avaliação,

- A *construção de uma base de experiências*: onde são armazenadas as experiências obtidas com o programa de medição, ajudando no reuso do conhecimento em projetos de *software* futuros e em programas de avaliação.

O GQM é uma das principais abordagens existentes para mensuração e tem sido amplamente utilizada e frequentemente referenciada em diversos trabalhos da área. Neste trabalho apresentamos uma visão geral da abordagem, sendo que maiores detalhes sobre a abordagem podem ser encontrados em (BASILI, 1994).

Devido a grande importância desta abordagem, várias iniciativas em relação a sua adaptação e evolução tem sido desenvolvidas por diversos pesquisadores. Uma

destas evoluções é o *Goal-driven Software Measurement*, proposto pelo SEI, sendo apresentado de forma resumida na seção seguinte.

2.6.2 Goal-driven *Software* Measurement

Nesta seção apresentaremos o *Goal-Driven Software Measurement,* uma extensão do paradigma GQM proposta em 1996 pelo SEI (*Software Engineering Institute*), na Universidade de Carnegie Mellon (PARK, 1996). O *Goal-Driven Software Measurement* utiliza os princípios propostos pelo GQM, propondo um processo para a definição de uma política de mensuração, detalhando passo a passo uma seqüência de atividades que produz ao final um conjunto de medidas adequado às necessidades de uma organização.

Além de descrever mais detalhadamente os passos a serem seguidos, o *Goal-Driven Software Measurement* traz duas importantes contribuições em relação à proposta do GQM (BORGES, 2003):

Ao invés de partir das metas específicas de medição, o processo toma como base as metas de negócio da organização, que são mais abrangentes e geralmente mais conhecidas. A seqüência de passos apresentada descreve como detalhar as metas de negócio até obter as metas específicas de medição, a partir de quando os princípios do GQM podem ser seguidos.

No GQM, as medidas necessárias são deduzidas diretamente a partir das perguntas. No *Goal-Driven Software Measurement* são introduzidos os indicadores, que constituem um nível intermediário entre perguntas e métricas para auxiliar a identificação das métricas mais adequadas. Essa versão adaptada do GQM é freqüentemente denominada GQ(I)M – *Goal-Question-Indicator-Metric*.

A estrutura do processo é composta de dez passos, que devem ser seguidos de maneira ordenada, sendo eles:

1. Identificação de metas de negócio;
2. Identificação do que se deseja aprender;
3. Identificação de submetas, que refinam as metas de negócio;
4. Identificação de entidades e atributos envolvidos;
5. Formalização de metas de medição;
6. Identificação de questões quantitativas e indicadores relacionados às metas de medição;
7. Identificação de elementos de dados a serem coletados para a construção dos indicadores apontados;
8. Definição e padronização das medições a serem realizadas;
9. Identificação de ações necessárias para a implementação do processo de medição;
10. Preparação de um plano para a implementação do processo.

Os quatro primeiros passos têm como objetivo gerar uma estrutura de informações que servirá como base para a definição das metas de medição. Tendo

sido definidas essas metas, os princípios do GQM são aplicados, e por fim os três últimos passos procuram tornar as medidas obtidas aplicáveis operacionalmente, através da padronização do processo de coleta e do planejamento da implantação desse processo na organização.

A organização dos sete primeiros passos pode ser vista na figura 2.3, onde são demonstrados os resultados obtidos em cada passo (o número dos passos está indicado entre parênteses), e como esses resultados formam insumos para os passos seguintes.

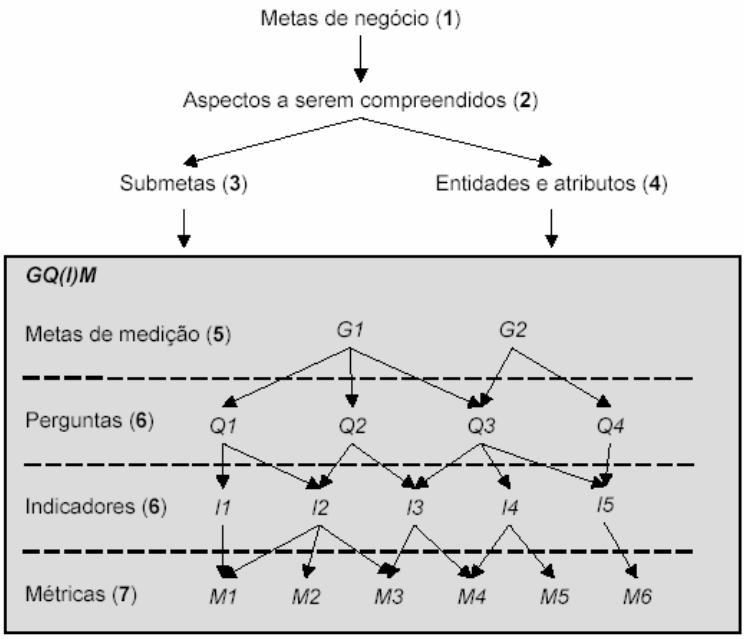


Figura 2.3: *Goal-driven Software Measurement* (BORGES, 2003)

As metas de negócio representam objetivos estratégicos da organização, e não precisam estar diretamente relacionadas a aspectos mensuráveis do processo. Um exemplo de meta de negócio seria o aumento da satisfação do usuário quanto aos produtos desenvolvidos. Para cada meta, é possível enumerar uma série de aspectos que devem ser compreendidos, verificados, previstos ou melhorados para que ela possa ser alcançada. Para a meta de aumento da satisfação dos usuários, por exemplo, os seguintes aspectos poderiam ser elicitados:

- A documentação fornecida ao usuário é suficientemente clara?
- Os produtos possuem níveis aceitáveis de defeitos?
- O tempo gasto para a correção de um defeito detectado atende às expectativas dos usuários?
- Os produtos são entregues em dia?
- A interface homem-máquina oferecida pelos produtos é satisfatória?

Assim como esses, vários outros aspectos poderiam ser listados. Agrupando-os em temas é possível desdobrar a meta de negócio em outras mais específicas, chamadas de submetas.

Para a meta exemplificada, possíveis submetas seriam:

- Melhorar a legibilidade dos documentos;
- Melhorar a confiabilidade e o desempenho do produto final;
- Melhorar o acompanhamento do progresso em relação aos compromissos assumidos;
- Melhorar a eficácia do processo de manutenção;
- Melhorar a comunicação com o usuário.

Após a definição das submetas, o passo seguinte consiste na identificação de entidades e atributos envolvidos no processo que estará sendo medido. Para cada entidade (código-fonte, manual de usuário, defeitos, processo de manutenção, etc) define-se um conjunto de atributos mensuráveis, como tamanho, distribuição, freqüência, tempo e esforço.

A lista de submetas obtidas para cada meta de negócio e o conjunto de entidades enumeradas com seus respectivos atributos forma a base para a aplicação do paradigma GQM, que ocorre a partir do quinto passo do processo. A diferença entre o GQM utilizado aqui – chamado também de *Goal-Question-Indicator-Metric* ou simplesmente GQ(I)M – e o GQM padrão, é a utilização dos indicadores. Um indicador é uma representação de uma combinação de medidas em um formato específico (geralmente listas, tabelas ou gráficos), que se construído poderia fornecer informações úteis para responder às perguntas derivadas das metas de medição.

Aplicando esse princípio a todas as metas de medição especificadas, obtém-se um conjunto de medidas alinhado com as metas de negócio da organização. Resta então que os procedimentos de coleta sejam padronizados e que sua implantação no processo de desenvolvimento utilizado pela organização seja planejada.

Nesta seção apresentamos uma breve introdução ao GQ(I)M, o qual podemos obter maiores detalhes de suas características e sua implementação em (PARK, 1996). Na seção seguinte iremos apresentar o PSM, uma das principais abordagens existentes para mensuração de *software*.

2.6.3 Practical Software Measurement

Nesta seção iremos apresentar o *Practical Software Measurement* (USA, 2003), um projeto patrocinado pelo Departamento de Defesa e forças armadas norte-americanas que tem como objetivo estabelecer um conjunto de práticas, ferramentas e serviços para auxiliar os gerentes de projetos a obter informações objetivas sobre os projetos em andamento, para que estes atinjam suas metas de prazo, custo e qualidade.

Em linhas gerais, o PSM descreve como definir e implementar um programa de mensuração para suportar as informações necessárias relativas ao desenvolvimento de *software* e suprir as organizações com informações acerca do processo de desenvolvimento. Seu desenvolvimento foi sustentado com a intenção de ser utilizado por:

- *Gerentes técnicos e de projetos*: para obter melhor entendimento do uso da mensuração para gerenciamento de seus projetos de *software*.
- *Equipe técnica de projeto*: para ajudar a implementar medições em um ambiente de projeto;
- *Empresários:* para entender os requisitos associados com a implementação da mensuração em suas organizações.

O guia é baseado na experiência atual do governo e de projetos industriais e representa as melhores práticas em mensuração utilizadas por profissionais da área e comunidades desenvolvedoras de *software*. A iniciativa conta com a participação de órgãos do governo, universidades e empresas privadas, e sua estratégia é reunir as práticas de maior sucesso na área de mensuração de *software* para compor um processo único, que possa ser utilizado como guia para a implantação de uma política de mensuração em organizações desenvolvedoras de *software*.

O foco principal do PSM é o processo de mensuração, mas o mesmo também discute aspectos chave em relação aos riscos e gerenciamento financeiro do projeto. Assim, as quatro principais atividades na qual o PSM se concentra são as seguintes:

- *Adaptação* das medidas de *software* para projetos específicos;
- *Aplicação* de medidas de *software* para converter dados mensuráveis em informações usáveis;
- *Implementação* de um processo de mensuração;
- *Avaliação* do programa de mensuração.

As iniciativas tomadas pelo projeto resultaram em dois produtos principais, as diretrizes e recomendações técnicas para o processo de mensuração, reunidas sob a forma de um livro (USA, 2003) e uma ferramenta de apoio ao processo, denominada PSM *Insight* (PSM, 2005).

A primeira versão do PSM foi publicada em 1997, e atualmente encontra-se em sua quarta versão. Em 2001, os princípios do PSM foram formalizados em um padrão, o ISO/IEC 15939, que estabeleceu algumas convenções terminológicas (BORGES, 2003).

O PSM considera que as atividades de mensuração de *software*, para obterem sucesso, devem possuir duas características, um alinhamento direto das atividades de coleta, análise e divulgação de dados medidos com as necessidades de informação dos responsáveis pela tomada de decisões nos projetos (nesse ponto, os princípios do PSM se assemelham aos do GQM, ressaltando a importância de se estabelecer de antemão os objetivos que se pretende atingir com a mensuração) e a existência de um processo de mensuração estruturado e bem documentado, que defina com precisão as atividades de medição (BORGES, 2003).

Alguns princípios defendidos no PSM são os seguintes (USA, 2003);

- Usar necessidades e objetivos para direcionar os requisitos de mensuração;
- Definir e coletar medidas baseadas em um processo técnico e gerencial;

- Coletar e analisar dados em um nível de detalhe suficiente para identificar e isolar problemas;
- Implementar uma capacidade independente de análise
- Usar um processo sistemático de análise para traçar as medidas às decisões
- Interprete o resultado das medições no contexto de outras informações do projeto
- Integrar a mensuração ao processo de gerenciamento do projeto durante todo o ciclo de vida do *software*
- Usar o processo de mensuração como base para comunicações objetivas
- Focar inicialmente no projeto - nível de análise

Fundamentado nestes princípios apresentados anteriormente, o PSM define um modelo de processo que é baseado em quatro atividades principais, que segundo o PSM são essenciais para o sucesso da implementação de qualquer política de medição. A figura 2.4 apresenta este processo, e exibe como essas atividades são combinadas para prover a implementação do *framework* a um determinado projeto.

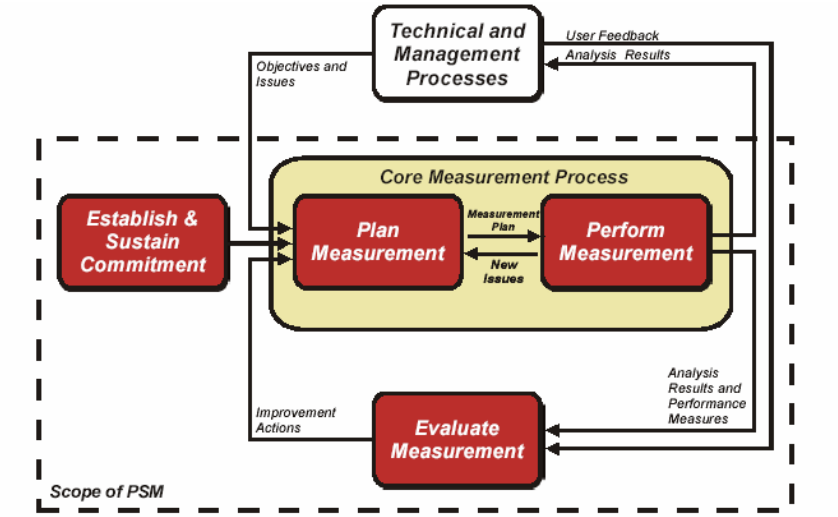


Figura 2.4: Atividades do PSM (USA, 2003)

O processo inclui quatro atividades primárias, cada qual com sua importância para o sucesso da implementação da mensuração. Estas atividades são resumidas como segue:

Planejar mensuração: envolve a identificação das necessidades de informação de um projeto e a seleção das medidas mais adequadas para atender a essas necessidades. Também inclui tarefas de definição dos procedimentos de coleta, análise e divulgação dos dados, tarefas relacionadas ao planejamento das avaliações dos resultados das medições, e tarefas de avaliação do processo de mensuração em si. De forma mais significativa, estas atividades provêem a integração das mensurações ao processo técnico e processo de gerenciamento existentes na organização. A saída da atividade de planejamento é uma abordagem bem definida das medições que irão suportar as necessidades de informação de cada projeto.

Executar mensuração: consiste na execução das atividades de coleta e processamento dos dados medidos, usando os dados para analisar tanto a informação individual necessária como sua relação com outros aspectos. A geração da informação produz base para a análise de resultados, possibilitando a obtenção de cursos alternativos do projeto e recomendações para os tomadores de decisão do projeto.

Avaliar mensuração: consiste em aplicar técnicas de medição e análise para avaliar o processo de mensuração em si. O objetivo dessa atividade é identificar possibilidades de melhoria tanto nas medidas aplicadas quanto ao processo de mensuração, para garantir que ele seja continuamente atualizado para suprir novas necessidades de informação, promovendo um crescente amadurecimento da organização quanto ao processo de mensuração e seus projetos.

Estabelecer e sustentar comprometimento: assegura que a mensuração é apoiada tanto em nível de projeto quanto em nível de organização. Visa prover os recursos e a infra-estrutura organizacional requerida para implementar um projeto viável de mensuração.

Uma quinta atividade apresentada trata-se dos *Processos técnicos e de gerenciamento*, aos quais não correspondem a uma atividade específica de mensuração, mas mantém uma relação indireta com o processo de mensuração. Os gestores de um projeto operam nestes processos definindo as informações necessárias e usando as informações resultantes das medições realizadas para tomada de decisão.

Neste sentido, o PSM recomenda que o processo de mensuração deve estar integrado ao ciclo de vida do processo de *software.* Uma vez que o processo de *software* por natureza é um processo dinâmico, considera-se que o processo de mensuração deve também poder se adaptar aos projetos envolvidos. Assim, o processo de mensuração do PSM é interativo por definição. Ele permite sua adaptação às características e contextos de projetos particulares e também a adaptação a mudanças nos projetos e nos requisitos de informação.

Em relação a sua estrutura, o PSM está baseado em dois modelos: i) um modelo de informação, que determina como as medidas a serem utilizadas devem ser especificadas e estruturadas para atender às necessidades de informação; e ii) um modelo de processo, que especifica como o processo de mensuração deve ser conduzido (BORGES, 2003).

O modelo de informação é um mecanismo para mapear as necessidades de informação em termos dos atributos do produto ou processo que devem ser mensurados. Segundo ele, a construção de uma medição deve descrever como os atributos serão quantificados e combinados para formar indicadores que forneçam a base para a tomada de decisões. Para isso, devem ser definidas medidas em três diferentes níveis:

- *Medidas básicas*: medidas obtidas diretamente da observação das entidades do produto ou processo que está sendo medido, onde atributos dessas entidades são mapeados em valores de uma escala através de um método de medição;

- *Medidas derivadas*: valores obtidos a partir da aplicação de algoritmos ou funções matemáticas que combinam duas ou mais medidas (que podem ser medidas básicas ou mesmo outras medidas derivadas);
- *Indicadores*: visão de um conjunto de medidas em um formato (geralmente gráfico ou tabular) voltado para o atendimento de uma ou mais necessidades de informação.

A figura 2.5 apresenta como esses conceitos são combinados no PSM, formando uma estrutura para a construção das medições.

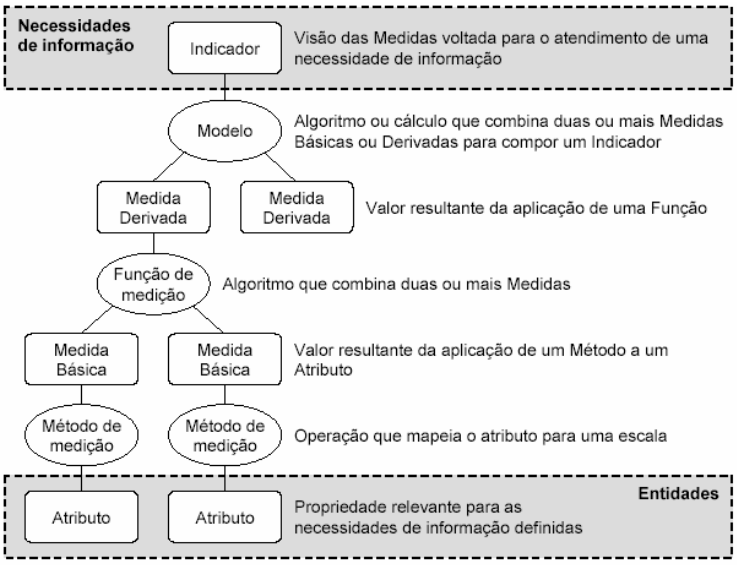


Figura 2.5: Construção de uma medição pelo PSM (BORGES, 2003)

Enquanto o modelo de informação fornece uma estrutura para relacionar as necessidades de informação a um conjunto de medidas, o modelo de processo apresenta referências para a implementação da mensuração em um projeto, descrevendo as principais atividades que devem ser realizadas.

Uma das características principais em relação ao mecanismo de seleção das medidas definidos pelo PSM é o mapeamento das necessidades de mensuração. De maneira geral, as necessidades de informação do projeto são mapeadas para às áreas comuns (*Issues*), categorias de medição (*Categories*) e medidas genéricas (*Measures*) que são definidas pelo PSM.

Common Issues Areas - são áreas de preocupação que possuem impacto ao atingimento dos objetivos de um projeto, e incluem problemas, riscos e carência de informações para previsão de impacto em um projeto (USA, 2003).

Measurement Categories - categorias de medição definem grupos de medidas relacionadas. As medidas em cada categoria provêem informações similares sobre uma área de medição (*issues*). Elas mapeam as características de um projeto e respondem questões similares.

Measures - algumas medidas candidatas são usualmente disponíveis para cada característica específica de um projeto. Uma medida é a quantificação de uma característica de um projeto ou de um produto.

A figura abaixo ilustra este mapeamento, também chamado de modelo ICM (*Issues - Categories - Measures*).

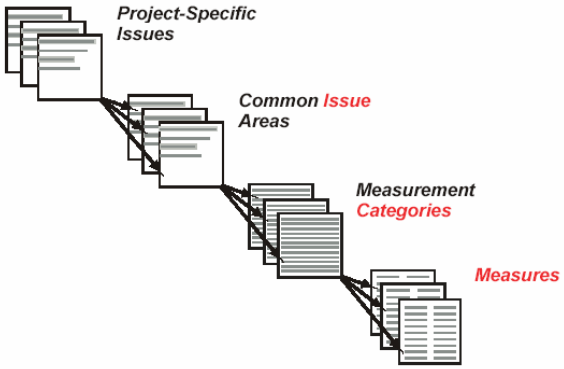


Figura 2.6: Mecanismo de seleção de medidas do PSM (Modelo ICM) (USA, 2003)

Em relação às medidas utilizadas no processo de mensuração, o PSM apresenta uma lista com várias sugestões (51 ao todo), que compões o modelo ICM e foram elaboradas com base na experiência de implantação de programas de medição nas organizações participantes. Essas medidas cobrem as seguintes áreas (*common issue areas*):

- *Cronograma e progresso*: área relacionada ao cumprimento tempestivo de marcos de projeto e à conclusão de unidades de trabalho nos prazos previstos.
- *Recursos e custos*: área que lida com a adequação entre o trabalho a ser executado e os recursos alocados ao projeto.
- *Tamanho e estabilidade de produtos*: lida com a estabilidade da funcionalidade ou da capacidade requerida do *software*, e com o volume de *software* necessário para contemplar essa capacidade.
- *Qualidade de produtos*: área relacionada à capacidade do *software* produzido em atender às expectativas dos usuários.
- *Desempenho de processos*: trata da capacidade do processo em atender às necessidades impostas por cada projeto.
- *Eficácia de tecnologia*: lida com a viabilidade e adequação das soluções técnicas propostas (arquiteturas, ferramentas e tecnologias), tendo em vista as características de cada projeto.
- *Satisfação dos clientes*: área relacionada ao grau com que os produtos e serviços oferecidos atendem às expectativas dos clientes.

Por fim, para auxiliar o registro e a análise das medidas, o material do PSM inclui também uma ferramenta, chamada PSM *Insight*. Disponibilizada gratuitamente, essa ferramenta permite a definição das medidas a serem coletadas, o registro dos valores medidos para cada uma delas e a construção de indicadores gráficos para auxiliar a análise dos resultados.

No entanto, a ferramenta disponibilizada pelo PSM é limitada aos registros básicos das informações relativas aos dados coletados, sendo totalmente

independente do processo de *software*. Além disto sua utilização é comprometida pelo fato de não suportar acessos concorrentes, característica fundamental para viabilizar o registro das medições pelos desenvolvedores que trabalham em equipes (BORGES, 2003).

Nesta seção apresentamos uma síntese dos principais conceitos relacionados ao PSM. Maiores detalhes podem ser encontrados na literatura em (USA, 2003), a qual apresenta na íntegra todo o guia do PSM.

Após conhecidas algumas das principais abordagens existentes na literatura referentes à mensuração em processos de desenvolvimento de *software*, na próxima seção iremos apresentar alguns requisitos relativos às atividades de mensuração, os quais foram identificados ao longo das pesquisas realizadas neste trabalho.

2.7 Requisitos relativos a mensuração em processos de *software*

Durante o estudo realizado foi possível identificar um número significativo de abordagens e recomendações importantes relativas à prática de mensuração em processos de *software*. Muitas destas, independentemente do contexto ao qual foram estudadas, são valiosas em termos práticos proporcionando um relevante referencial, tanto de boas práticas quanto das principais dificuldades existentes, fornecendo assim alguns indicativos de como conduzir de forma mais assertiva um processo de mensuração.

Apesar da grande quantidade de contribuições encontradas na literatura, não foi possível identificar (de forma direta) um conjunto específico de requisitos a serem utilizados como base para construção de novas ferramentas e metodologias que suportem a realização de mensuração. Neste sentido, optou-se pela elicitação de um conjunto de requisitos relevantes para criação de metodologias e ferramentas para mensuração, de forma a ser utilizado como base para este trabalho e também para futuras pesquisas, procurando reunir algumas das diversas contribuições dispersas na literatura de uma forma unificada e categorizada.

Cabe ressaltar que não pretende-se neste trabalho apresentar o melhor conjunto de requisitos, mas um conjunto básico, com base em recomendações consideradas por diversos autores como sendo fundamentais para a execução com qualidade de medições em *processos de software*, sendo este conjunto uma das contribuições deste trabalho para a área de pesquisa.

Assim, este conjunto de requisitos foi classificado em três categorias, visando um agrupamento contextual, divididos em:

1) Requisitos relativos ao planejamento e definição das medições;
2) Requisitos relativos ao processo de mensuração;
3) Requisitos relativos a análise, avaliação e divulgação dos resultados das medições.

Estas categorias e seus requisitos são descritos nas sub-seções que seguem.

2.7.1 Requisitos relativos ao planejamento e definição das medições

Uma das principais e mais importantes etapas de uma abordagem de mensuração é, sem dúvida, o planejamento das medições, pois é no planejamento que são estabelecidas as estratégias e as definições necessárias para a efetiva realização de cada etapa do processo de mensuração.

Devido a esta grande importância, entende-se como necessário que metodologias consistentes e reconhecidamente eficazes nos orientem na escolha das métricas e de como estas serão obtidas. Neste sentido, um conjunto de básico de requisitos relativos ao planejamento das medições foi identificado, visando apontar o que necessariamente deve contemplar o planejamento das medições. À seguir são apresentados estes requisitos.

R1.1 - Suporte ao planejamento das medições: a definição de um planejamento adequado (muitas vezes chamado de plano de medição) é fundamental para obtermos sucesso na realização da mensuração de *software*. Desta forma, é necessário entender a informação necessária para o projeto e definir medidas e métricas apropriadas para satisfazê-las, bem como os procedimentos para registro, análise e publicação da informação, resultando em um plano de medição. O armazenamento das definições das medições, para reuso em futuros projetos, também deve ser viabilizado (ABREU, 1992) (BASILI, 1994) (CARD, 2003) (FUGGETTA, 1998) (LANE, 1997) (MENESES, 2001) (OLSINA, 2002) (STATZ, 1999).

R1.2 - Suporte ao planejamento e definição do processo de medição: para garantir a confiabilidade e acurácia das informações coletadas, a definição das responsabilidades, calendários, procedimentos e obtenção de ferramentas de apoio à coleta e tratamento dos dados deve fazer parte do planejamento das medições. Além disto, para cada métrica a ser coletada, deve-se especificar os mecanismos e procedimentos a serem feitos no momento da coleta dos dados, a fim de obtermos o valor de uma métrica e uma repetibilidade na forma de coleta (ABREU, 1992) (AGUIAR, 2002) (DAWSON, 2003) (FUGGETTA, 1998) (OLSINA, 2002).

R1.3 - Seleção e classificação dos objetivos da medição: a literatura corrente aponta que uma formalização prévia das metas no plano de medição é um aspecto importante para descartar metas irrelevantes tão logo for possível, permitindo assim a seleção e classificação dos alvos em uma base clara e objetiva. Deve-se manter o foco nos alvos da organização (ABREU, 1992) (BASILI, 1994) (USA, 2003) (LANE, 1997) (FUGGETTA, 1998) (PERKINS, 2003) (PARK, 1996)

R1.4 - Compartilhamento das medições: sempre que possível, as medições realizadas devem poder ser compartilhadas por diferentes perspectivas de informação, ou seja, ainda que diferentes métricas necessitem de um mesmo dado coletado para seu cálculo, este dado deve ser coletado uma única vez, podendo satisfazer diferentes necessidades analíticas sobre ele. Durante a definição das medições, este compartilhamento deve ser previsto (FUGGETTA, 1998).

R1.5 - Adequação aos diferentes contextos e níveis de maturidade das empresas: as empresas apresentam diferentes níveis de maturidade e, por isto, a sistemática de mensuração deve se adaptar aos objetivos e ao negócio da organização, se adequando tanto a uma organização com pouca maturidade como

àquelas que estejam em níveis superiores. Além disto, as medições são realizadas no sentido de suprir diferentes necessidades de informação de uma organização. Mesmo dentro de uma mesma organização a necessidade de informação varia por usuário. A escolha das métricas deverá enquadrar-se no processo adotado e, se possível, ter um impacto direto na qualidade do *software* produzido. Para diferentes produtos do mesmo produtor poderão ser apropriadas diferentes métricas. Desta forma, a sistemática de mensuração deve dar apoio a medições básicas de controle, como também deve permitir que, caso a organização já possua alguma maturidade, o sistema de medição possa estar mais alinhado aos processos da organização, podendo manipular medições mais sofisticadas. (ABREU, 1992) (ANACLETO, 2003) (BASILI, 1994) (FRANÇA, 1998) (FUGGETTA, 1998) (HERBSLEB, 1998) (LANE, 1997) (OLSINA, 2002) (STATZ, 1999)

R1.6 - Possibilidade de criação de nova métricas: uma organização com pouca maturidade pode **c**omeçar com poucas medições e evoluir aos poucos, avaliar e refinar as medições, categorizando-as. Neste sentido, novas métricas podem ser criadas no sentido de atender novas demandas de informação. A necessidade de obter informações mais detalhadas envolvendo as aplicações desenvolvidas, faz com que métricas relacionadas a diferentes tipos de aplicações sejam criadas. Assim, o conjunto de métricas não deve ser estático, isto é, algumas das métricas inicialmente propostas poderão ser abandonadas e outras poderão ser adotadas em sua substituição. (ABREU, 1992) (ANACLETO, 2003) (FRANÇA, 1998)

R1.7 - Reuso e adaptação dos planos de medição: a criação de um plano de medição não constitui tarefa simples, e mesmo quando realizamos um planejamento prévio da medições, não existem garantias de que o plano de medição será adequado. Assim, permitir o reuso de planos de medição anteriormente utilizados, realizando-se as devidas adaptações quando necessário, pode reduzir significativamente os custos de desenvolvimento de novos planos, uma vez que podemos aproveitar as experiências de projetos passados em prol de um novo planejamento mais consistente (FUGGETTA, 1998) (LANE, 1997) (OLSINA, 2002) (STATZ, 1999).

R1.8 Catálogos de medições: a criação de catálogos de medições contendo informações sobre o que medir, como e para que, pode auxiliar significativamente na definição do plano de medição (OLSINA, 2002).

2.7.2 Requisitos relativos ao processo de medição

Tão importante quanto a realização de um planejamento adequado e a definição das medidas relevantes para cada contexto, é a garantia de que o processo de medição seja eficaz no sentido da confiabilidade das medições. Além disto, um cuidado importante que deve ser levado em consideração é o custo deste processo de mensuração. Neste sentido, são apresentados aqui um conjunto de requisitos relevantes em relação ao processo de medição, ou seja, como deverão ser coletadas e analisadas as informações acerca do processo de desenvolvimento de um *software*.

R2.1 - Integração das atividades de mensuração ao processo de produção do *software*: as atividades relativas à mensuração (planejamento, execução,

avaliação, etc.) devem fazer parte do processo de produção de *software*, tal como é corrente na generalidade dos processos industriais. Desta forma, poderá permitir a previsão e monitorização dos custos, controlar a qualidade e, por outro lado, melhorar a compreensão e validação não somente do processo de desenvolvimento, mas também do próprio processo de mensuração, uma vez que o acréscimo de atividades de mensuração ao processo de desenvolvimento, pode implicar em aumento dos custos e tempo do ciclo de desenvolvimento. A obtenção desses dados quantitativos é importante não só para fins de avaliação, mas também em função da possibilidade de se estabelecer condições contratuais (prazos e orçamentos), tomar decisões de atribuição de recursos humanos e financeiros, ou tentar atingir objetivos definidos. Um ponto importante a ser observado é que o processo de mensuração deve estar relacionado ao processo desenvolvimento, mas deve-se ter a preocupação de mantê-lo com poucas rupturas. (ABREU, 1992) (AGUIAR, 2002) (ANACLETO, 2003) (DAWSON, 2003) (FUGGETTA, 1998) (STATZ, 1999).

R2.2 – Suporte a coleta de dados das medições: consiste na realização da coleta propriamente dita dos dados da medição, permitindo a obtenção da base de informações para o cálculo de métricas e/ou indicadores, para produzir os resultados necessários para suporte aos tomadores de decisão. A coleta de dados sempre que possível deve ser feita através de uma ferramenta, utilizando-se de uma base de dados para armazenamento das medições durante as atividades. A coleta de dados automática é uma possibilidade que deve sempre ser considerada, pois pode permitir a redução do tempo gasto em atividades de coleta de dados (ABREU,1992) (BASILI,1995) (CARD,2003) (DAWSON,2003) (FRANÇA,1998) (HERBSLEB,1998) (LANE, 1997) (MENESES, 2001) (STATZ, 1999).

R2.3 - Histórico das medições: as medições devem ser registradas em um banco de dados. Este banco de dados deve servir tanto para o registro do plano de medição quanto das medidas coletadas, para que possa ser utilizado como base para tomada de decisão (ABIB, 1999) (AGUIAR, 2002) (BASILI, 1995) (FRANÇA,1998) (FUGGETTA,1998) (KRISHNAM, 1999) (STATZ,1999).

R2.4 – Avaliação e monitoramento do custo/esforço relativo ao processo de medição: o planejamento e a execução de um plano de medição naturalmente gera um esforço adicional ao projeto como um todo. Este esforço, e, consequentemente o custo agregado a ele, deve ser medido para avaliar o quanto o processo de medição escolhido influencia no andamento do projeto. A possibilidade de avaliar o aumento do tempo do processo de desenvolvimento através da introdução das atividades de medição deve ser considerada (ANACLETO, 2003) (FUGGETTA, 1998).

R2.5 Extração de dados de outras ferramentas/bases de dados: a informação útil nem sempre está disponível de forma que possa ser lida. Freqüentemente estão espalhadas por bases de dados corporativas, ou ainda misturadas com outros dados que não são relevantes. Assim, para a obtenção dos dados necessários para realização das medições, muitas vezes existe a necessidade de obtermos dados oriundos de outras fontes, o que faz necessário que ferramentas que apoiem a coleta de dados de medição possibilitem a aquisição de dados externos (FUGGETTA, 1998) (OLSINA,2002).

2.7.3 Requisitos relativos a análise, avaliação e divulgação dos resultados

A medição por si só não gera resultados relativos a melhoria do processo ou do produto, a menos que os dados obtidos sejam efetivamente analisados e ações com base nestas análises seja tomadas. Para tanto, uma abordagem de mensuração deve prover os dados necessários para estas análises de forma adequada e de fácil utilização. Além disto, existe a necessidade de avaliarmos também se a estratégia utilizada para mensuração está sendo adequada e produz os resultados esperados. Neste sentido, são apresentados nesta seção alguns requisitos básicos acerca da divulgação dos dados relativos às medições realizadas, bem como das análises dos dados coletados, além da própria avaliação do programa de mensuração.

R3.1 - Divulgação das Medições: a sistemática de mensuração deve prover mecanismos de divulgação das medições realizadas, permitindo que várias pessoas de diferentes lugares possam ter pronto acesso às informações mais atuais sempre que necessário. Apresentação dos dados de forma gráfica e de fácil compreensão, além de relatórios sobre as análises realizadas e as mudanças no processo de desenvolvimento (BASILI,1995) (DAWSON, 2003) (FRANÇA, 1998) (MENESES, 2001) (LANE, 1997)

R3.2 - Suporte às análises dos dados coletados: uma das fases mais importantes do processo de mensuração compreende a análise e interpretação dos dados coletados. Assim, a análise das métricas computadas (com base nos dados recolhidos e sua confrontação com os objetivos definidos), no sentido de obter embasamento para tomada de decisão, pode possibilitar a melhora do processo de desenvolvimento, bem como do próprio processo de mensuração. Além disto, a revisão da qualidade do produto com base nas medições realizadas, a melhoria dos problemas encontrados e a identificação e aplicação de melhorias ao processo de desenvolvimento também podem ser obtidas através da análise dos dados da mensuração (AGUIAR, 2002) (FUGGETTA, 1998) (OGASSAWARA, 1996).

R3.3 - Registro das decisões: é bastante útil registrar o que foi racionado sobre as principais decisões que foram tomadas durante o processo, armazenando-os em uma base de experiências, as lições, resultados, aprendizados, sucessos, fracassos, e toda documentação produzida pela mensuração (ABIB,1999) (AGUIAR, 2002) (FRANÇA, 1998) (FUGGETTA, 1998) (KISHNAM,1999) (MENESES, 2001).

R3.4 - Permitir a avaliação do plano de medição: consiste em avaliar as métricas utilizadas, o processo de mensuração e suas estratégias, verificando sua consistência e o atingimento dos objetivos propostos. Além disto, deve-se registrar as falhas no processo, aumentando o conhecimento sobre o mesmo, e prevenindo a repetição de erros futuros, permitindo também embasamento para novos planejamentos (AGUIAR, 2002) (CARD, 2003) (DAWSON, 2003) (FUGGETTA, 1998) (MENESES, 2001).

2.8 Considerações sobre o capítulo

No capítulo 2 apresentamos uma breve revisão teórica relativa aos principais conceitos ligados à área de mensuração. Também foram apresentadas algumas das

principais abordagens existentes na literatura e que têm servido de embasamento teórico para inúmeras pesquisas relacionadas a esta importante área.

De maneira geral, as principais abordagens existentes ressaltam a grande importância na realização de um forte planejamento e coordenação das atividades de mensuração, para que os resultados obtidos sejam efetivos. Assim, neste capítulo também procuramos apontar, de maneira geral, alguns requisitos que apoiam justamente estas estratégias de planejamento e execução de planos de mensuração, sendo estes requisitos fundamentados em diversos trabalhos e pesquisas ao qual foram objetos de estudo desta pesquisa.

Uma vez apresentadas as necessidades inerentes a melhoria dos processos de *software*, os principais fundamentos relativos à mensuração de *software* (como uma das estratégias para melhoria dos processos) e traçados também os requisitos necessários ao atingimento dos objetivos propostos neste trabalho, na próxima seção apresentaremos alguns dos trabalhos relacionados com os objetivos de pesquisa deste trabalho, visando apresentar um pouco do estado da arte na área de pesquisa ao qual este trabalho está inserido.

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns dos trabalhos relacionados ao contexto do estudo realizado. Entre os vários trabalhos realizados por pesquisadores desta área, um grande número concentra-se em experimentos e relatos de experiências realizadas com a utilização das principais abordagens utilizadas para mensuração, tais como GQM, GQ(i)M, PSM, além de outras variações destas abordagens. Alguns exemplos são as pesquisas de (FUGGETTA, 1998), (HERBSLEB, 1998), (PERKINS, 2003), (ZUBROW, 2001), entre outros. Outras pesquisas concentram-se em apresentar novas abordagens para realização de mensuração, ou ainda ferramentas que apoiem abordagens. Alguns exemplos seriam (ANACLETO, 2003), (BRIAND,2002), (BORGES, 2003), (GOETHERT, 2003), (FRANÇA, 1998), (OLSINA, 2002), entre outros.

Como o âmbito deste trabalho concentra-se principalmente na definição de um modelo para apoiar a realização de mensuração em processos de *software*, foram relacionados neste capítulo três trabalhos que teriam características similares, um em nível nacional (BORGES, 2003), e outros dois em nível internacional (BRIAND, 2002) e (OLSINA, 2002). Nas sub-seções à seguir apresentaremos uma breve introdução sobre cada um deles.

3.1 Modelo de medição para processos de desenvolvimento de *software*

Nesta sub-seção será apresentado um modelo de medição para processos de desenvolvimento de *software*, descrito em (BORGES, 2003) e desenvolvido na Universidade Federal de Minas Gerais. O trabalho explora aspectos comuns dos processos de desenvolvimento mais utilizados pela indústria de *software* e propõe um arcabouço genérico de medição que, segundo o autor, pode ser configurado e aplicado a boa parte deles. Técnicas consagradas pela literatura são utilizadas para selecionar e definir um conjunto de medidas capaz de caracterizar quantitativamente alguns aspectos chave do ciclo de desenvolvimento (tais como: esforço, tamanho dos produtos, defeitos, progresso, cronograma e estabilidade de requisitos) e fornecer dados que contribuam para a melhoria da qualidade dos produtos e da produtividade e previsibilidade dos processos.

A proposta principal deste trabalho consiste na confecção de um modelo de medição voltado para processos de desenvolvimento de *software* que sejam baseados em arquiteturas matriciais[1], baseado nos seguintes elementos.

- Definição de um conjunto de atributos cuja mensuração seja significativa para a obtenção de informações úteis sobre o andamento dos projetos e a melhoria dos processos (exemplos de atributos são o tamanho dos produtos, o esforço dedicado aos projetos e os defeitos existentes nos produtos desenvolvidos);
- Convenções e procedimentos que descrevam a forma pela qual cada atributo deve ser mensurado;
- Um modelo conceitual que une todos os atributos em uma única estrutura, formalizando os conceitos envolvidos e explicitando os relacionamentos existentes entre eles.

Além disso, a meta do trabalho é que o modelo de medição proposto atenda aos seguintes requisitos:

- Ser flexível o suficiente para que possa ser adaptado a diferentes processos de arquitetura matricial, ou a personalizações de um mesmo processo;
- Ser fundamentado em técnicas consagradas pela literatura e amplamente utilizadas, como o paradigma GQM, o *Goal-Driven Software Measurement* e o PSM.

Para validar o modelo e exemplificar como o mesmo pode ser aplicado, este trabalho prevê também a descrição de sua instanciação em um processo, o Praxis 2.0.

Também fazem parte deste trabalho a especificação e modelagem de uma ferramenta para esse fim.

Segundo o autor, a proposta apresenta um foco direcionado ao modelo de informação, ou seja, um modelo que determina como as medidas a serem utilizadas devem ser especificadas e estruturadas para atender às necessidades de informação, não abrangendo assim o modelo de processo, que especifica como o processo de mensuração deve ser conduzido.

De forma resumida, a abordagem proposta se baseia nas seguintes etapas:

Seleção das medidas: para guiar a seleção dos elementos que irão compor o modelo de medição, foi utilizada a seqüência de passos proposta pelo GQ(i)M (*Goal-Driven Software Measurement*), com algumas alterações no processo. Estas alterações concentram-se basicamente na utilização de metas geralmente comuns a todas as organizações produtoras de *software* e a omissão das etapas de utilização de descrições do ambiente, que tratam de detalhes técnicos sobre a organização e

[1] Processos que dividem as atividades no ciclo de desenvolvimento em 2 dimensões, formando uma estrutura matricial: uma dimensão relacionada ao tempo (etapas), e uma divisão em fluxos de trabalho, que classifica as atividades executadas ao longo do desenvolvimento de um produto em disciplinas de Engenharia de *Software* (requisitos, implementação, testes etc.) (BORGES, 2003).

projetos que serão submetidos à medição, e também do planejamento da implantação das práticas de medição na organização, que consiste na identificação das ações necessárias e da confecção de um plano de implantação na organização.

Definição das metas de negócio e submetas: definição de metas genéricas básicas para qualquer organização;

Formalização das métricas: especificação do objeto de interesse, proposta, perspectiva, descrição do ambiente;

Formulação das perguntas e indicadores: especificação das perguntas necessárias para atingir as metas, e os indicadores;

Identificação dos elementos de dados: definir as medidas a serem coletadas para elaboração dos indicadores;

Confecção do modelo: apresenta uma formalização das medidas básicas, através de diagramas de classe, aos quais apresenta uma descrição e um entendimento sobre cada medida, além da forma como o mesmo deverá ser coletado (através de um formulário). Por fim, todas as medidas são agrupadas em um modelo conceitual. Permite a derivação de medidas, a partir das medidas básica apresentadas anteriormente.

Aplicação do modelo: o modelo proposto foi aplicado a um processo específico, no caso o PRAXIS, um processo de desenvolvimento de *software* para pequenas empresas, que define as etapas e interações necessárias para o desenvolvimento. Foram realizadas algumas adaptações necessárias para a evolução do processo incluindo a parte de medições.

Ferramenta Quantum: além do modelo referido, uma ferramenta de apoio foi desenvolvida, com o foco na coleta e armazenamento das medidas. A ferramenta não possui facilidades de análise e nem extração automática das medidas através do processo, nem mesmo facilidades para integração com outras ferramentas para obtenção dos dados.

Considerações sobre a abordagem:

O modelo proposto é baseado no modelo de informação (especificação e estruturação das medidas), sendo centrado nas métricas e seus relacionamentos e não no processo de medição. Já a ferramenta proposta trata-se de uma ferramenta para suporte (coleta e armazenamento), não oferecendo suporte ao programa de mensuração, nem a avaliação da mensuração realizada.

Como sugestões de trabalhos futuros são apontados a definição de um processo de medição, incorporado ao processo de desenvolvimento, técnicas mais formais para análise das medições e a coleta automática das medidas.

3.2 *Framework* para definição e exploração de métricas de qualidade

Este trabalho refere-se à definição de um *framework* para permitir a definição, documentação e exploração de métricas de qualidade apresentado em (OLSINA, 2002) e desenvolvido na *Universidad de la Pampa*, na Argentina, em colaboração

com pesquisadores da *Universidad de Málaga* (Espanha) e *Universidad de la Habana* (Cuba).

O trabalho apresenta alguns requisitos para criação de um *framework* para implementação de um repositório de informações de atributos e métricas, resumidos em:

- Interoperabilidade: facilitar o intercâmbio das informações entre aplicações distintas, usuários e plataformas, utilizando padrões de mercado como XML;
- Generalidade: proporcionar um marco comum de trabalho e de documentação de atributos e métricas, que possa ser aproveitado em diferentes contextos;
- Simplicidade: o repositório seja de fácil utilização por administradores, usuários, através de operações e padrões simples;
- Segurança: o acesso aos serviços, dados e os meta-dados do repositório possuam diferentes níveis de segurança e com papéis bem definidos;
- Autoridade: que responda às normas estabelecidas e da maneira de discutir e acertar atributos e métricas candidatas, assim como os procedimentos para registrar, atualizar e publicar a informação.

Com base nestes requisitos, foi desenvolvido um modelo conceitual para projetar e implementar um repositório de métricas, representando seus elementos e suas relações necessárias para definir e explorar informações de distintas métricas e atributos para diferentes domínios. A idéia base é prover uma infra-estrutura genérica, capaz de integrar diferentes tipos de modelos de medição, utilizando como base conceitos de Web Semântica e tecnologias amplamente utilizadas como UML, XML e RDF. O modelo não prevê o armazenamento de instâncias de métricas de diferentes projetos, sendo proposto como trabalhos futuros.

De maneira geral, os autores apresentam um modelo baseado em objetos (apresentado na figura 3.1), os quais podem ser processos, produtos, recurso, projeto, etc, divididos em entidades, que podem ser medidas diretamente.

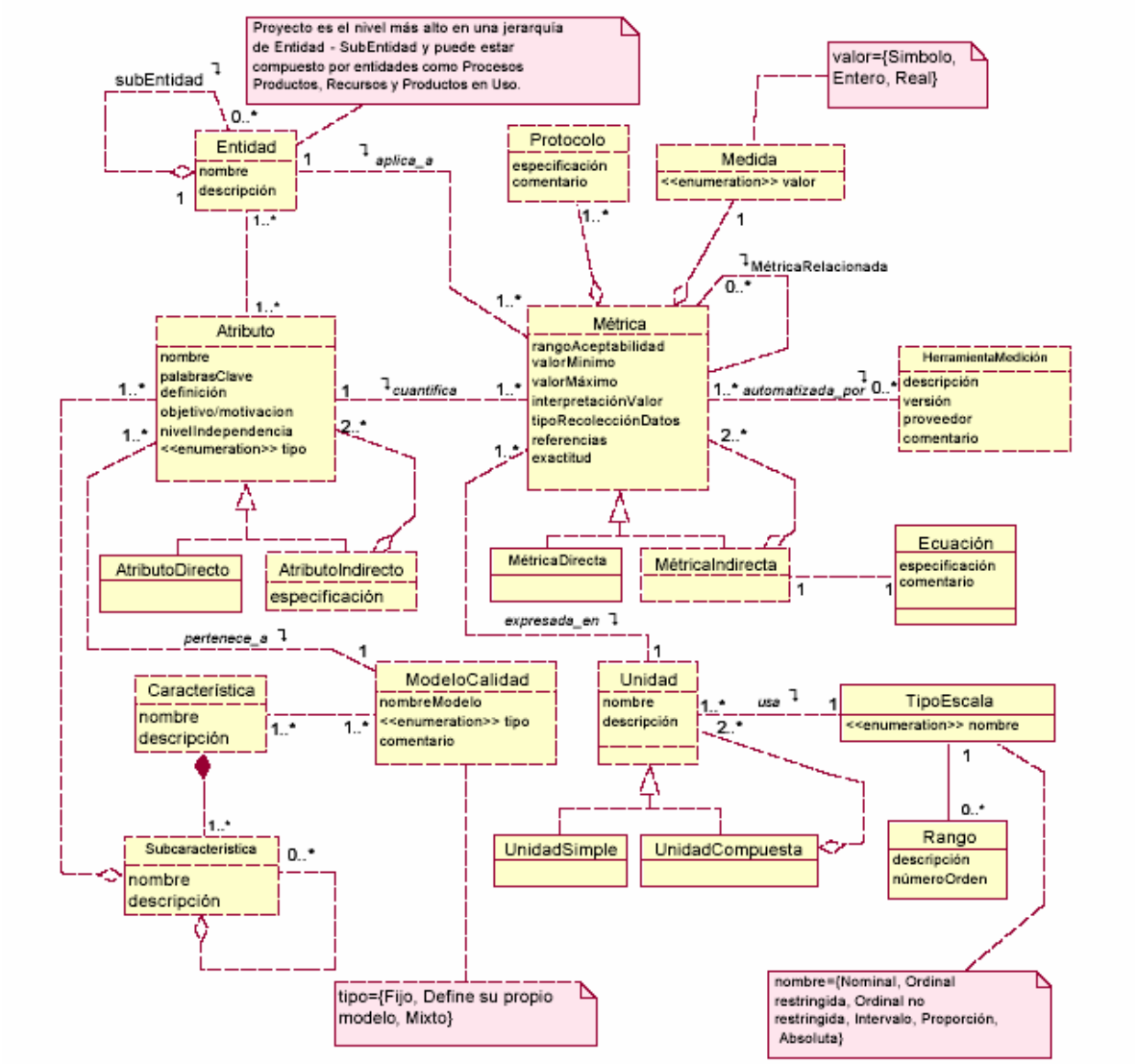


Figura 3.1: Modelo conceitual para domínio de métricas (OLSINA, 2002)

O modelo define um conjunto de classes representando as métricas, protocolos de medição, métricas diretas, indiretas, atributos, entre outras classes relacionadas com a medição propriamente dita.

Os principais elementos do modelo conceitual são resumidos em:

- Classes entidade e atributo: uma entidade representa um objeto tangível ou intangível, que exibe um comportamento observável do mundo real (um domínio empírico). No âmbito de *software*, o autor exemplifica os entes por projetos, processos, recurso, produto ou produto em uso. Estes por sua vez podem ser divididos em sub-entes de diferentes tipos. Estes entes não podem ser medidos diretamente, senão através de propriedades, representadas pelos atributos. Um exemplo seria um *site* da *Web*, que possui páginas. Um atributo mensurável poderia ser, por exemplo, o tamanho da página.
- Classe métrica e classes relacionadas: um atributo pode ser quantificado por uma ou várias métricas, que podem ser diretas ou indiretas. Cada métrica deve ter tipo de valor a se obter, a unidade em que é expresso, o tipo de

escala que se usa, regras de conteúdo e de procedimentos (protocolo de medição) e se existe coleta automática.

Para documentação dos atributos de qualidade das métricas, é utilizado XML, para favorecer a independência das tecnologias. Além disto, os autores apresentam um *framework*, que é utilizado através de *Web-services* através de regras de acesso.

Considerações sobre a abordagem:

O *framework* proposto também é baseado em um modelo de informação, sendo centrado nas métricas e seus relacionamentos e não no processo de medição em si. A proposta de uma infra-estrutura genérica pode ser uma alternativa bastante interessante apresentada pelos autores em função da aplicabilidade do modelo a diferentes contextos, diferentemente do trabalho apresentado anteriormente cuja aplicabilidade era mais restrita aos modelos de processo matriciais.

Neste trabalho não são enfatizados aspectos relativos a condução da definição das medidas apropriadas a cada caso (aspecto que está relacionado ao processo de mensuração propriamente dito). Em relação a integração das medições ao processo de desenvolvimento, o presente trabalho também não trabalha estes aspectos.

3.3 *GQM/MEtric DEfinition Approach*

A abordagem GQM/MEDEA envolve pesquisadores de instituições dos Estados Unidos (*University of Maryland* - Victor Basili), Itália (*Università degli Studi dell'Insubria* - Sandro Morasca), e Canadá (*Carleton University* - Lionel Briand) e refere-se a definição de um processo para definição de medidas de *software*, que pode ser utilizado como um diretrizes práticas para projeto e reuso de medidas úteis. A abordagem é derivada da experiência adquirida pelos autores em um grande número de projetos em diferentes ambientes e baseada na abordagem GQM.

O foco da abordagem é a construção de sistemas de predição, isto é, modelos que estabelecem uma correspondência entre medidas para atributos de *software*. Estas medidas podem quantificar variáveis dependentes, usualmente relatadas como um atributo de um produto ou processo de interesse da indústria (como por exemplo, propensão a falhas, custos, tempo de entrega). Outras medidas quantificam produtos e atributos de processo (como por exemplo, acoplamento de componentes) e são variáveis independentes do modelo.

Baseado no conhecimento do ambiente de aplicação, uma definição explícita dos objetivos específicos a serem indicados e um conjunto de hipóteses que necessitam ser validadas são identificadas. Além disto, são identificados atributos de interesse e definidas as medidas, teoricamente válidas para eles. Estas medidas são usadas para validar as hipóteses experimentais.

A abordagem apresenta um *framework* que utiliza vantagens de diversas pesquisas presentes na literatura, tratando-se de uma extensão do paradigma GQM (apresentado na seção 2.6.1), e que provê um mecanismo para geração de modelos de mensuração (segundo os autores, o maior dos desafios do GQM). A figura 3.2 apresenta uma visão em alto nível dos passos do GQM/MEDEA, apresentada em notação de um DFD (Diagrama de Fluxo de Dados).

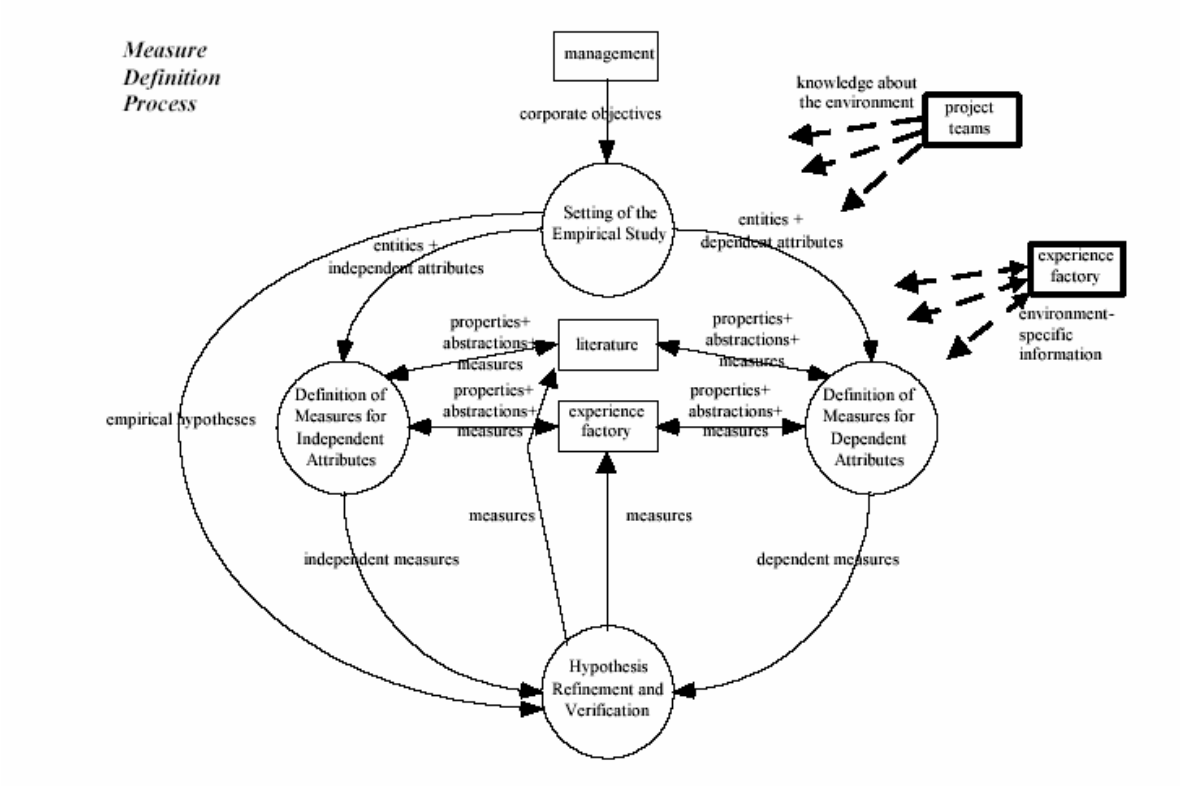


Figura 3.2: Passos em alto nível do GQM-MEDEA (BRIAND, 2002)

De maneira resumida, as atividades apresentadas no DFD da figura 3.2 consistem em:

Configuração do estudo empírico: onde os objetivos corporativos são refinados em alvos da medição, baseado no conhecimento sobre o ambiente, que ajudam a identificar processos e produtos aos quais medidas precisam ser indicadas. Baseado nos alvos da mensuração, um conjunto de hipóteses empíricas são estabelecidas e relacionam atributos (independentes) de algumas entidades (exemplo: componentes de *software*) para outros atributos (dependentes) da mesma ou de outras entidades. Atributos dependentes são usualmente atributos externos de qualidade de sistemas de *software* ou partes deles, por exemplo, confiabilidade, manutenabilidade, esforço, etc. Atributos independentes capturam fatores que tem uma relação causal com o atributo independente. Por exemplo, considere um alvo de mensuração relativo a predição do esforço de desenvolvimento. A hipótese empírica indica que o tamanho do produto é diretamente relacionada ao esforço.. Objetivos claros indicam a definição clara de atributos dependentes e independentes e de como são referenciados na mensuração de *software*.

Definição das medidas para atributos independentes: atributos independentes são formalizados através de um conjunto de propriedades genéricas que caracterizam suas medidas (exemplo: propriedades matemáticas, como adição). Entidades são formalizadas via abstrações (isto é, gráficos de dependência) que são escolhidas com base nas entidades, atributos independentes e propriedades genéricas. As propriedades genéricas são então instanciadas em abstrações e refinadas para obter-se uma suposição adicional que depende de um ambiente de

aplicação específico. Baseado neste conjunto refinado de propriedades, medidas são definidas ou selecionadas dos atributos ou entidades.

Definição das medidas para atributos dependentes: o GQM/MEDEA trabalha com atributos dependentes e independentes da mesma maneira. No contexto do projeto experimental, passos 2 e 3 correspondem ao passo de procedimento de definição de medidas que otimizam o que é referido como uma construção válida, isto é, o fato de que a medida adequada captura o atributo proposto pela medida.

Refinamento e verificação das hipóteses: as hipóteses empíricas são verificadas usando as medidas definidas para atributos dependentes e independentes. Eles podem ser refinados para prover uma forma funcional específica para os relacionamentos entre medidas dependentes e independentes, isto é uma relação exponencial. Estas hipóteses empíricas novas e mais específicas são então utilizadas para construir o modelo preditivo baseado nos dados atuais do desenvolvimento. Este modelo pode ser usado para verificar a plausibilidade das hipóteses empíricas e como um modelo de predição.

A figura 3.2 mostra que a maioria das saídas (exemplo: abstrações, medidas) dos passos definidos acima são reusáveis. Elas precisam ser empacotadas e armazenadas para que sejam eficientemente e efetivamente reutilizadas, de modelo a reduzir o custo de mensuração em uma organização. Em um ambiente de desenvolvimento maduro, entradas para maioria dos passos devem vir de um conhecimento reutilizado.

Os conceitos envolvidos no processo do GQM/MEDEA foram formalizados e apresentados na figura 3.3, através de um diagrama de classes UML. O diagrama, segundo os autores, pode servir como um ponto de início para o projeto orientado a objetos de uma ferramenta que suporte a metodologia e reuso de informações de um programa de mensuração.

Na figura 3.4, vemos que os alvos táticos (*TacticalGoals)* são relacionados com os objetivos corporativos (*Corporateobjective)* e guiam os alvos da mensuração (*MeasurementGoals)*, mostrando quais alvos são priorizados. Os alvos de mensuração possuem um ponto de vista (*Viewpoint*), propósito (*Purpose*), e um ambiente (*Environment)*. O foco de qualidade é modelado por uma associação de mesmo nome entre os alvos de mensuração e atributos (*Attribute*). Alvos táticos e alvos de mensuração não compartilham associações e atributos porque não foi definida a super classe “Alvo” (*Goal*).

Um alvo de mensuração vincula a medição a um atributo, na qual é associado com uma medida (*Measure)*. Atributos possuem propriedades (*Properties*) que são envolvidas em hipóteses empíricas (*Empirical hypotheses*), descrevendo relacionamentos entre dois ou mais atributos. Um alvo de mensuração é parte de um programa de mensuração *(MeasurementProgram)*, na qual consome recursos *(Resources)*, (pessoas por exemplo). Medidas são envolvidas em um modelo de predição *(PredictiveModels)* podendo ser variáveis dependentes ou independentes (modeladas por uma restrição OR). Medidas são obtidas pelas abstrações (um gráfico dirigido (*DirectedGraph*), por exemplo), na qual são representações analisáveis de entidades (um componente (*Component*), por exemplo). Medidas podem ser definidas com base em outras medidas, conseqüência da associação

reflexiva na classe. Modelos preditivos são definidos com base nas hipóteses refinadas (*RefinedHypothesis)*, na qual refinam previamente as hipóteses empíricas definidas.

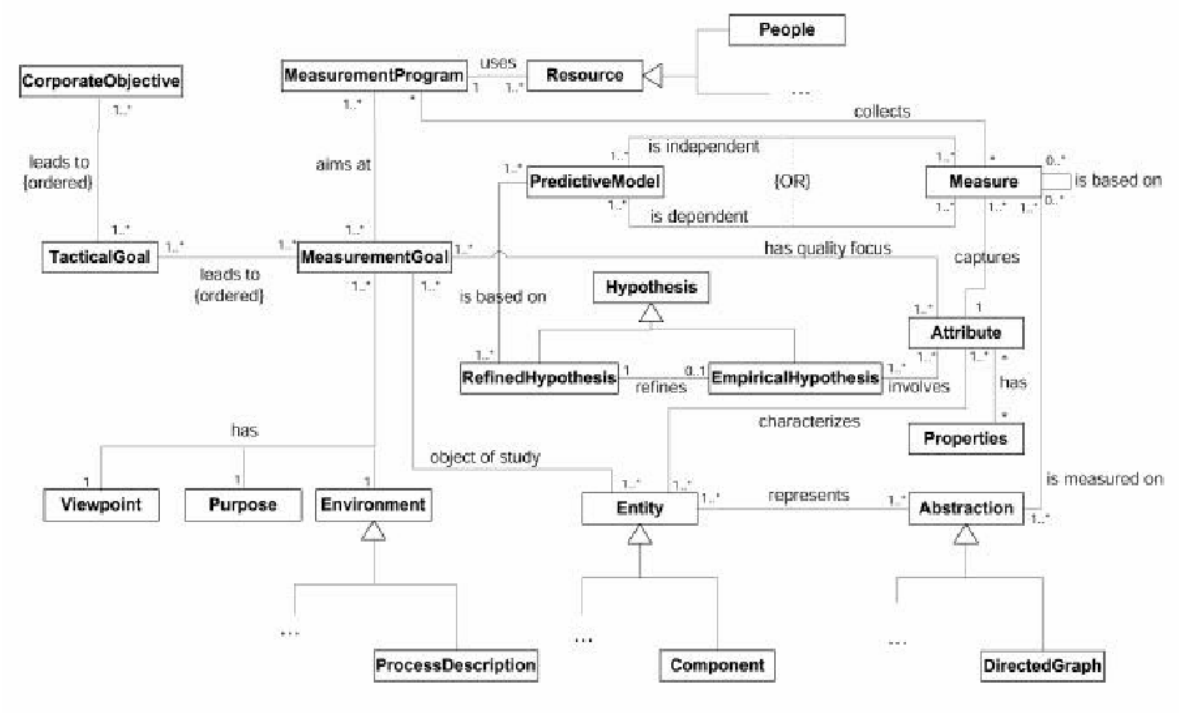


Figura 3.3: Modelo conceitual usando diagrama de classes UML (BRIAND, 2002)

Considerações sobre a abordagem:

A principal contribuição do GQM/MEDEA para o GQM é a definição de passos organizados para a definição de medidas para produtos de *software* baseada nos alvos GQM.

O GQM preenche a lacuna entre alvos e medidas pelo entendimento de questões e modelos, e estabelece recomendações de como defini-las. Já o GQM/MEDEA não se baseia em questões para preencher a lacuna, mas define os passos para levar em conta os relacionamentos entre eles, as fontes de informação, e o uso integrado de hipóteses, abstrações e validação teórica e empírica.

Os autores afirmam que as medidas de *software*, para serem definidas, necessitam de uma forma rigorosa e disciplinada, baseada em alvos precisos, suposições, propriedades, e através de validação experimental. O GQM/MEDEA propõe uma abordagem prática para definir medidas de *software* que implementam estes princípios e integra muitas das contribuições da literatura. Ela resulta da experiência dos autores e foi validada através de exemplos e campos de estudo reais. Este trabalho completa uma pesquisa prévia dos autores em uma maneira prática e operacional, fruto de algumas das lições aprendidas como resultado da experiência dos autores.

Como contribuições específicas, o *framework* provê uma descrição detalhada de várias atividades envolvidas na definição de medidas e do fluxo de informações sobre estas atividades, como: a) a ligação entre a definição de medidas para os

alvos corporativos e o ambiente de desenvolvimento; b) apresenta que as medidas não devem ser definidas por si, elas precisam ser definidas através de um contexto teórico; c) melhor ajuda para justificar, interpretar e reuso de medidas; d) ajuda na identificação de problemas que podem surgir durante a definição das medidas, levando em consideração que trata-se de um processo intensamente humano; e) provê um modelo conceitual que pode ser usado para implementar um esquema de repositório que contém conhecimento relevante para mensuração.

Futuros trabalhos relacionados pelos autores tratam de estudos mais detalhados e validação de cada um dos passos envolvidos na abordagem de mensuração. Especificamente, é necessário: a) melhor enteder como resultados experimentais podem ser usados para guiar o refinamento das medidas, baseado em aumento do entendimento sobre o processo de desenvolvimento e sua evolução; b) melhor identificar o que pode ser reutilizado entre os ambientes e projetos, por exemplo, medidas, suposições, conceitos mensuráveis, abstrações; c) prover diretrizes mais precisas para ajudar em algumas decisões subjetivas envolvendo a abordagem GQM/MEDEA.

Lacunas observadas na abordagem referem-se a não existência no modelo de informações relativas ao *feedback* obtido com as medições realizadas, ou seja, o empacotamento das experiências e análises referentes às medidas coletadas. Outro aspecto que poderia ter sido considerado refere-se a forma como deverão ser conduzidas as atividades de mensuração e sua relação com o processo técnico e gerencial.

3.4 Considerações sobre o capítulo

Neste capítulo apresentamos alguns importantes trabalhos relacionados com a definição de modelos para aplicação de mensuração em processos de *software*. Diante das pesquisas apresentadas, tivemos um importante entendimento em relação a definição de modelos de métricas e sua estruturas relacionadas.

Uma análise importante apresentada por (OLSINA, 2002), em relação as propostas existentes na área, aponta que apesar do valor inquestionável de várias propostas, o fato de terem sido desenvolvidas de uma forma independente e para domínios muito específicos, apresentam dificuldades na hora de serem utilizadas de forma mais genérica. Em particular, observa-se que: a maioria das propostas existentes na literatura, não estão estruturadas de uma forma consistente entre si, e a terminologia que utilizam, tanto do ponto de vista sintático como semântico, não são unificadas (a pesar de serem baseadas em padrões internacionais). Além disto, a documentação associada a cada métrica não é uniforme, o que impede sua possível reutilização em outros ambientes e a construção de ferramentas que façam uso delas por terceiros, sendo que os trabalhos sobre métricas parecem estar completamente distantes das propostas e métodos de seleção e avaliação da qualidade.

Outro ponto de vista encontrado em alguns trabalhos presentes na literatura, como por exemplo (ANACLETO, 2003), (ANDRADE, 2003) apresentam que existem dificuldades para aplicação de mensuração em pequenas e microempresas,

devido a falta de abordagens adaptadas às suas necessidades, em função da dificuldade de aplicação de abordagens como GQM e PSM a estes contextos, devido a sua complexidade. Neste sentido, os sistemas de medição deveriam ser flexíveis e facilmente adaptáveis às características de cada organização, sendo elas grandes ou pequenas.

Em relação aos trabalhos apresentados, o que percebe-se é a carência nos modelos propostos de maior suporte ao processo de mensuração, bem como sua integração ao processo de desenvolvimento, sendo que os mesmos enfatizam de forma mais ampla a infra-estrutura relacionada às métricas e demais elementos relacionados. Outro ponto pouco enfatizado é a avaliação do programa de mensuração, que pode proporcionar resultados em relação a qualidade deste processo.

Entende-se que a possibilidade de definição de métricas específicas para cada ambiente, com base nas reais necessidades dos mesmos pode favorecer a construção de programas de mensuração mais adequados a cada contexto, permitindo inclusive sua evolução progressiva. Os aspectos relacionados ao processo de mensuração também são importantes em função de uma melhor coordenação das atividades de mensuração, inclusive de forma integrada ao processo de desenvolvimento. A avaliação deste processo também deve ser considerada.

Considerando estes aspectos apresentados, na próxima seção apresentaremos um modelo para mensuração de processos de *software*, o APSEE-*Metrics*, o qual está inserido no contexto do projeto Prosoft.

4 O MODELO APSEE-*METRICS*

Nesta seção será apresentada um modelo para mensuração em processos de *software*, chamado APSEE-*Metrics*, desenvolvido com base em um estudo realizado sobre as principais práticas existentes na literatura, experiências de sucesso e também levando em consideração algumas recomendações e práticas reconhecidas, desenvolvidas por especialistas na área de mensuração. Esse trabalho está inserido no contexto do grupo de pesquisa Prosoft, no qual o ambiente APSEE, um PSEE que suporta processos de *software* centralizados foi desenvolvido (REIS, 2003).

Inicialmente são apresentados os objetivos gerais do modelo proposto. A partir dos objetivos são descritas as características principais do meta-modelo APSEE, utilizado como base para os estudos aqui apresentados. Posteriormente são descritas as principais características do modelo, por meio de uma visão geral de seus principais componentes.

4.1 Objetivos

Uma das principais dificuldades apontadas pela maioria dos especialistas na área de mensuração de *software*, como destacado nas seções anteriores, refere-se a dificuldade de se conduzir de forma organizada e objetiva um programa de mensuração em organizações desenvolvedoras de *software*. Segundo Briand (2002), uma abordagem disciplinada para definição de medidas pode permitir:

- criar sobre uma sólida base teórica;
- vincular as medidas às aplicações;
- prover uma razão esclarecedora por detrás da definição das medidas e suas aplicações;
- julgar o que é necessário para definir novas medidas ou reutilizar medidas existentes para uma aplicação específica; e
- interpretar os resultados de um experimento ou estudo de caso, especialmente quando não se obtém os resultados esperados.

Além destes aspectos, as motivações apresentadas no capítulo 1, bem como os requisitos apresentados na seção 2.7, bem como a necessidade de ferramentas que possibilitem auxílio na condução dos procedimentos necessários para mensuração,

vislumbram a possibilidade da elaboração de um modelo para mensuração, que possibilite o atingimento das principais necessidades apontadas neste estudo.

Sendo assim, o modelo proposto neste trabalho tem como principal objetivo: *prover suporte ao planejamento e execução de programas de mensuração em processos de software,* envolvendo atividades como:

- definição das métricas apropriadas ao contexto de cada projeto;
- modelagem do processo de mensuração;
- coleta, visualização e análise das métricas obtidas;
- avaliação do programa de mensuração;
- integração destas atividades com o processo de desenvolvimento.

Estes objetivos aqui apresentados foram estabelecidos no sentido de minimizar alguns dos principais problemas apresentados nas seções anteriores, no que se refere a definição e execução de programas de mensuração em organizações desenvolvedoras de *software*, levando em consideração o atendimento de alguns dos requisitos apontados na seção 2.7.

Assim, com base nos objetivos acima, foi definido um modelo para mensuração em processos de *software*, contemplando tanto o processo de mensuração quanto o modelo de informação para apoiar o gerenciamento das medidas oriundas deste processo. A abordagem proposta neste trabalho prevê a definição de um modelo de mensuração de forma integrada a um ambiente de engenharia de *software* centrado em processos (PSEE)[2], o APSEE, descrito na próxima seção.

4.2 O ambiente para automação de Processos de *Software* – APSEE

Nesta seção será apresentado de forma resumida o ambiente APSEE, um meta-modelo unificado para automação da gerência de processos de *software*, o qual serviu de base para o modelo aqui proposto.

O ambiente APSEE é uma infra-estrutura de *software* que evoluiu a partir de um mecanismo de gerência de processos de *software* proposto em (REIS, 1998). Segundo Reis (2003), o modelo do Gerenciador de Processos, tinha como principal objetivo definir um meta-modelo para a execução automatizada de processos de *software*.

Atualmente, sua arquitetura evoluiu, sendo introduzidos novos recursos para apoiar de forma mais adequada as características evolucionárias e dinâmicas do processo de *software*, dando origem ao modelo chamado APSEE (REIS, 2001).

À partir de um dos trabalhos futuros sinalizados pela definição do APSEE (REIS, 2003), o modelo de mensuração aqui proposto foi construído, visando a evolução do APSEE no que diz respeito ao gerenciamento de métricas neste ambiente.

---

[2] PSEE - *Process-centered Software Engineering Environment*

A figura 4.1 apresenta os principais componentes do APSEE, organizados em três níveis (interação com usuário, mecanismos para gerência de processos e meta-modelo), os quais são descritos nas subseções a seguir, sendo que maiores detalhes relativos às características e relacionamentos entre os componentes do modelo podem ser encontrados em (REIS, 2003).

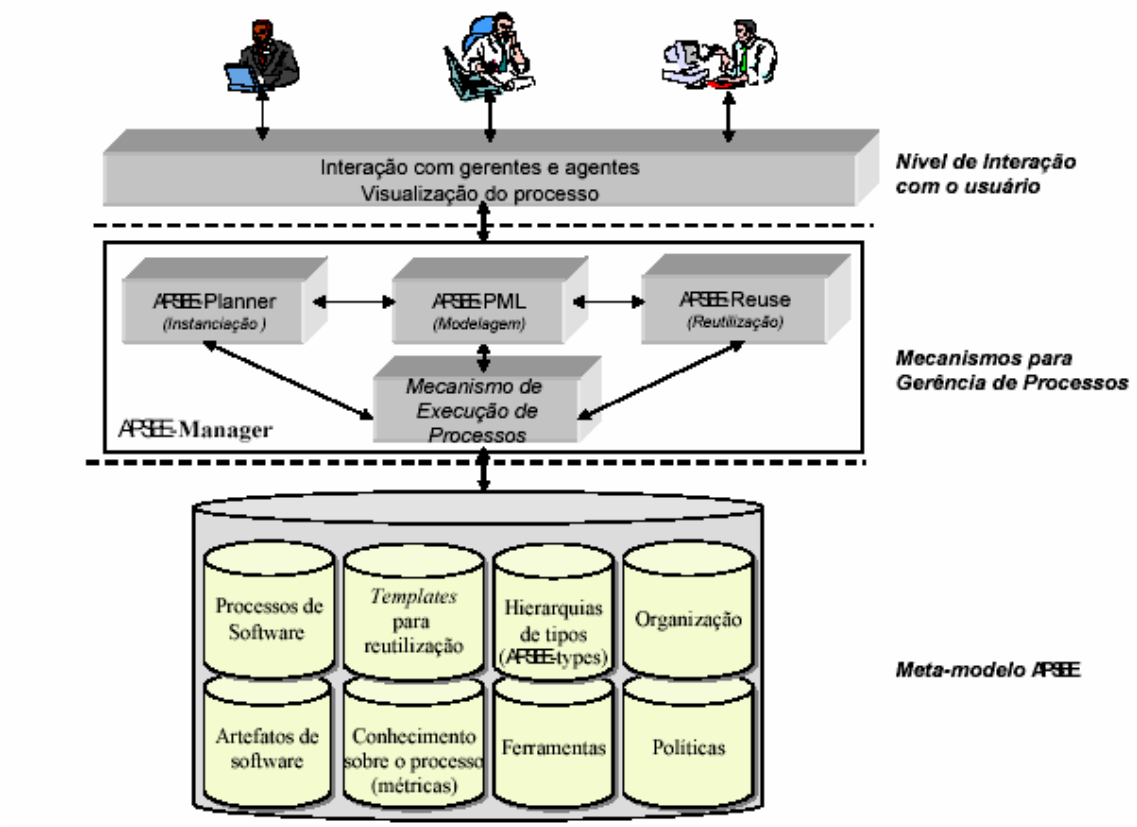


Figura 4.1: Visão geral do modelo APSEE (REIS, 2003)

4.2.1 Interação com o usuário

A camada de interação do meta-modelo APSEE é responsável pela coordenação da interação entre o usuário e o sistema através da tradução das ações do usuário pela ativação de uma determinada funcionalidade do sistema, também permitindo que os resultados destas ações possam ser observados.

É através desta camada que o usuário se comunica com o sistema, avaliando o estado de um determinado processo, percebendo a evolução dinâmica das entidades que compõem todo o processo e intervindo quando necessário.

Este componente provê mecanismos de interação especializados para os diferentes usuários do ambiente. Por exemplo, agendas para os agentes e facilidades de visualização de processos para projetista/gerentes, conforme proposto em (SOUSA, 2004).

4.2.2 Mecanismo para Gerência de processos

A camada de mecanismos para gerência de processos descreve um conjunto de serviços que constituem o componente denominado APSEE-*Manager,* existente na

implementação atual do sistema. Abaixo são listados os mecanismos para gerência de processos apresentados na figura 4.1:

- **APSEE-PML**: consiste em uma linguagem para modelagem de processos que permite definição de instâncias de processos de *software* e seus relacionamentos com os outros componentes do modelo;
- **Mecanismo de Execução do Processo**: coordena as atividades do processo em execução através da interpretação da linguagem de modelagem de processos APSEE-PML;
- **APSEE-*Planner***: auxilia na instanciação de processos de *software* através do uso de Políticas de Instanciação definidas pelo usuário;
- **APSEE-*Reuse***: componente de apoio à reutilização de processos responsável pela criação, recuperação e adaptação de *templates* reutilizáveis.

O mecanismo para execução de processos constitui a base que interpreta os modelos de processos descritos com o editor da APSEE-PML. Deste modo, o mecanismo de execução é responsável por fornecer dados sobre a dinâmica da execução de processos que são úteis para o mecanismo de instanciação APSEE-*Planer* (REIS, 2002).

A APSEE-PML auxilia a descrição de processos e *templates*, sendo também útil para habilitar políticas em modelos de processos de *software* (REIS, 2002).

No contexto deste trabalho, os componentes acima serão fundamentais, uma vez que a execução de atividades ligadas à mensuração, poderão ser modeladas e interpretadas segundo as regras definidas pelo próprio ambiente, de forma integrada ao processo de desenvolvimento.

4.2.3 O Meta-Modelo APSEE

O meta-modelo APSEE, camada mais inferior da figura 4.1, apresenta alguns dos tipos de dados mais importantes utilizados pelo ambiente definidos em (REIS, 2003), como veremos à seguir:

Componentes do meta-modelo:

- **Processos de *Software***: referem-se aos modelos de processo de *software* em seus diferentes estados (modelados – em nível abstrato ou instanciado, e em diferentes estados de execução);
- **Templates para reutilização**: descrevem os processos abstratos que podem ser reutilizados em diferentes contextos;
- **Hierarquias de Tipos** (*APSEE-Types*): são relacionadas aos componentes do APSEE e utilizadas na descrição de processos abstratos e de reutilização, além de permitir o raciocínio sobre elementos do processo de forma genérica;
- **Organização:** incorpora o modelo de recursos de apoio utilizados pelas atividades, o modelo de pessoas da organização (agentes), suas habilidades, afinidades, cargos e grupos de trabalho;

- **Artefatos de *software*:** correspondem aos itens de dados manipulados, criados e utilizados durante o desenvolvimento de *software*. Exemplos de artefatos são documentos e código;
- **Ferramentas:** correspondem a informações sobre as ferramentas disponíveis no ambiente para realização das atividades;
- **Políticas:** são regras definidas pelo usuário que permitem estabelecer quando um modelo de processos está correto (estáticas), como atividades devem ser instanciadas (de instanciação) e que ações realizar na ocorrência de eventos durante execução (dinâmicas). Políticas podem ser habilitadas na organização, em processos ou atividades específicas.
- **Conhecimento sobre o Processo:** é o componente que permite definir e armazenar métricas e estimativas para os componentes do processo, as quais podem ser consultadas dinamicamente durante a execução do mesmo.

4.3 Visão geral do modelo APSEE-*Metrics*

Nesta seção será apresentada uma visão geral do modelo APSEE-*Metrics*, um modelo para mensuração em processos de *software*. Esta definição é resultado de um estudo realizado sobre práticas referentes a mensuração em processos de *software*, onde foi identificado um conjunto de requisitos, apresentados na seção 2.7, os quais serviram de base para o modelo.

Como premissa básica para construção deste modelo, levou-se em consideração que, durante a execução de um processo de *software,* a utilização efetiva de um programa de mensuração pode proporcionar melhor entendimento dos processos adotados, dando suporte à avaliação do processo de desenvolvimento, além de proporcionar embasamento para identificação dos problemas sistemáticos encontrados nos produtos e nos processos.

Assim, a fundamentação do modelo está baseada na utilização de programas de mensuração, observando-se os requisitos apresentados na seção 2.7. Para atingir os objetivos propostos, a abordagem proposta neste trabalho procura utilizar algumas das práticas consagradas na literatura e na indústria para seleção e análise das métricas apropriadas a cada contexto.

No que tange a etapa de escolha das métricas, o modelo foi inspirado nas práticas propostas pelo PSM – *Practical Software and System Measurement*, proposto pelo Departamento de Defesa e Forças armadas dos EUA e apresentado na seção 2.6.3. Esta escolha deu-se em função de ser uma abordagem já consagrada e bastante evoluída em relação às demais técnicas apresentadas na seção 2.6, sendo a mesma amplamente utilizadas pela comunidade científica.

De maneira geral, o modelo proposto define um conjunto de recursos necessários para operacionalização das atividades de planejamento e execução de um programa de mensuração, além de um repositório de informações. Este repositório irá possibilitar o armazenamento das informações definidas, coletadas e analisadas no processo de mensuração, bem como do próprio processo, possibilitando relacionar as experiências obtidas com a mensuração nos projetos

realizados pela organização, servindo também como base para outras atividades importantes relativas à gerência do processo de *software*, tais como avaliação, adaptação, ou ainda a própria avaliação do programa de mensuração.

Uma das principais características propostas por este modelo é a integração das atividades de relacionadas ao processo de mensuração à um ambiente de desenvolvimento de *software* (ADS). O ambiente escolhido para esta integração é o ambiente APSEE, apresentado na seção anterior. Na seção seguinte serão apresentados os detalhes relativos ao modelo e sua integração com o APSEE.

4.4 Evolução do meta-modelo APSEE

Nesta seção será apresentada uma descrição do APSEE-*Metrics*, bem como a sua integração com o APSEE. Como veremos à seguir, os estudos realizados para o desenvolvimento deste modelo resultaram na evolução da arquitetura do APSEE original.

Na evolução proposta para o ambiente, o APSEE-*Metrics* foi integrado à camada de Gerência de Processos. Além disto, houve a necessidade de evolução do meta-modelo do APSEE, no que diz respeito ao gerenciamento da medições realizadas no processo de desenvolvimento. Na figura 4.2 podemos visualizar esta evolução, na qual os itens destacados em marrom na figura abaixo denotam os acréscimos propostos ao meta-modelo APSEE.

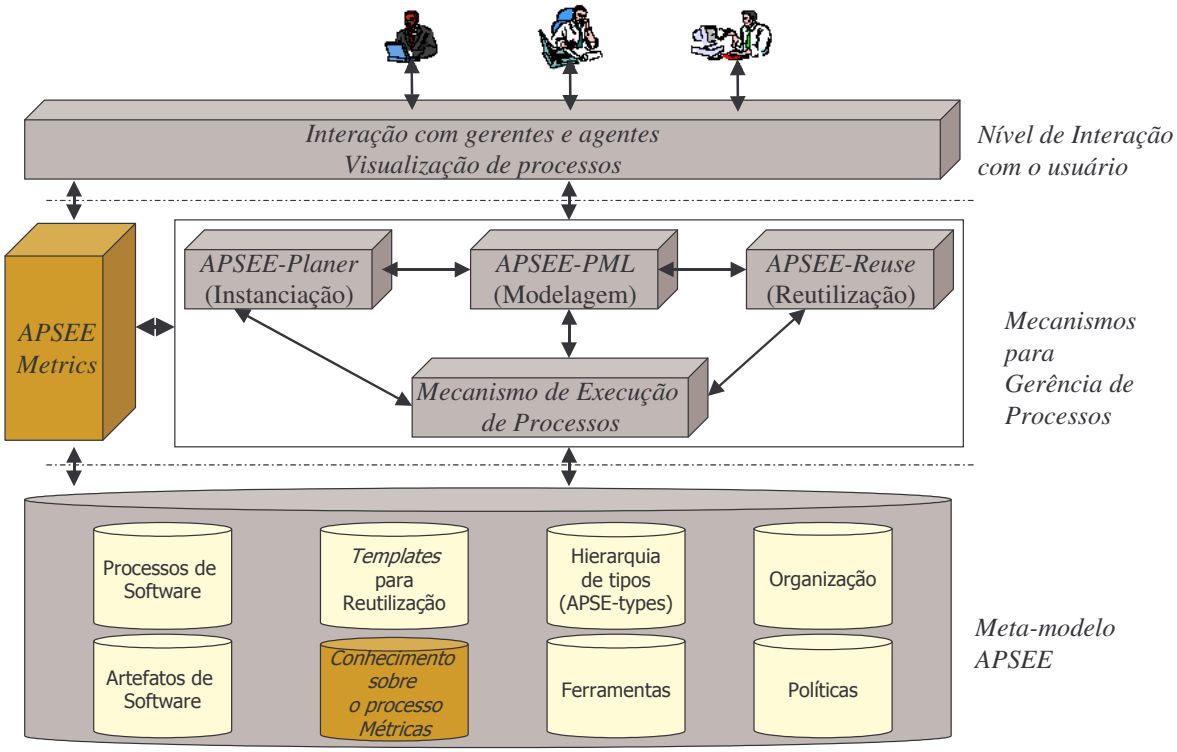


Figura 4.2: Evolução do meta-modelo APSEE

Como vimos na figura 4.2, está sendo integrado à camada de gerenciamento de processos o componente APSEE-*Metrics*, que é responsável pelo gerenciamento das atividades de mensuração, de forma integrada ao ambiente APSEE.

Em relação ao meta-modelo apresentado na figura 4.2 (parte inferior), o componente ***Conhecimento sobre o Processo***, foi originalmente definido para permitir a definição e armazenamento de métricas e estimativas para os componentes do processo, as quais podem ser consultadas dinamicamente durante a execução do mesmo. A evolução deste componente em relação as métricas está sendo proposta, de forma a permitir a definição e armazenamento de programas de mensuração de uma forma mais abrangente.

Nas seções que seguem, serão apresentados em detalhes os componentes do meta-modelo que estão sendo modificados/inseridos, sendo eles o componente de gerenciamento das mensurações APSEE-*Metrics*, e o repositório de informações.

4.5 Componente para gerenciamento das mensurações APSEE-*Metrics*

Esta seção descreve o componente proposto para gerenciamento das atividades de mensuração no ambiente APSEE, além da utilização do meta-modelo APSEE para definição e execução das atividades previstas nesta abordagem.

Buscando o atendimento das principais atividades relativas a mensuração em processos de *software,* o APSEE-*Metrics* foi organizado logicamente em quatro sub-componentes principais: Planejamento, Coleta, Análise e Controle, e Avaliação, apresentados na figura 4.3, e descritos em maiores detalhes na seqüência.

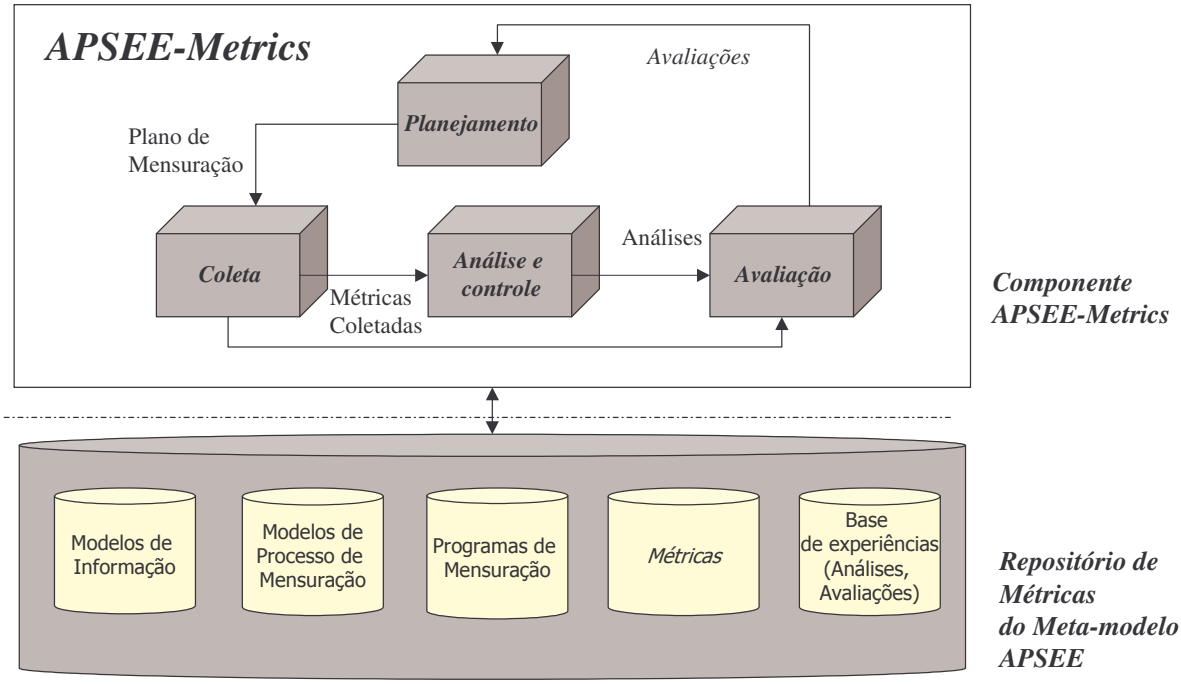


Figura 4.3: Arquitetura do APSEE-*Metrics*

Planejamento: responsável pelo suporte ao planejamento das atividades de mensuração, abrangendo desde a definição do processo de mensuração até a definição do modelo de informação a ser utilizado, que contempla basicamente a

definição e especificação das métricas a serem utilizadas e dos procedimento de coleta e de análise dos resultados, resultando em um plano de mensuração.

Coleta: responsável pelo suporte a coleta dos dados das métricas definidas na etapa de planejamento, utilizando como base o plano de mensuração.

Análise e controle: visa suportar a visualização das informações coletadas, bem como o registro das análises realizadas e ações efetuadas com base nestas análises.

Avaliação: compreende o armazenamento das experiências obtidas com a mensuração, na forma de avaliação das métricas utilizadas, das atividades do processo de mensuração, além do próprio plano de mensuração como um todo, proporcionando um *feedback* sobre o programa de mensuração utilizado. As informações providas servirão de apoio para novos planejamentos relativos às mensurações.

De forma simplificada, a sistemática de mensuração se inicia quando o gerente do processo define um ***plano de mensuração***, definindo as informações relativas ao contexto do projeto, do processo de desenvolvimento utilizado e informações relativas à organização onde o processo está inserido. Com base nestas informações, o gerente pode então escolher um ***programa de mensuração*** mais adequado às necessidades do projeto, ou então definir um novo programa de mensuração. Fazem parte do programa de mensuração o modelo de informação e o processo de mensuração.

No ***modelo de informação*** são definidos os ***objetivos*** da mensuração, e as ***informações críticas do projeto*** e, baseado no modelo ICM (*Issue-Category-Measures*) proposto pelo PSM, o gerente pode derivar e especificar as ***métricas*** necessárias para prover as informações relativas aos objetivos definidos. Também são definidas as ***análises*** a serem realizadas visando o acompanhamento do projeto, bem como as ***avaliações*** a serem realizadas ao final do processo. Para definição do modelo de informação o gerente poderá utilizar também as experiências anteriores armazenadas no repositório.

No ***processo de mensuração*** são definidos os procedimentos (atividades) de planejamento, coleta, análise e avaliação das métricas. Além disto, as atividades do processo de mensuração são integradas ao processo de desenvolvimento, sendo modeladas através dos recursos providos pelo ambiente APSEE para gerenciamento de atividades em um processo de *software*.

Uma das premissas para a utilização da abordagem proposta é que a organização possua um modelo processo de desenvolvimento de software definido, para que o mesmo possa ser integrado ao processo de mensuração, através da introdução das atividades necessárias para realizar cada uma das etapas do processo de mensuração.

Após o planejamento, com base no plano de mensuração, a execução do plano é então seguida, executando-se o processo de mensuração. Conforme as atividades previstas no processo, os dados necessários para obtenção das métricas definidas são ***coletados*** e ***armazenados***. Além disto, o gerente poderá ainda realizar o ***acompanhamento*** das métricas obtidas e, com base nestas informações, executar

as ***análises*** previstas no processo de mensuração, armazenando as análises, as ações tomadas e os seus resultados, proporcionando base para tomada de decisão em futuros projetos.

Por fim, ***avaliações*** das métricas utilizadas, das atividades do processo de mensuração e do programa como um todo são realizadas e armazenadas, visando a qualificação dos programas de mensuração e verificação de sua adequação ao contexto utilizando, permitindo a criação de uma base de informações úteis para criação de novos programas de mensuração, ou de sua utilização em contextos similares.

Nas próximas sub-seções serão apresentados cada um dos sub-componentes de forma mais detalhada.

4.5.1 Planejamento

O sub-componente de planejamento corresponde ao principal elemento do modelo. É através dele que são definidos todos os itens do plano de mensuração. A abordagem propõe o planejamento da mensuração utilizando algumas práticas do PSM (USA, 2003), apresentado na seção 2.2.6. A escolha pelo PSM foi realizada em função de tratar-se de um padrão amplamente difundido e utilizado como base para importantes padrões da indústria de *software*, tais como a norma ISO/IEC 15939 (*Software Measurement Process)* e a área de medição e análise do modelo CMMI (*Capability Maturity Model Integrated*), sendo hoje uma das principais referências em termos de padrões para mensuração.

No modelo proposto, o planejamento é conduzido pela identificação dos objetivos da mensuração e das informações críticas do projeto, onde cada uma delas é mapeada para uma área de informação do modelo ICM (*issue*, *category, measure*), proposto pelo PSM (exemplos de áreas são: cronograma, recursos e custos, etc.). Cada uma das áreas, por sua vez, prevê a organização de diferentes categorias de mensuração, que são pré-definidas no modelo (tais como: progresso das unidades de trabalho, eficiência do processo, etc.), cada qual contendo medições genéricas associadas a ela (tais como: esforço, experiência da equipe, custo, etc.), facilitando e conduzindo a seleção das métricas a serem utilizadas no plano de mensuração. Assim, a definição das métricas é realizada seguindo as seguintes etapas:

a) Identificação das necessidades de medição: a condução eficiente de um programa de mensuração depende fundamentalmente de quais os objetivos foram determinados para sua implementação. Neste sentido, a identificação e registro dos objetivos e das informações críticas do projeto são fundamentais para a construção do plano de mensuração. Estes objetivos e necessidades podem ser obtidas das informações do projeto, tais como as necessidades, restrições, estratégias técnicas, estimativas, riscos inerentes ao projeto ou quaisquer outras informações relevantes da organização ou do projeto.

Alguns exemplos de informações críticas seriam:

- Disponibilidade de pessoal;
- Níveis de satisfação dos usuários;

- Disponibilidade financeira;
- Expectativas em relação à qualidade do produto;
- Limitação de prazos de entrega, entre outros.

Definidos os objetivos e as informações críticas do projeto, estas são então mapeadas para as Áreas Comuns de Informação do PSM (*Common Issues*), de forma a classificá-las e organizá-las de uma maneira uniforme, auxiliando também na identificação das métricas apropriadas para cada objetivo específico. As áreas previstas pelo PSM são:

- Cronograma e progresso;
- Recursos e custos;
- Tamanho e estabilidade do produto;
- Qualidade do produto;
- Performance do processo;
- Eficácia de Tecnologia;
- Satisfação do cliente.

No modelo aqui proposto, estas informações deverão ser identificadas pelo gerente do projeto, sendo armazenadas em um repositório. Com relação às áreas propostas pelo PSM, as mesmas serão utilizadas como guias para condução da seleção das métricas, porém, será permitido ao gerente customizar e criar novas áreas mais apropriadas às necessidades de cada contexto.

Após o mapeamento, o gerente deve determinar as prioridades de cada objetivo. As informações críticas do projeto devem ter suas probabilidades e impacto definidos, de maneira que as análises dos dados possam ser conduzidas de forma orientada aos objetivos mais relevantes de cada projeto, levando em consideração as informações críticas do projeto.

b) Definição das métricas: com base nas necessidades de medição apontadas na etapa anterior, as informações de projeto, e as áreas às quais foram mapeadas, a definição das métricas apropriadas pode ser conduzida. O modelo prevê que, para cada informação crítica de projeto mapeada a uma área específica do ICM, possa ser derivada uma categoria de medição aplicável a ela, ou seja, são verificadas as categorias de medição mais apropriadas para a obtenção das medições genéricas, da qual poderão ser derivadas as métricas à serem utilizadas.

Desta forma o gerente terá condições de selecionar os dados a serem coletados às partir destas derivações, selecionando as métricas mais apropriadas para proporcionar as informações necessárias para suas análises.

A figura 4.4 ilustra este mapeamento das necessidades informações do projeto às áreas do modelo ICM, onde as informações específicas do projeto são relacionadas às áreas comuns do modelo ICM. Estas por sua vez se relacionam com as categorias de medição, cada qual possuindo um conjunto de medidas genéricas associadas a ela. No anexo D deste trabalho encontra-se uma tabela que apresenta as áreas e medidas do modelo ICM.

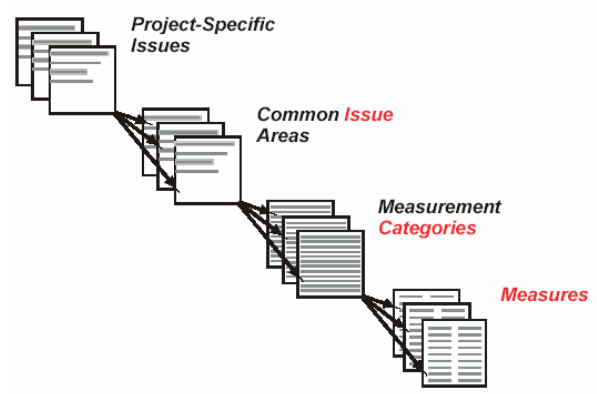


Figura 4.4: Mapeamento das necessidades de medição através do modelo ICM (USA, 2003)

Um exemplo simples desta interação seria a criticidade de informações referentes ao prazo de entrega do *software*. Utilizando o modelo proposto, poderíamos definir as medidas necessárias derivando a necessidade de informação com base na área de Cronograma e Progresso do modelo ICM. À partir desta área, poderíamos derivar as categorias de medição específicas que são pré-definidas no modelo, onde podemos obter, por exemplo, a categoria de Atingimento de Marcos de Projeto. Como cada categoria possui medições genéricas apropriadas para as necessidades de informação, o gerente pode selecionar a medição apropriada para cada caso. Neste caso, por exemplo, poderia ser a data de atingimento dos marcos de projeto.

c) Definição e especificação das métricas: após selecionadas as medições genéricas apropriadas, o gerente pode então definir as métricas e os dados a serem coletados. Além disto, os detalhes sobre as mesmas devem ser especificados, de forma a garantir a correta interpretação dos dados a serem coletados, para que possam ser o mais consistentes possíveis. A especificação consiste na definição dos dados a serem coletados, do tipo de medição, do método de medição, da origem dos dados, dos critérios de contagem e também o escopo da medição.

d) Definição das análises: além das métricas a serem utilizadas, também deverão ser especificadas e previamente definidas as análises a serem realizadas em relação aos dados obtidos. Esta definição prévia irá permitir que estas análises estejam de acordo com os objetivos de mensuração previamente definidos, e também promovendo um acompanhamento sistemático das informações coletadas.

e) Definição das avaliações: da mesma forma que as análises, também deverão ser previstas as avaliações a serem realizadas, permitindo que se tenha um *feedback* sobre o programa de mensuração utilizado. Deverá ser realizado tanto a avaliação das métricas utilizadas (se foram adequadas, se houveram problemas na coleta, se trouxeram os resultados esperados, etc.), quanto das atividades definidas (se foram eficientes, se oneraram o processo, dificuldades obtidas, etc.). Com base nestas avaliações, o programa como um todo pode ser avaliado, permitindo que sejam armazenadas as contribuições, problemas, critérios e outras informações relevantes para futuros projetos.

f) Definição do processo de mensuração: uma vez definidas e especificadas as métricas a serem obtidas, e as análises e avaliações a serem realizadas, é necessário então realizar a definição do processo de coleta, análise e avaliação destas informações. No modelo proposto, propõe-se a modelagem das atividades de mensuração de forma integrada ao processo de desenvolvimento, realizada através de uma linguagem de modelagem de processos provida pelo ambiente APSEE, a APSEE-PML.

A modelagem integrada ao processo de desenvolvimento irá permitir a visualização do processo como um todo, tendo em vista a integração destas atividades, possibilitando inclusive a verificação do aumento de tempo gasto com as atividades incluídas.

4.5.2 Coleta

O sub-componente de coleta corresponde basicamente ao registro dos dados oriundos da execução de atividades de coleta previstas no processo de mensuração.

A coleta de dados é baseada nas atividades de coleta e na definição das métricas previstas no planejamento, sendo que os responsáveis pela coleta deverão realizá-las conforme a especificação dada para cada métrica. Nos casos de métricas automáticas, que podem ser providas pelo próprio ambiente de desenvolvimento integrado, os dados poderão ser automaticamente coletados e armazenados no repositório de medições. No caso das métricas manuais, os dados deverão ser coletados e armazenados conforme previsto na especificação das métricas.

4.5.3 Análise e controle

O sub-componente de análise e controle compreende funcionalidades para registro e visualização dos dados oriundos das atividades de análise previstas no processo de mensuração.

Durante a execução do processo de desenvolvimento, os dados parciais das coletas realizadas podem ser visualizados e analisados, permitindo ao gerente do processo controlar a execução do mesmo através dos dados obtidos pelas medições realizadas. Ao final do processo, as análises previstas no processo de mensuração também são realizadas e registradas.

No modelo proposto, as informações relativas às análises realizadas são armazenadas no repositório, permitindo que estas informações sejam resgatadas em projetos futuros. Além disto, as ações oriundas destas análises também deverão ser armazenadas no repositório, bem como os resultados obtidos com as ações tomadas pelo gerente, sendo estas informações providas pelo gerente e registradas no repositório.

4.5.4 Avaliação

A avaliação das etapas de mensuração constitui um importante passo em um programa de mensuração. Neste modelo propõe-se a realização de avaliações referentes ao plano de mensuração, do processo de mensuração e das métricas utilizadas. As atividades de avaliação são também previstas no processo de

mensuração, e sua definição faz parte do planejamento do modelo de informação, onde são pré-estabelecidos os critérios para avaliação de cada um destes itens.

Assim, a realização das avaliações segue o planejamento realizado, sendo que o sub-componente de avaliação visa justamente o registro das avaliações realizadas e o que foi racionado durante esta etapa. Esta avaliação é armazenada em uma base de experiências, possibilitando a criação de um conjunto de informações úteis para o planejamento de novos programas de mensuração, possibilitando resgatar as avaliações de cada métrica utilizada, das atividades utilizadas e do programa em si, auxiliando também na etapa de planejamento.

4.6 Evoluções no meta-modelo geral do APSEE

Como citado anteriormente, o meta-modelo do APSEE sofreu algumas adaptações para proporcionar a infra-estrutura necessária para a definição e execução de programas de mensuração. Nesta seção serão apresentadas tais adaptações, onde serão descritos primeiramente os pacotes do modelo original que sofreram algum tipo de adaptação, e por fim o componente principal em relação a este trabalho, na qual consiste a definição do meta-modelo do APSEE-*Metrics*.

4.6.1 Meta-modelo geral do APSEE

Na seção 4.2 foi apresentada uma visão geral do meta-modelo APSEE, utilizado como base para o modelo aqui proposto. Nesta seção serão apresentados com mais detalhes alguns dos componentes do meta-modelo que foram utilizados para descrição do APSEE-*Metrics*, onde também serão apresentadas as adaptações propostas ao modelo. O diagrama é definido segundo a notação UML[3] (OMG, 2004).

A Figura 4.5 apresenta o diagrama de pacotes do meta-modelo APSEE, ilustrando os seus componentes e seus relacionamentos de dependência. Alguns componentes do meta-modelo são refinados em novos pacotes. Isto ocorre com os pacotes *Organization*, *Policies* e *SoftwareProcesses*.

---

[3] O meta-modelo APSEE tem servido de base para construção de ferramentas em um projeto que envolve cooperação internacional (PROSOFT). Assim, os diagramas correspondentes são apresentados em inglês.

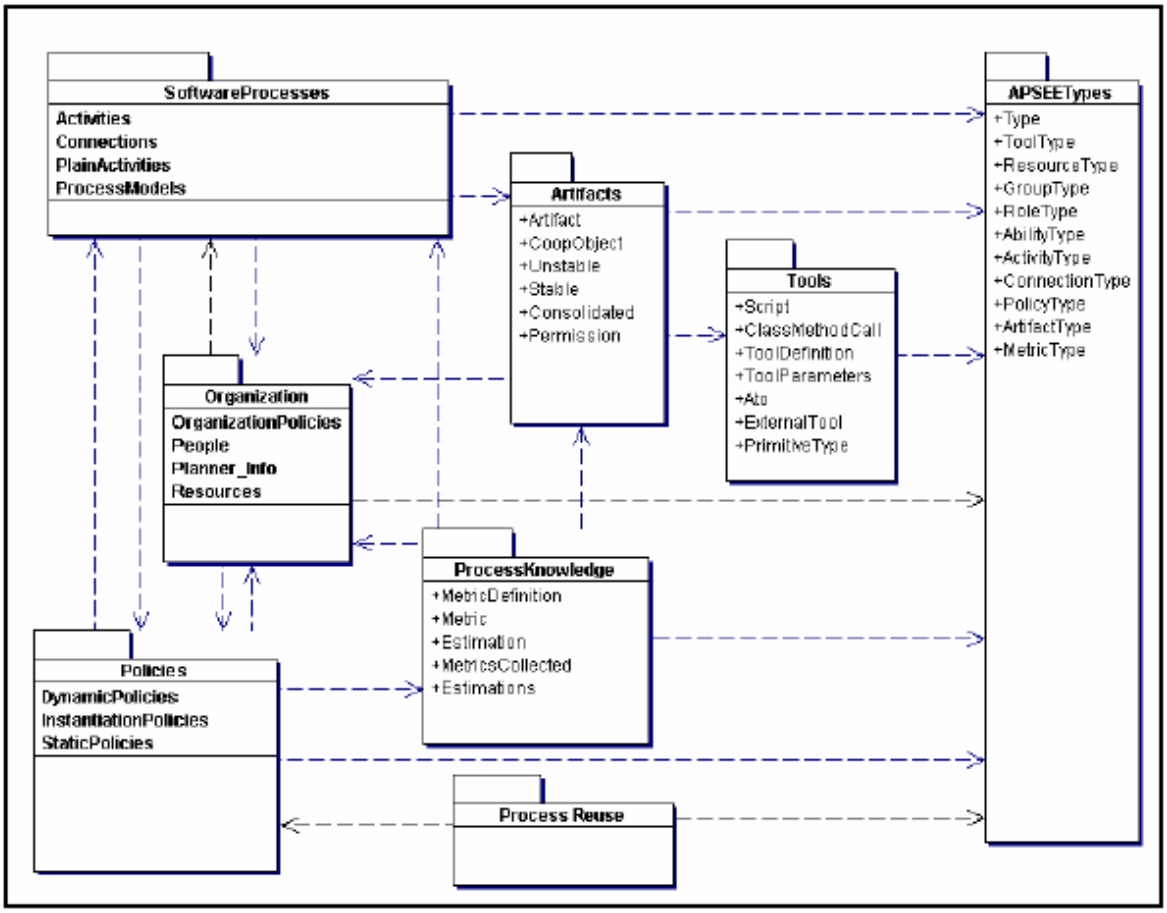


Figura 4.5: Meta-modelo APSEE representado através de pacotes UML

(REIS, 2003)

4.6.2 Componente *ProcessKnowledge*

O componente *ProcessKnowledge* do APSEE (figura 4.6) funciona como uma base de conhecimento sobre todos os componentes do meta-modelo e foi integrado para ser consultado nas fases de modelagem, instanciação, simulação e execução de processos, além de permitir estudos empíricos acerca do impacto de algumas métricas em aspectos importantes, como por exemplo, na qualidade do *software* produzido.

Este trabalho propõe a inclusão de novos elementos ao componente *Processknowledge*, elementos estes que permitirão o gerenciamento de forma mais completa das mensurações realizadas no ambiente APSEE. A figura 4.6 apresenta a definição original deste componente, que terá a introdução de um novo pacote *MeasurementManagement*, responsável por este gerenciamento.

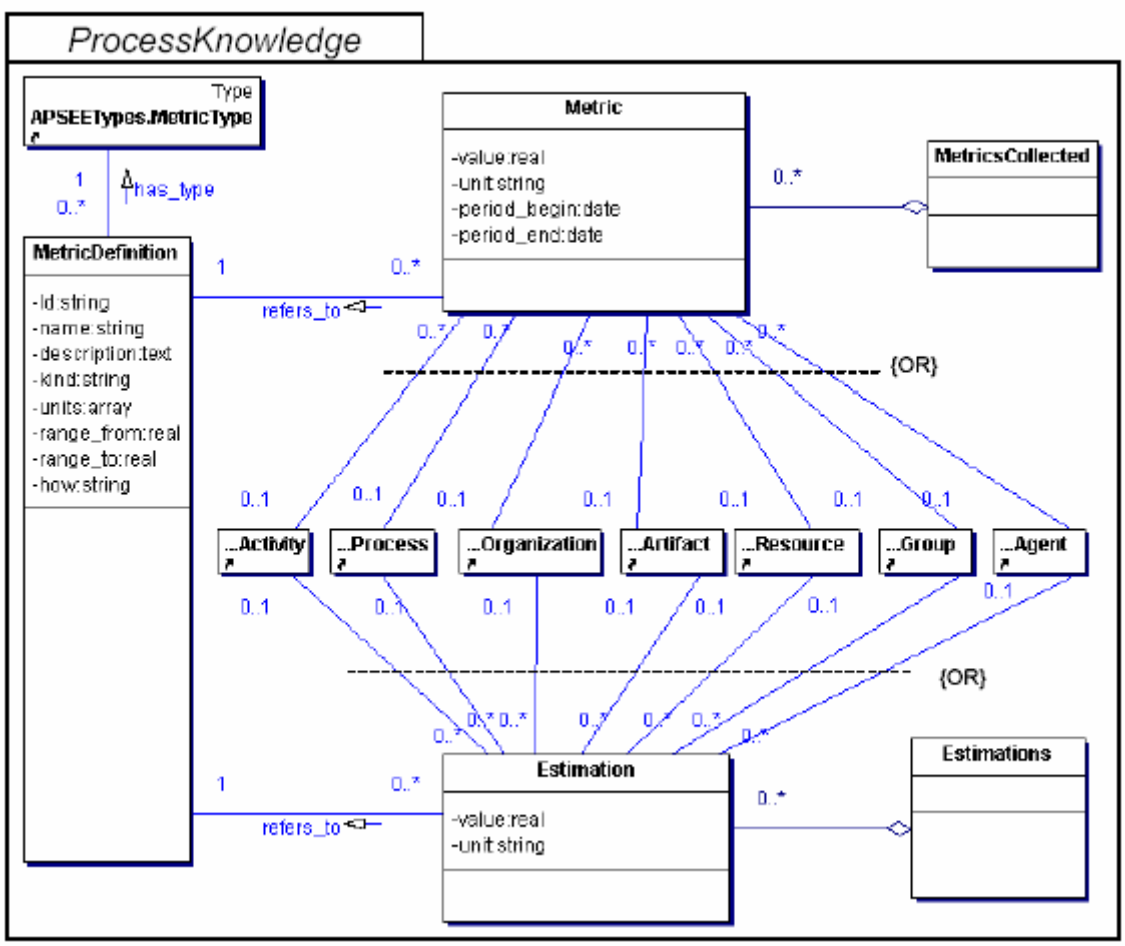


Figura 4.6: Pacote *ProcessKnowledge*

(REIS, 2003)

De acordo com a definição original do pacote (REIS, 2003), a classe *MetricDefinition* permite definir métricas importantes para o processo e para a organização enquanto que as classes *MetricsCollected* e *Estimations*[4] armazenam instâncias de métricas associadas aos componentes do modelo. As características principais sobre as instâncias da classe *MetricsDefinition* são:

- Possuem um identificador único, estão associadas a um tipo de métrica em APSEE-*Types*, possuem um nome e uma descrição textual;
- O tipo de métrica sendo definida (*Kind*): podendo ser de processo, de produto ou individual;

  – **Métrica de Processo**: indica uma característica de um processo, fragmento de processo ou atividade. Por exemplo, esforço de um processo, tempo total do processo, percentual de atividades atrasadas, percentual de atividades concluídas, dentre outros;

[4] O presente trabalho não abrange as estimativas de projeto ou processo, pois estão fora do escopo estabelecido para o trabalho, sendo que as definições relativas à este âmbito de estudo podem ser encontradas na definição original do APSEE em (REIS, 2003).

– **Métrica de Produto**: indica uma característica de um artefato produzido ou consumido por atividades. Podem ser métricas de tamanho, por exemplo, pontos por função, ou de qualidade, por exemplo, defeitos por ponto por função;

– **Métricas Individuais**: indicam características de pessoas (agentes e grupos) e recursos envolvidos no processo. Estas métricas também podem se aplicar a cargos ou à organização como um todo. Podem ser definidas métricas que representam a produtividade de agentes, quantidade de defeitos produzidos, dentre outros.

- Unidades de medida da métrica (*unit*). Cada métrica pode ter várias unidades de medida. Por exemplo, a métrica "Tamanho" pode ser medida por "linhas de código" ou "pontos por função". Entretanto a definição de métricas é livre e pode ser criada uma métrica diferente para tamanho em linhas de código e outra para tamanho em pontos por função;
- Intervalo de validade da métrica (*range_from* e *range_to*). Cada métrica definidas poderá possuir um intervalo de medida que delimitará os valores mínimo e máximo a serem informados para a mesma;
- Como a métrica pode ser obtida (*how*). Neste campo pode ser registrado o método utilizado para medir, como por exemplo, "cálculo automático", "contagem manual" ou "chamada de operação de uma ferramenta", dentre outros.

As métricas coletadas na classe *MetricsCollected* contém as instâncias de métricas que estão associadas a definições de métricas (*MetricDefinition*). Cada métrica possui as seguintes características:

- Uma associação indicando a definição da métrica em *MetricDefinition*;
- Uma associação indicando o elemento medido. Neste caso, o elemento pode ser um artefato, um processo, uma atividade, um recurso, um agente, um grupo ou toda a organização. Deve-se observar que o elemento medido deve ser compatível com o tipo de métrica definido pela associação com *MetricDefinition*;
- O valor medido (*value*);
- A unidade de medida (*unit*), que deve ser uma unidade válida para a métrica;
- O período da medição (*period_begin, period_end*). Em alguns casos, pode-se medir várias vezes o mesmo elemento do processo variando o período da medição.

4.6.3 Componente *Measurement Management*

À partir destas definições preliminares, o modelo de gerenciamento de mensuração do APSEE-*Metrics* foi construído de forma integrada ao meta-modelo do APSEE, com a incorporação do pacote *MeasurementManagement*. A figura 4.7 apresenta o diagrama de classes em UML referente ao pacote.

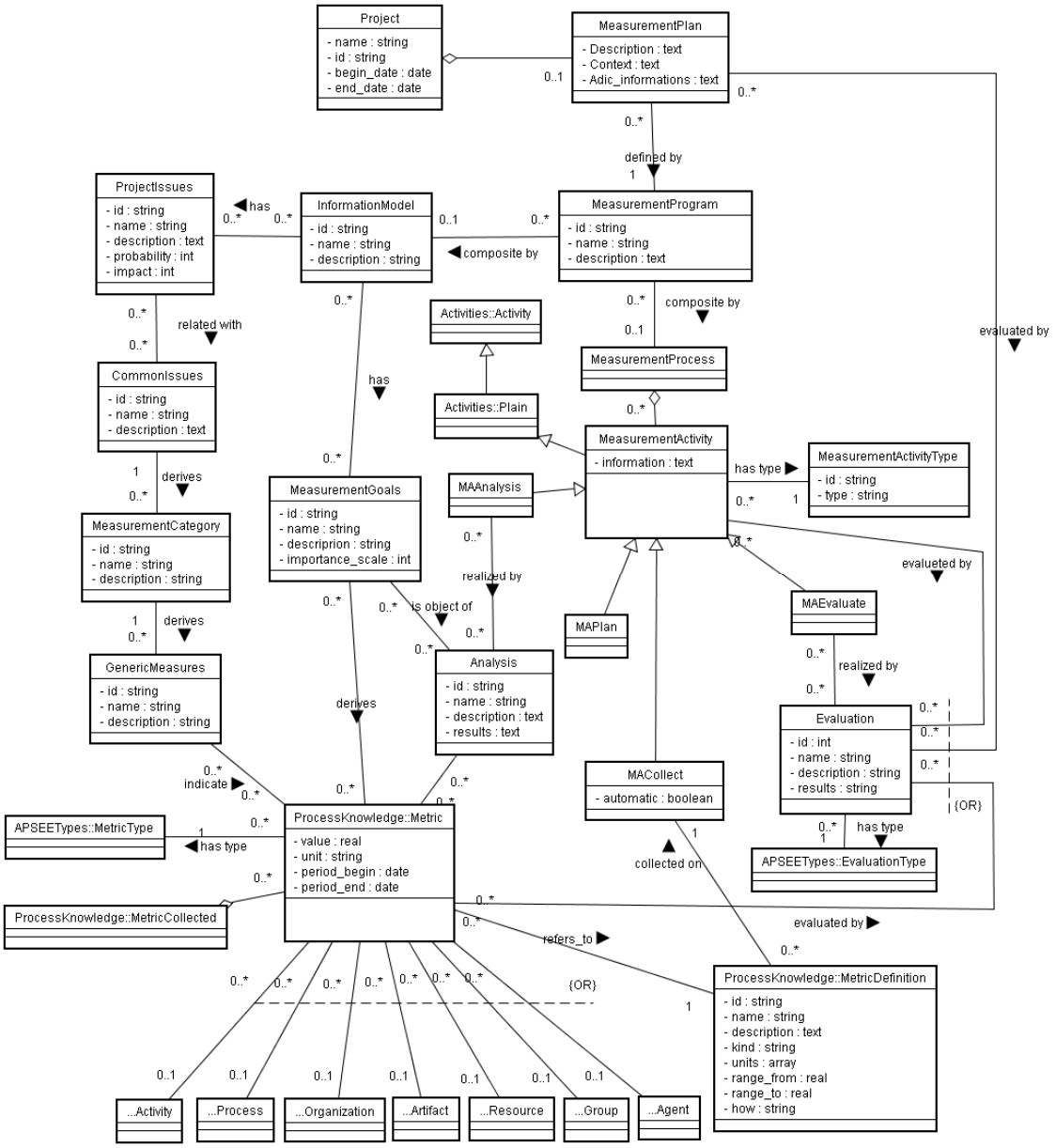


Figura 4.7: Pacote *MeasurementManagement*

O pacote de gerenciamento de mensurações (figura 4.7), permite a definição e armazenamento de programas de mensuração para processos de desenvolvimento de *software*. Através de seus componentes, o gerente poderá realizar a definição de um plano de mensuração para cada projeto, realizando a definição das métricas apropriadas para cada contexto, bem como a definição do processo de mensuração. A descrição e os relacionamentos entre as classes do modelo são descritos da seguinte forma:

Primeiramente, o gerente do processo precisará definir um **programa de mensuração** (*MeasurementProgram*), definindo para o mesmo um identificador, um nome e a descrição do mesmo, além do contexto ao qual o programa poderá ser aplicado. Esta informação será bastante útil para permitir a reutilização do programa em outros projetos que sejam relacionados à contextos similares. Cada programa de mensuração é composto por um modelo de informação (*InformationModel*) e um processo de mensuração (*MeasurementProcess*).

O **modelo de informação** corresponde basicamente a definição das informações necessárias para a correta execução da mensuração, e é composto por:

- Objetivos da mensuração (*MeasurementGoals*): correspondem a uma das principais informações do modelo, na qual o gerente deverá definir as necessidades de informação do programa de mensuração. Os objetivos são compostos por um identificador, um nome e uma descrição dos objetivos e resultados esperados com a mensuração de cada objetivo. Além disto, cada objetivo pode ser priorizado, através de uma escala numérica que indica sua relevância em relação aos demais.

- Informações críticas do projeto (*ProjectIssues*): uma vez obtidos os objetivos de mensuração, são definidas as informações críticas específicas do projeto (riscos, critérios, necessidades críticas do projeto, etc.), onde cada uma é mapeada para uma área comum de informação do modelo ICM (*CommonIssues*). Cada informação crítica do projeto possui um identificador, um nome, uma descrição com detalhes sobre a informação, e também a probabilidade de ocorrência da mesma, bem como o impacto da ocorrência da mesma em relação ao projeto, expressas em uma escala numérica.

- **Modelo ICM**: no modelo ICM (descrito na seção 2.2.6), são definidas algumas **áreas comuns** que podem ser utilizadas como guia para definição da métricas apropriadas. Estas áreas são representadas neste modelo pela classe *CommonIssues,* que é definida por um identificador, um nome e uma descrição, na qual poderá ser utilizada para entendimento de cada área. Cada uma destas áreas por sua vez, é composta por **categorias de mensuração**, representadas pela classe *MeasurementCategory*, e definidas por um identificador, um nome e uma descrição. Cada categoria possui um conjunto de **medições genéricas** (*GenericMeasures*), também definidas por um identificador, um nome e uma descrição, sendo que estas medições genéricas são mapeadas pelo usuário aos objetivos de mensuração definidos anteriormente. Cada uma das medições genéricas do modelo ICM pode conter um conjunto de métricas, que por sua vez indicam quais os dados devem ser coletados para atender aos objetivos de mensuração definidos.

- **Métricas**: são representada pela classe *Metric*, originalmente proposta no meta-modelo do APSEE, que representam as métricas a serem obtidas durante o processo de desenvolvimento. As definições originalmente propostas para esta classe foram mantidas (descritas na seção 4.5.2). Cada métrica possui uma **definição**, representada pela classe *MetricDefinition*, e um **tipo** de métrica *MetricType*. As **métricas obtidas** são armazenadas pela classe *MetricCollected*, uma instância das métricas (*Metric).*

- **Análise**: representada pela classe *Analysis*, as análises a serem realizadas com base nas métricas coletadas durante o processo de desenvolvimento deverão ser especificadas durante o planejamento das mensurações. Cada análise é relacionada com um ou mais objetivos de mensuração definidos pelo gerente, visando também o armazenamento das experiências adquiridas durante a mensuração. A classe possui uma identificação, um nome, uma

descrição da análise a ser realizada (definida no planejamento das mensurações), e os resultados das análises realizadas.

- **Avaliações** (*Evaluation*): visando contemplar as avaliações necessária ao programa de mensuração, é proposto neste modelo a definição e armazenamento de diferentes tipos de avaliações. A classe *Evaluation* é responsável por isto, e é definida por um identificador, um nome, uma descrição da avaliação a ser realizada (definida no planejamento das mensurações). Cada avaliação prevista poderá ter um tipo específico e o resultado da análise realizada. Os tipos podem denotar:

  a) a ***avaliação das métricas*** utilizadas, indicando por exemplo, se as métricas foram apropriadas ao contexto específico, se foram medidas corretamente, ou outras informações relevantes sobre cada métrica utilizada;
  b) a ***avaliação das atividades*** de mensuração, indicando por exemplo, se as atividades definidas no processo de mensuração foram corretamente executadas, se oneraram o processo, se houveram dificuldades enfrentadas durante a sua execução, entre outras informações relevantes acerca da avalização do processo de mensuração;

  c) a ***avaliação do plano de mensuração***, indicando, em um contexto mais amplo, a eficácia do plano de mensuração, a confiabilidade dos resultados obtidos, entre outras informações relevantes acerca de todo o plano de mensuração.

O **processo de mensuração** representado pela classe *MeasurementProcess,* corresponde a definição das atividades necessárias para contemplar as etapas de um programa de mensuração. É composto por atividades de mensuração (*MeasurementActivity*), que representam cada uma das atividades do processo de mensuração. Cada atividade corresponde a um tipo especial de atividade (*Plain Activity)*, definida no meta-modelo APSEE, sendo que a mesma além de herdar todas as características de *Plain Activity,* pode ainda ser especializada em 4 tipos (definidos por *MeasurementActivityType*):

a) *Atividades de planejamento das mensurações* (*MAPlan*): que correspondem às atividades de planejamento do programa de mensuração, como por exemplo a definição dos objetivos, especificação das métricas, etc.. Estas atividades vão resultar no modelo de informação e no processo de mensuração, que compõe o plano de mensuração do processo de desenvolvimento;

b) *Atividades de análise* (*MAAnalysis*): que correspondem às atividades de análise do programa de mensuração, definidas no plano de mensuração. Nestas atividades serão utilizadas as definições de análises feitas na etapa de planejamento e serão registrados os resultados obtidos;

c) *Atividades de coleta das métricas* (*MACollect*): que correspondem às atividades de coleta de dados das métricas definidas, estas podendo ser manuais ou automáticas;

d) *Atividades de avaliação* (*MAEvaluate*): que correspondem às atividades de avaliação do programa de mensuração, das métricas ou das atividades do processo.

Nestas atividades serão utilizadas as definições de avaliação feitas na etapa de planejamento e serão registrados os resultados obtidos;

Uma das características importantes deste modelo é justamente a possibilidade de modelagem do processo de mensuração de forma integrada ao processo de desenvolvimento. A definição do processo de mensuração como uma especialização dos processos definidos pelo meta-modelo APSEE irá permitir que o mesmo possa ser integrado diretamente ao processo de desenvolvimento, trazendo vantagens como a modelagem do processo, a visualização, e outras funcionalidades já existentes no ambiente, permitindo ao gerente do processo de desenvolvimento verificar o impacto da adoção de tais atividades no contexto geral de cada projeto.

Por fim, tendo sido definido um programa de mensuração com seus objetivos, contexto, métricas, processo, e as demais informações previstas neste modelo, podemos relacionar um projeto (classe *Project*) do meta-modelo APSSE a este programa de mensuração. Desta forma, é representado pela classe (*MeasurementPlan*) o plano de mensuração do projeto.

O **plano de mensuração** é definido pela escolha de um dos programas de mensuração armazenados no modelo, além de informações gerais referentes a descrição textual do plano (contendo informações de relevância para o gerente do projeto, tais como fatores críticos de sucesso, considerações importantes sobre o projeto, etc.), o contexto ao qual o projeto está inserido (ambiente, tecnologias utilizadas, tipo de processo, etc.), além de outras informações adicionais relevantes. Todas estas informações, servirão como base histórica para os projetos, relacionando as experiências com a utilização de mensuração no ambiente APSEE.

4.7 Considerações sobre o capítulo

Neste capítulo foi apresentado o modelo APSEE-*Metrics*, um modelo para mensuração em processos de desenvolvimento de *software*. O modelo adotado permite que o mecanismo APSEE-*Metrics* guie a definição dos programas de mensuração de forma sistemática e orientada ao objetivos do projeto, atuando de forma integrada em relação ao processo de desenvolvimento. O próximo capítulo apresenta a especificação formal dos componentes e das funcionalidades descritas neste capítulo.

5 ESPECIFICAÇÃO DO MODELO APSEE-*METRICS*

Este capítulo descreve o modelo proposto através de uma especificação algébrica. Inicialmente é apresentado uma breve introdução, que descreve de forma resumida as principais características do formalismo utilizado para a especificação. Posteriormente, é descrita a especificação do modelo, utilizando o formalismo apresentado.

5.1 PROSOFT Algébrico

De forma análoga aos demais trabalhos do grupo de pesquisa Prosoft, nesta seção será apresentada uma breve introdução ao formalismo utilizado nos trabalhos desenvolvidos pelo grupo, sendo também adotado para especificação do modelo aqui proposto.

O formalismo adotado para a especificação formal do modelo proposto é denominado PROSOFT-Algébrico (descrito em (NUNES, 1992)(NUNES, 1994)). Este formalismo permite a descrição de tipos abstratos de dados através de um paradigma algébrico baseado em objetos.

Segundo Reis (2002), esse paradigma adota uma abordagem *data-driven* para o desenvolvimento de *software*, estimulando o desenvolvimento através da composição dos tipos de dados necessários.

No PROSOFT-Algébrico, cada tipo de dados é instanciado a partir de um ATO (Ambiente de Tratamento de Objetos). Um ATO implementa um tipo abstrato de dados, e é composto por uma classe e um conjunto de operações que atuam sobre o objeto dessa classe. É importante ressaltar que o conceito de classe no PROSOFT é uma estrutura de dados que corresponde aos atributos do objeto, ou variáveis de instância nas linguagens orientadas a objeto (KÖRBES, 1996). A figura 5.1 apresenta a estrutura de um ATO, que segundo Nunes (1994) é formado por três partes:

- Classe: define uma instanciação do tipo através de uma linguagem gráfica que representa a especificação algébrica dos tipos abstratos de dados. A instanciação é uma árvore cuja as folhas são referências a outros tipos de dados e cujos nodos são tipos de dados compostos;
- Interface: especifica a funcionalidade das novas operações; e
- Operações: define a semântica das funções (axiomas) que atuam sobre o objeto.

Todo ATO possui um conjunto de objetos (que poderão somente ser manipulados pelo ATO que o originou, pois cada ATO trata apenas termos do *sort* por ele definido). Nos casos em que um ATO necessita manipular um objeto criado por outro ATO ele envia uma mensagem através da Interface de Comunicação do Sistema (ICS), para o ATO criador. Assim, o ATO receptor da mensagem procura a operação contida na mensagem, substitui os parâmetros pelos seus respectivos argumentos, aplica a operação e devolve o resultado, novamente via ICS, para o ATO emissor. A figura 5.2 apresenta a arquitetura relacionada aos ATOS e a ICS.

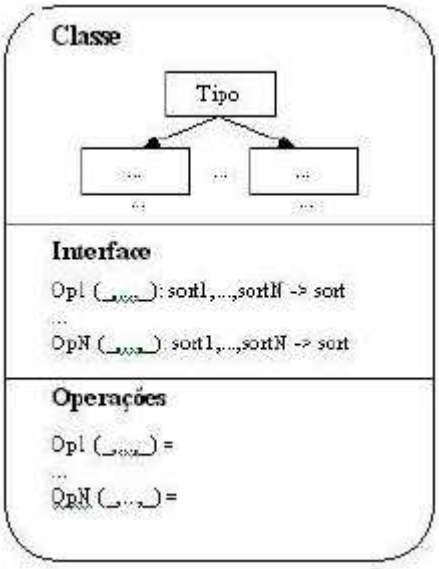


Figura 5.1: Estrutura de um ATO

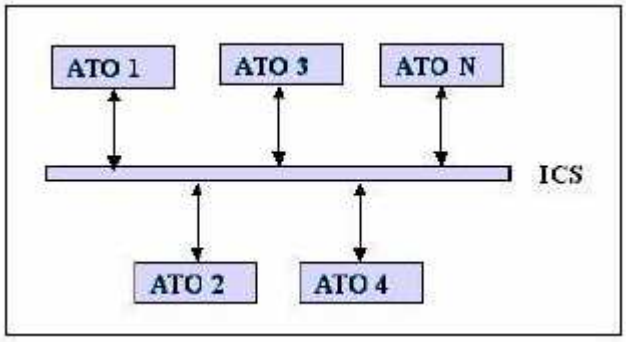


Figura 5.2: Arquitetura relacionada aos ATOS e a ICS

A ICS é o meio de comunicação entre os ATOs, sendo responsável pelo fluxo de dados e pelo controle da execução das operações. Uma ICS possui a seguinte sintaxe:

ICS(<nome do ATO>, <operação>, <seletor>, <argumentos>)

- Nome do ATO: é o nome do ATO da operação a ser utilizada;
- Nome da operação: é o nome da operação a ser utilizada;
- Seletor: é um objeto do tipo do ATO para qual a mensagem será enviada. Se na operação que está sendo chamada o seletor for um objeto, a ICS compara

o seletor com os argumentos correspondentes. Caso os objetos sejam iguais então, a operação é então executada. No caso do seletor na operação for uma variável, a operação receberá o objeto enviado, como também os argumentos e executar a operação;

- Argumentos: lista de argumentos necessários para utilizar a operação desejada, porém esta lista não precisa ser tratada com operações referentes ao tipo de dados "Lista".

Um exemplo da utilização de uma chamada ICS de um ATO para outro é apresentado na figura 5.3. Segundo Nunes (1994), a pesquisa da operação é feita de cima para baixo dentro do ATO. Se o nome do ATO presente na ICS coincidir com o ATO encontrado, e o *nome* da operação fizer referência a uma operação pertencente ao ATO, e o seletor for o mesmo, então os argumentos serão ligados aos parâmetros formais e ser executada a operação.

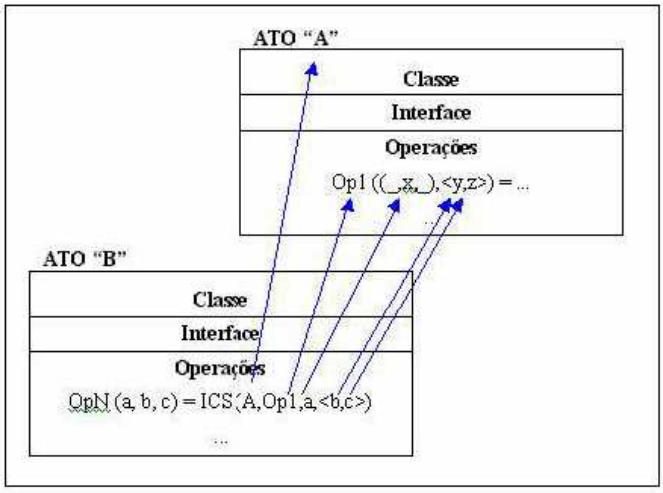


Figura 5.3: Exemplo de uma chamada ICS

No PROSOFT-Algébrico é usada uma linguagem gráfica para instanciação de tipos de dados derivada da notação de Jackson (JACKSON, 1993), conforme exemplificado nas figuras a seguir. Assim, a classe facilita o desenvolvimento das especificações algébricas, fornecendo uma notação gráfica que é uma instanciação do tipo definido. Uma classe representa a hierarquia de composição de tipos de dados, onde o nodo superior descreve o nome do tipo definido e os nodos-folha são associados a tipos primitivos ou definidos pelo usuário (REIS, 2002).

Os tipos de dados utilizados no PROSOFT são classificados da seguinte forma:

- Primitivos: são os tipos derivados da linguagem hospedeira, o Pascal. São eles: *integer, real, string, boolean, char, time, date*;
- Compostos: são tipos definidos no método algébrico: conjunto, lista, mapeamento, registro e união disjuntiva;
- Definidos pelo usuário: são tipos construídos a partir de outros tipos, e são representados pelos ATOs desenvolvidos no ambiente.

Como os tipos primitivos são triviais, eles não serão apresentados neste trabalho. A seguir serão apresentados os tipos compostos e os tipos desenvolvidos pelo usuário.

O tipo composto Conjunto, representa uma coleção de objetos onde a ordem dos mesmos não é importante. Por exemplo, a figura 5.4 ilustra uma Escola que é formada por um conjunto (representado por S) de Sala. O tipo Sala é uma referencia ao ATO Sala.

Figura 5.4: Classe Escola

O tipo Mapeamento, representa uma função, onde ocorre a relação de um Tipo-Domínio com um Tipo-Imagem. Por exemplo, como ilustra a figura 5.5, um Indivíduo pode ser representado pelo seu Nome e pelo numero da sua Carteira de Identidade. Sendo que Nome é do tipo *string* (tipo primitivo) e Carteira de Identidade do tipo *integer* (tipo primitivo).

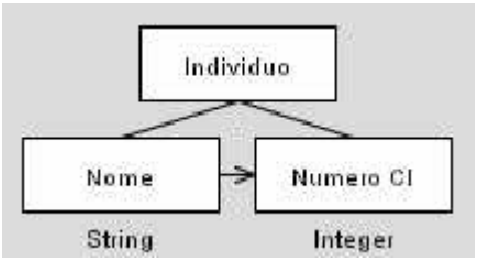


Figura 5.5: Classe indivíduo

O tipo Lista representa uma seqüência ordenada de zero ou mais componentes. Por exemplo, como ilustra a figura 5.6, uma Sala é formada por uma lista (representada pelo símbolo *) de Aluno. Sendo que Aluno é uma referência ao ATO Aluno.

Figura 5.6: Classe Sala

O tipo Registro define tuplas de tipos heterogêneos, expressando assim um produto cartesiano. Por exemplo, como ilustra a figura 5.7, um Livro possui Titulo, Autor, Editora, Ano e Número de Páginas. Sendo que Titulo, Autor e Editora são do tipo *string* e Ano e Número de Páginas do tipo *integer*.

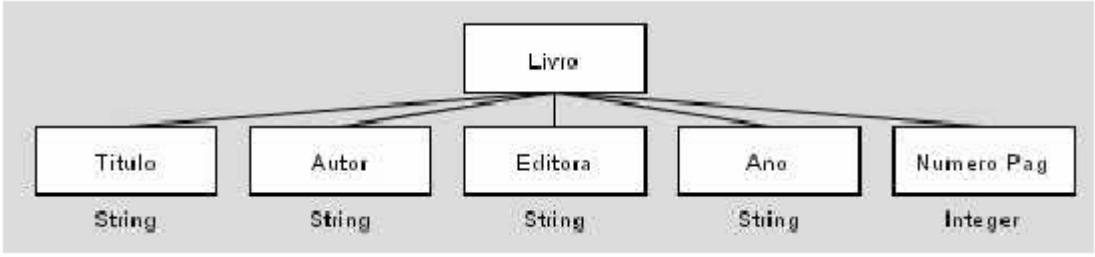


Figura 5.7: Classe Livro

O tipo União Disjuntiva define a união de tipos diferentes de elementos, sendo útil quando uma classe de dados necessita expressar valores alternativos. Por exemplo, como ilustra a figura 5.8, um Professor pode ser Titular, Horista ou Assistente. Cada tipo de Professor é do tipo *string*.

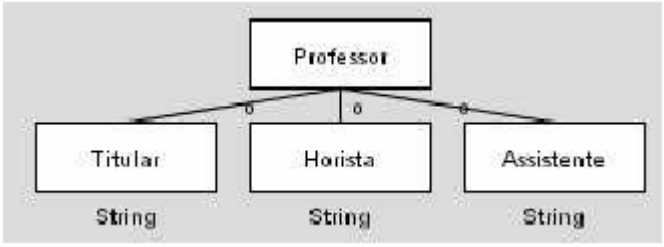


Figura 5.8: Classe Professor

Os tipos definidos pelo usuário são os ATOs existentes no PROSOFT. Sendo que estes ATOs já especificados podem ser usados na criação de novos ATOs.

Tendo sido apresentado esta breve visão geral sobre a notação da especificação algébrica do Prosoft, na próxima seção será apresentada a especificação do APSEE-*Metrics*. Maiores detalhes referentes ao formalismo do Prosoft podem ser encontrados em (NUNES, 1994) e (REIS, 2002). No anexo E deste trabalho encontra-se uma visão geral das operações do Prosoft Algébrico.

5.2 Especificação algébrica do APSEE-*Metrics*

A especificação algébrica do modelo APSEE-*Metrics* é apresentada nesta seção, onde são descritas as classes PROSOFT envolvidas na definição dos tipos abstratos de dados necessários.

A adoção do Prosoft-Algébrico para especificação da estrutura dos componentes do APSEE-*Metrics* proporcionou uma derivação parcial para o nível de implementação no ambiente Prosoft-Java[5], visto que há uma correspondência semântica entre os elementos usados na especificação e a implementação dos componentes de *software* descritos nesse paradigma. Além disso, o grupo de pesquisa no qual o trabalho foi desenvolvido possui uma vasta experiência no uso desse formalismo.

Utilizando o formalismo referido, serão descritas nas subseções que seguem cada um dos componentes do modelo.

---

[5] Prosoft-Java é a denominação do ambiente escolhido para prototipação do modelo proposto.

5.2.1 ATO APSEE-*Metrics*

O Ato APSEE-Metrics corresponde a classe responsável pelo gerenciamento das mensurações no ambiente APSEE, sendo esta incorporada ao ATO APSEE, originalmente proposto em (REIS, 2003). A figura 5.9 apresenta o ATO APSEE incluindo o APSEE-*Metrics*. A figura 5.10 apresenta o ATO APSEE-*Metrics*, sendo seus componentes apresentados à seguir.

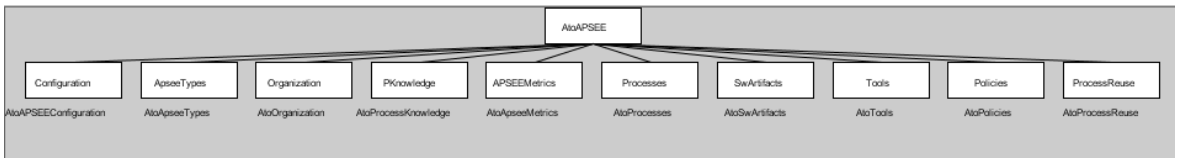


Figura 5.9: Representação gráfica do ATOAPSEE

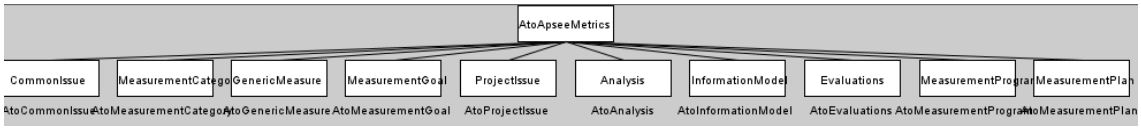


Figura 5.10: Representação gráfica do ATOApseeMetrics

O ato APSEE-*Metrics* é composto basicamente pelos elementos que compõe o modelo apresentado na seção 4.6, ou seja, os componentes do modelo ICM (Áreas, Categorias, Medidas Genéricas), os componentes do modelo de informação (objetivos de mensuração, informações de projeto, métricas, análises, avaliações), os componentes do programa de mensuração (modelo de informação, processo de mensuração) e por fim o plano de mensuração. Cada um destes elementos serão apresentados nas sub-seções que seguem.

5.2.2 ATOS do Modelo ICM

Para representar o modelo ICM foram construídos três ATOs, o ATO *CommonIssue*, ATO *MeasurementCategory* e o ATO *GenericMeasure*. Estes ATOs serão apresentados à seguir.

5.2.2.1. ATO CommonIssue

O ATO *CommonIssue* representa as áreas comuns do modelo ICM e permite ao especialista, incluir, excluir ou modificar as áreas comuns do modelo que servirão de guias para a seleção das métricas necessárias para o plano de mensuração.

A figura 5.11 apresenta a classe do respectivo ATO. Para construção do ATO *CommonIssue* foram utilizados os tipos mapeamento e registro. Assim, as características de uma área comum são definidas através de um mapeamento, onde o domínio da função é um identificador da área, e sua imagem como sendo um registro contendo dois atributos, nome e descrição.

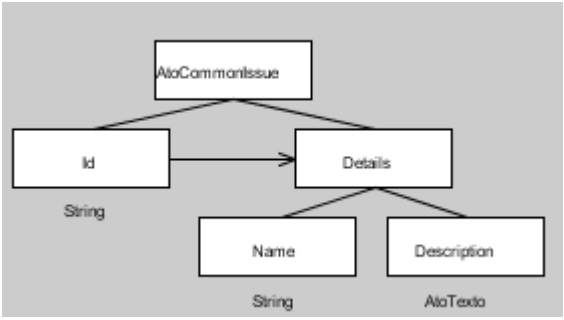


Figura 5.11: Representação gráfica do ATO *CommonIssue*

O Ato *CommonIssue* possui as seguintes operações:

- *newCommonIssue:* criação de um conjunto vazio de áreas comuns de medição;
- *addCommonIssue:* adição de uma nova área;
- *updCommonIssue:* alteração de uma área existente;
- *removeCommonIssue:* remoção de uma área existente;
- *existsCommonIssue:* retorna se uma área informada existe no conjunto de áreas;
- *getCommonIssues:* retorna uma dada área comum de um conjunto de áreas.

Na figura 5.12 é apresentada a especificação algébrica do ATO, através de sua interface, variáveis formais e também de suas operações.

INTERFACE

newCommonIssue		→ CommonIssue
addCommonIssue (_,_,_,_)	CommonIssue, String, String, Text	→ CommonIssue
updCommonIssue (_,_,_,_)	CommonIssue, String, String, Text	→ CommonIssue
removeCommonIssue (_,_)	CommonIssue, String	→ CommonIssue
existsCommonIssue (_,_)	CommonIssue, String	→ Boolean
getCommonIssue(_,_)	CommonIssue, String	→ CommonIssue

VARIÁVEIS FORMAIS

cIssue	COMMONISSUE
id, name	STRING
description	TEXTO
new_name	STRING
new_description	TEXTO
s_id	STRING

OPERAÇÕES

```
newCommonIssue = empty-mapping

addCommonIssue(cIssue, id, name, description)
 = modify (id, (Name name, Description description), cIssue)

updCommonIssue(modify(id, (Name name, Description description), cIssue), s_id, new_name, new_description)
=  if id = s_id
   then modify(id, (Name new_name, Description new_description), cIssue)
   else modify(id, (Name name, Description description), updCommonIssue(cIssue, s_id, new_name new_description))

updCommonIssue(empty-mapping,_,_,_) = empty-mapping

removeCommonIssue(cIssue, s_id) = restrict_with (cIssue, add(empty-set, s_id))

removeCommonIssue(empty-mapping, _) = empty-mapping

existsCommonIssue(modify(id, (Name name, Description description), cIssue), s_id)
=  if id = s_id
then true
```

```
else existsCommonIssue(cIssue, s_id)

existsCommonIssue (empty-mapping,_) = false

getCommonIssue(modify(id, (Name name, Description description), cIssue), s_id)
=  if id = s_id
then modify(id, (Name name, Description description), cIssue)
else getCommonIssue(cIssue), s_id)

getCommonIssue (empty-mapping,_) = empty-mapping
```

Figura 5.12: Especificação algébrica do ATO *CommonIssue*

5.2.2.2. ATO MeasurementCategory

O ATO *MeasurementCategory* representa as categorias que compõe as áreas do modelo ICM e permite ao gerente do processo de *software*, incluir, excluir ou modificar as categorias das áreas comuns do modelo que servirão de guias para a seleção das medidas necessárias para o plano de mensuração, agrupando as medidas genéricas por afinidades.

A figura 5.13 apresenta a classe do respectivo ATO. Para construção do ATO *MeasurementCategory* foram utilizados os tipos mapeamento e registro. Assim, as características de uma categoria são definidas através de um mapeamento, onde o domínio da função é um identificador da categoria, e sua imagem como sendo um registro contendo três atributos, nome, descrição e a área a qual ela pertence.

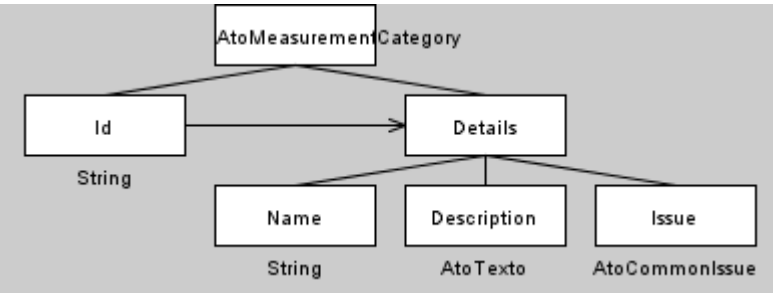


Figura 5.13: Representação gráfica do ATO *MeasurementCategory*

O Ato *MeasurementCategory* possui as seguintes operações:

- *newMeasurementCategory:* criação de um conjunto vazio de categorias de medição;
- *addMeasurementCategory:* adição de uma nova categoria;
- *updMeasurementCategory:* alteração de uma categoria existente;
- *removeMeasurementCategory:* remoção de uma categoria;
- *existsMeasurementCategory:* retorna se uma categoria informada existe no conjunto de categorias;
- *getMeasurementCategory:* retorna uma dada categoria de um conjunto de categorias.

No apêndice A é apresentada a especificação algébrica do ATO, através de sua interface, variáveis formais e também de suas operações.

5.2.2.3. ATO GenericMeasure

O ATO *GenericMeasure* representa as medições genéricas que compõe do modelo ICM e permite ao gerente do processo de *software*, incluir, excluir ou modificar as medidas genéricas de cada categoria do modelo. Estas medições servirão de apoio para seleção das métricas a serem utilizadas no plano de mensuração.

A figura 5.14 apresenta a classe do respectivo ATO. Para construção do ATO *GenericMeasure* foram utilizados os tipos mapeamento e registro. Assim, as características de uma medição genérica são definidas através de um mapeamento, onde o domínio da função é um identificador da categoria, e sua imagem como sendo um registro contendo três atributos, nome, descrição e a categoria a qual ela pertence.

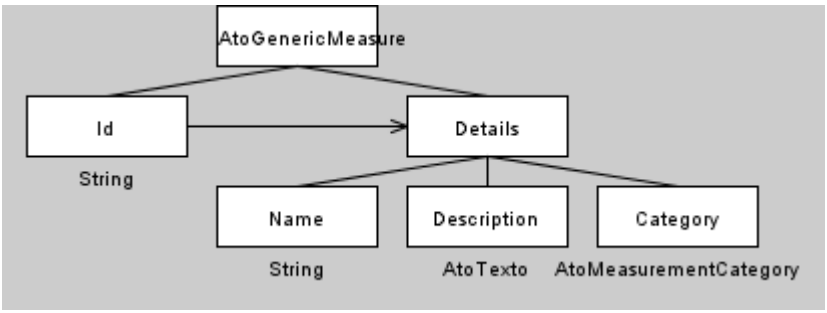


Figura 5.14: Representação gráfica do ATO *Generic Measure*

O Ato *GenericMeasure* possui as seguintes operações:

- *newGenericMeasure:* criação de um conjunto vazio de medidas genéricas;
- *addGenericMeasure:* adição de uma nova medida genérica;
- *updGenericMeasure:* alteração de uma medida existente;
- *removeGenericMeasure:* remoção de uma medida genérica;
- *existsGenericMeasure:* retorna se uma medida informada existe no conjunto de medidas genéricas;
- *getGenericMeasure:* retorna uma medida genérica de um conjunto.

No apêndice A é apresentada a especificação algébrica do ATO, através de sua interface, variáveis formais e também de suas operações.

5.2.3 ATOS do Modelo de Informação

Para representar o modelo de informação foram definidos os atos *InformationModel* (modelo de informação), *MeasurementGoal* (Objetivos da mensuração), *ProjectIssues* (Informações do projeto), *MetricDefinition* (Definição das Métricas), *MetricsCollected* (Métricas Coletadas), *Analyses* (Análises) *e Evaluations* (Avaliações). Estes ATOs serão apresentados à seguir.

5.2.3.1. ATO InformationModel

O ATO *InformationModel* representa o modelo de informação das mensurações e permite ao gerente do processo de *software*, incluir, excluir ou modificar o modelo de informação, conforme as necessidades.

A figura 5.15 apresenta a classe do respectivo ATO. Para construção do ATO *InformationModel* foram utilizados os tipos mapeamento, registro e conjunto. Assim, as características de um modelo de informação são definidas através de um mapeamento, onde o domínio da função é um identificador do modelo de informação, e sua imagem como sendo um registro contendo os seguintes atributos:

Name (nome): que representa o nome do modelo de informação;

Description (descrição): uma descrição resumida do modelo de informação, apresentando informações relevantes sobre os objetivos ao qual o modelo se propõe, cenários de aplicação, tipo de organização a ser aplicado, enfim, informações textuais para auxiliar na orientação de sua utilização futura pelos gerentes do processo de desenvolvimento.

Goals (objetivos de mensuração): conjunto de objetivos genéricos de mensuração que são associados ao modelo de informação. Estes alvos deverão ser previamente definidos e correspondem aos alvos genéricos do plano de mensuração definidos no modelo.

MetricDef (Definições de Métricas): conjunto de métricas que foram associadas ao modelo de informação, ou seja, são as especificações das métricas a serem coletadas durante a execução do processo de desenvolvimento. Estas métricas deverão ser previamente definidas e então associadas aos modelos de informação que irão utilizá-las.

MetricsCollected (Métricas coletadas): corresponde ao conjunto de métricas coletadas durante o processo de mensuração.

Analyses (Análises): conjunto de análises que foram associadas ao modelo de informação, ou seja, são as definições das análises a serem realizadas durante a execução das atividades de análise do processo de mensuração. Estas análises deverão ser previamente definidas na etapa de planejamento e então associadas ao modelo de informação que irá utilizá-la.

Evaluations (Avaliações): conjunto de avaliações que foram associadas ao modelo de informação, ou seja, são as definições das avaliações a serem realizadas sobre as métricas utilizadas, processo de mensuração ou do plano de mensuração como um todo. Estas avaliações deverão ser previamente definidas e então associadas aos modelos de informação que irão utilizá-las.

Os ATOs correspondentes aos conjuntos apresentados no ATO *InformationModel* serão apresentado em maior nível de detalhes nas sub-seções à seguir.

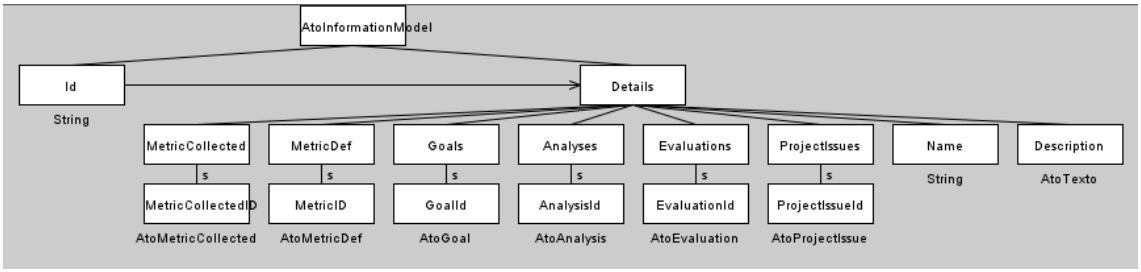


Figura 5.15: Representação gráfica do ATO *InformationModel*

O Ato *InformationModel* possui as seguintes operações:

- *newInformationModel:* criação de um conjunto vazio de modelos de informação;
- *addInformationModel:* adição de um novo modelo de informação ao conjunto de modelos;
- *updInformationModel:* alteração de um modelo de informação existente;
- *removeInformationModel:* remoção de um modelo;
- *existsInformationModel:* retorna se uma modelo informado existe no conjunto de modelos;
- *getInformationModel:* retorna um dado modelo de informação de um conjunto de modelos.
- *addGoal:* adiciona um objetivo de medição a um modelo de informação existente;
- *addAnalysis:* adiciona uma análise a um modelo de informação existente;
- *addEvaluation:* adiciona uma avaliação a um modelo de informação existente;
- *addProjectIssue:* adiciona uma informação específica do projeto a um modelo de informação existente;
- *addMetricsDef:* adiciona definições de métricas a um modelo de informação existente;
- *addMetricCollected* adiciona métricas coletadas a um modelo de informação existente;
- *removeGoal:* remove um objetivo de medição de um dado modelo de informação;
- *removeAnalysis:* remove uma análise de um dado modelo de informação;
- *removeEvaluation:* remove uma avaliação de um dado modelo de informação;
- *removeProjectIssue:* remove uma informação crítica de projeto de um dado modelo de informação;

- *removeMetricDef:* remove uma definição de métrica de um dado modelo de informação;
- *removeMetricCollected:* remove uma métrica coletada de um dado modelo de informação;

Na figura 5.16 é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

INTERFACE

newInformationModel		→ InformationModel
addInformationModel (_,_,_,_)	InformationModel, String, String, Text	→ InformationModel
updInformationModel (_,_,_,_)	InformationModel, String, String, Text	→ InformationModel
removeInformationModel (_,_)	InformationModel, String	→ InformationModel
existsInformationModel (_,_)	InformationModel, String	→ Boolean
getInformationModel (_,_)	InformationModel, String	→ InformationModel
addGoal (_,_,_)	InformationModel, String, MeasurementGoal	→ InformationModel
addAnalysis (_,_,_)	InformationModel, String, Analysis	→ InformationModel
addEvaluation (_,_,_)	InformationModel, String, Evaluation	→ InformationModel
addProjectIssue (_,_,_)	InformationModel, String, ProjectIssue	→ InformationModel
addMetricDef (_,_,_)	InformationModel, String, MetricDef	→ InformationModel
addMetricCollected (_,_,_)	InformationModel, String, MetricCollected	→ InformationModel
removeGoal (_,_,_)	InformationModel, String, MeasurementGoal	→ InformationModel
removeAnalysis (_,_,_)	InformationModel, String, Analysis	→ InformationModel
removeEvaluation (_,_,_)	InformationModel, String, Evaluation	→ InformationModel
removeProjectIssue (_,_,_)	InformationModel, String, ProjectIssue	→ InformationModel
removeMetricDef (_,_,_)	InformationModel, String, MetricDef	→ InformationModel
removeMetricCollected (_,_,_)	InformationModel, String, MetricCollected	→ InformationModel

VARIÁVEIS FORMAIS

model	INFORMATIONMODEL
id, name	STRING
description	TEXTO
goal	MEASUREMENTGOAL
analysis	ANALYSIS
evaluation	EVALUATION
projectissue	PROJECTISSUE
metricdef	METRICDEF
metriccollected	METRICCOLLECTED
goals	SETOFMEASUREMENTGOAL
analyses	SETOFANALYSIS
evaluations	SETOFEVALUATION
projectissues	SETOFPROJECTISSUE
metricsdef	SETOFMETRICDEF
metricscollected	SETOFMETRICCOLLECTED
s_id, new_name	STRING
new_description	TEXTO

OPERAÇÕES

newModel = empty-mapping

addModel (model, id, name, description)
= modify (id, (Name name, Description description, Goals add(empty-set), Analyses add(empty-set), Evaluations add(empty-set), ProjectIssues add(empty-set), Metricdef add(empty-set), MetricsCollected add(empty-set)), model)

updModel (modify(id, (Name name, Description description, _, _, _, _, _, _), model), s_id, new_name, new_description)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, _, _, _, _, _, _), model)
else modify(id, (Name name, Description description, _, _, _, _, _, _), updModel (model, s_id, new_name new_description))

updModel(empty-mapping,_,_,_) = empty-mapping

removeModel(model, s_id) = restrict_with (model, add(empty-set, s_id))

removeModel (empty-mapping, _) = empty-mapping

existsModel(modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model), s_id)
= **if** id = s_id
then true
else existsModel (model, s_id)

existsModel (empty-mapping,_) = false

```
getModel(modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues
projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model), s_id)
= if id = s_id
  then modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues
projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model)
  else getModel (model, s_id)


getModel (empty-mapping,_) = empty-mapping

addGoal (modify (id, (_,_, Goals goals, _, _, _, _, _), model), s_id, goal)
= if id = s_id
  then modify (id, (_,_,Goals add(goals, goal), _, _, _, _, _), model)
  else modify (id, (_,_, Goals goals, _, _, _, _, _), addGoals (model, s_id, goal))

addGoal (empty-mapping, _, _) = empty-mapping

removeGoal (modify(id, (_,_, Goals. goal, _, _, _, _, _), model), s_id, goal)
= if id = s_id
then modify(id, (_,_, Goals. delete(goals, goal), _,_,_,_,_), model)
else modify(id, (_,_, Goals. goals, _,_,_,_,_), removeGoal(model, s_id, goal))

removeGoal (empty-mapping,_,_) = empty-mapping

addAnalysis (modify (id, (_,_,_, Analyses analysis, _, _, _, _), model), s_id, analysis)
= if id = s_id
  then modify (id, (_,_,_, Analyses add(analyses, analysis), _, _, _, _), model)
  else modify (id, (_,_,_, Analyses analyses, _, _, _, _), addAnalysis (model, s_id, analysis))

addAnalysis (empty-mapping, _, _) = empty-mapping

removeAnalysis (modify(id, (_,_,_, Analyses. analysis, _, _, _, _), model), s_id, analysis)
= if id = s_id
then modify(id, (_,_,_, Analyses. delete(analyses, analysis), _,_,_,_), model)
else modify(id, (_,_,_, Analyses. analyses, _,_,_,_), removeAnalysis (model, s_id, analysis))

removeAnalysis (empty-mapping,_,_) = empty-mapping

addEvaluation (modify (id, (_,_,_,_, Evaluations evaluation, _, _, _), model), s_id, evaluation)
= if id = s_id
  then modify (id, (_,_,_,_, Evaluations add(evaluations, evaluation), _, _, _), model)
  else modify (id, (_,_,_,_, Evaluations evaluations, _, _, _), addEvaluation (model, s_id, evaluation))

addEvaluation (empty-mapping, _, _) = empty-mapping

removeEvaluation (modify(id, (_,_,_,_, Evaluations. evaluations, _, _, _), model), s_id, evaluation)
= if id = s_id
then modify(id, (_,_,_,_, Evaluations. delete(evaluations, evaluation), _,_,_), model)
else modify(id, (_,_,_,_, Evaluations. evaluations, _,_,_), removeEvaluation (model, s_id, evaluation))

removeEvaluation (empty-mapping,_,_) = empty-mapping

addProjectIssue (modify (id, (_,_,_,_,_,_, Issues issues, _, _), model), s_id, issue)
= if id = s_id
  then modify (id, (_,_,_,_,_,_, Issues add(issues, issue), _, _), model)
  else modify (id, (_,_,_,_,_,_, Issues issues, _, _), addProjectIssue (model, s_id, issue))

addProjectIssue (empty-mapping, _, _) = empty-mapping

removeProjectIssue (modify(id, (_,_,_,_,_,_,Issues. issues, _, _), model), s_id, issue)
= if id = s_id
then modify(id, (_,_,_,_,_,_, Issues. delete(issues, issue), _,_), model)
else modify(id, (_,_,_,_,_,_, Issues. issues, _,_), removeProjectIssue (model, s_id, issue))

removeProjectIssue (empty-mapping,_,_) = empty-mapping

addMetricDef (modify (id, (_,_,_,_,_,_,_,MetricsDef metricsdef, _), model), s_id, metricdef)
= if id = s_id
  then modify (id, (_,_,_,_,_,_,_,MetricsDef add(metricsdef, metricdef), _), model)
  else modify (id, (_,_,_,_,_,_,_,MetricsDef metricsdef,_), addMetricDef (model, s_id, metricdef))

addMetricDef (empty-mapping, _, _) = empty-mapping

removeMetricDef (modify(id, (_,_,_,_,_,_,MetricsDef. metricsdef, _), model), s_id, metricdef)
= if id = s_id
then modify(id, (_,_,_,_,_,_,_,MetricsDef. delete(metricsdef, metricdef), _), model)
else modify(id, (_,_,_,_,_,_,_,MetricsDef. metricsdef, _), removeMetricDef (model, s_id, metricdef))

removeMetricDef (empty-mapping,_,_) = empty-mapping

addMetricCollected (modify (id, (_,_,_,_,_,_,_,MetricsCollected metricscollected, _), model), s_id, metriccollected)
= if id = s_id
  then modify (id, (_,_,_,_,_,_,_,MetricsCollected add(metricscollected, metriccollected), _), model)
  else modify (id, (_,_,_,_,_,_,_,MetricsCollected metricscollected,_), addMetricCollected (model, s_id, metriccollected))

addMetricCollected (empty-mapping, _, _) = empty-mapping

removeMetricCollected (modify(id, (_,_,_,_,_,_,MetricsCollected. metricscollected, _), model), s_id, metriccollected)
= if id = s_id
```

```
then modify(id, (_,_,_,_,_,_,MetricsCollected. delete(metricscollected, metriccollected), _), model)
else modify(id, (_,_,_,_,_,_,MetricsCollected. metricscollected, _), removeMetricCollected (model, s_id, metriccollected))

removeMetricCollected (empty-mapping,_,_) = empty-mapping
```

Figura 5.16: Especificação algébrica do ATO *InformationModel*

5.2.3.2. ATO MeasurementGoal

O ATO *MeasurementGoal* representa os objetivos de mensuração que poderão ser utilizados para cada modelo de informação.

A figura 5.17 apresenta a classe do respectivo ATO. Para construção do ATO *MeasurementGoal* foram utilizados os tipos mapeamento e registro. Assim, as características de um objetivo de mensuração são definidas através de um mapeamento, onde o domínio da função é um identificador do objetivo e sua imagem como sendo um registro contendo três atributos, nome, descrição, a prioridade em relação aos demais objetivos de mensuração.

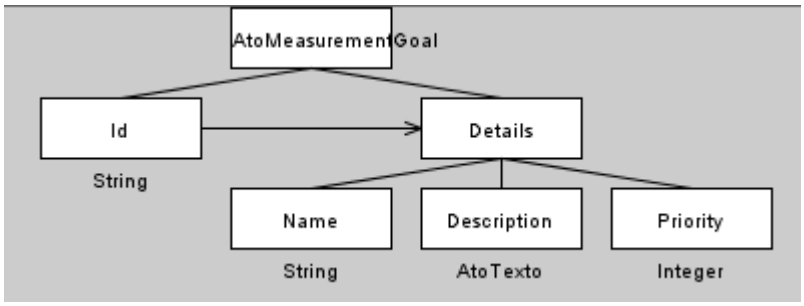


Figura 5.17: Representação gráfica do ATO *MeasurementGoal*

O Ato *MeasurementGoal* possui as seguintes operações:

- *newMeasurementGoal:* criação de um conjunto vazio de objetivos de medição;
- *addMeasurementGoal:* adição de um novo objetivo de medição;
- *updMeasurementGoal:* alteração de um objetivo de medição;
- *removeMeasurementGoal:* remoção de um objetivo;
- *existsMeasurementGoal:* retorna se um dado objetivo informado existe no conjunto de objetivos;
- *getMeasurementGoal:* retorna um dado objetivo de medição de um conjunto de objetivos.

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.3.3. ATO ProjectIssue

O ATO *ProjectIssue* representa as informações críticas do projeto definidas para o modelo de informação. A cada uma destas informações é permitido ao gerente definir uma ou mais áreas genérica associada a ele.

A figura 5.18 apresenta a classe do respectivo ATO. Para construção do ATO *MeasurementGoal* foram utilizados os tipos mapeamento, registro e conjunto. Assim, suas características são definidas através de um mapeamento, onde o domínio da função é um identificador e sua imagem como sendo um registro contendo os atributos, nome, descrição, a probabilidade, o impacto, bem como um conjunto de áreas comuns de mensuração do modelo ICM ao qual cada informação está associada.

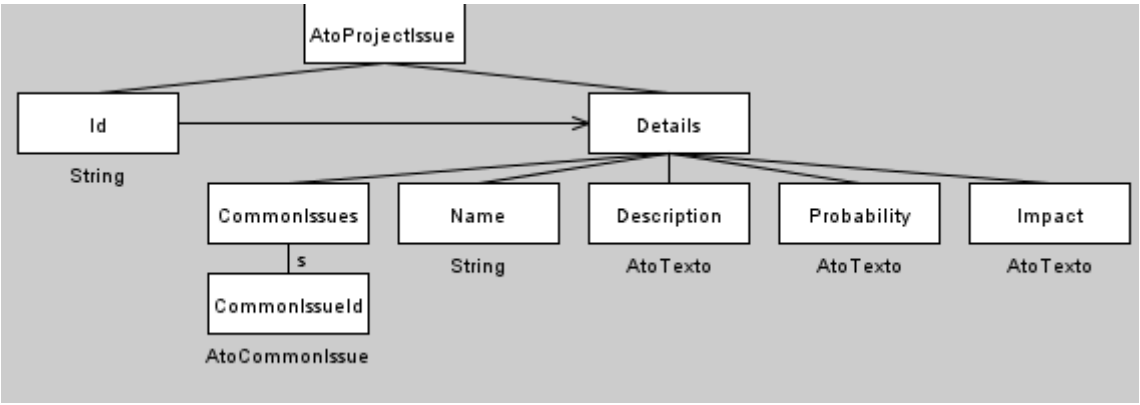


Figura 5.18: Representação gráfica do ATO *ProjectIssue*

O Ato *ProjectIssue* possui as seguintes operações:

- *newProjectIssue:* criação de um conjunto vazio de informações críticas de projeto;
- *addProjectIssue:* adição de uma nova informação crítica de projeto;
- *updProjectIssue:* alteração de uma informação crítica de projeto;
- *removeProjectIssue:* remoção de uma informação crítica de projeto;
- *existsProjectIssue:* retorna se uma dada informação crítica de projeto informada existe em um dado conjunto de informações críticas;
- *getProjectIssue:* retorna uma dada informação crítica de projeto de um dado conjunto.

Abaixo é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.3.4. ATO MetricsDef

O ATO *MetricDef* representa a definição de cada métrica a ser utilizada, conforme proposto em (REIS, 2003). A figura 5.19 apresenta a classe do respectivo ATO. Para construção do ATO *MetricsDef* foram utilizados os tipos mapeamento, registro, conjunto e união disjuntiva. Assim, as características de uma definição de métrica são definidas através de um mapeamento, onde o domínio da função é um identificador da métrica e sua imagem como sendo um registro contendo os atributos nome (*name*), descrição (*description*) e tipo (*type_id*). Além destes, uma união disjuntiva representa o tipo da métrica sendo definida (*kind*), que pode ser de processo, produto ou individual. O registro

também compreende o conjunto de unidades de medida da métrica (*unit*), o intervalo de validade da métrica (*range*), bem como a descrição de como a métrica poderá ser obtida (*how*).

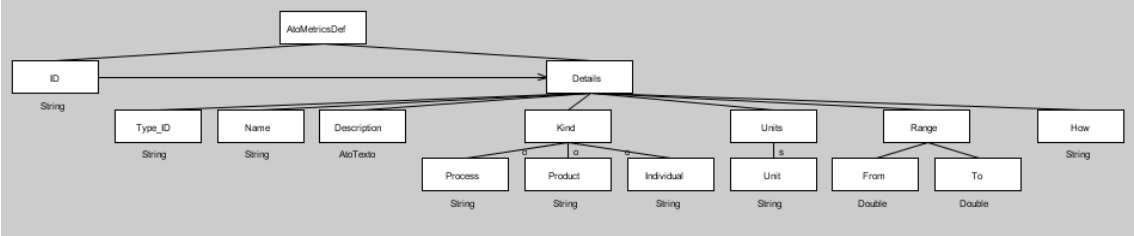


Figura 5.19: Representação gráfica do ATO *MetricsDef*

O Ato *MetricsDef* possui as seguintes operações:

- *newMetricsDef:* criação de um conjunto vazio de definições de métricas;
- *addMetricsDef:* adição de uma nova definição de métrica;
- *updMetricsDef:* alteração de uma definição de métrica;
- *removeMetricsDef:* remoção de uma definição de métrica;
- *existsMetricsDef:* retorna se uma dada definição de métrica informada existe em um dado conjunto;
- *getMetricsDef:* retorna dada definição de métrica existentes de um conjunto;
- *addUnit:* adição de unidades de mediçãoa uma definição de métrica existente;
- *removeUnit:* remoção de unidades de mediçãoa uma definição de métrica existente.

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.3.5. ATO MetricsCollected

O ATO *MetricsCollected* representa as métricas coletadas, conforme proposto em (REIS, 2003). A figura 5.20 apresenta a classe do respectivo ATO.

O ATO representa um conjunto de métricas coletadas, representadas através de um conjunto (*Metrics).* As características das métrica coletadas são definidas através de um registro contendo um identificador (*MetricID*), o valor coletado (*Value*), a unidade de medida (*Unit*), o elemento medido (*ElementID*), uma união disjuntiva podendo ser um artefato, processo, atividade, recurso, agente, grupo ou a organização em si, além do período da medição realizada (*Period*).

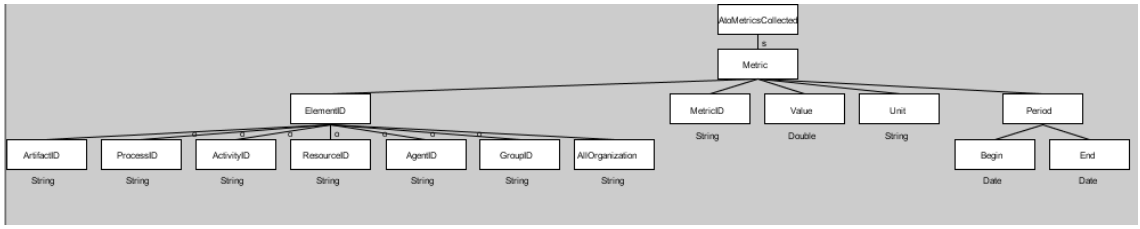


Figura 5.20: Representação gráfica do ATO *MetricsCollected*

O Ato *MetricsCollected* possui as seguintes operações:

- *newMetricsCollected:* criação de um conjunto vazio de métricas coletadas;
- *newMetric:* adição de uma métrica coletada ao conjunto de métricas;
- *removeMetric:* remoção de uma métrica coletada do conjunto;
- *existsMetric:* retorna se uma dada métrica coletad existe em um dado conjunto;
- *getMetric:* retorna uma métrica coletada de um dado conjunto;

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.3.6. ATO Analysis

O ATO *Analysis* representa a definição das análises a serem realizadas durante o processo de mensuração. A figura 5.21 apresenta a classe do respectivo ATO, onde foram utilizados os tipos mapeamento, registro e conjunto.

Assim, as características de uma análise são definidas através de um mapeamento, onde o domínio da função é um identificador da análise e sua imagem como sendo um registro, contendo como atributos um conjunto de objetivos do modelo de informação que serão associados a cada análise (visando orientar esta análise), bem como os atributos, nome, descrição e os resultados obtidos com as análises realizadas, sendo este último necessário para provimento de informações históricas e formando base de conhecimento acerca das mensurações realizadas.

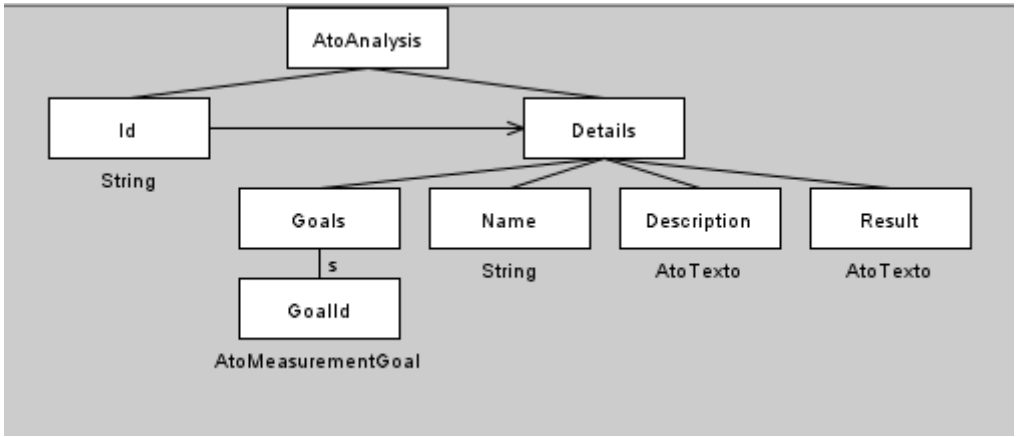


Figura 5.21: Representação gráfica do ATO *Analysis*

O Ato *Analysis* possui as seguintes operações:

- *newAnalysis:* criação de um conjunto vazio de análises;
- *addAnalysis:* adição de uma nova análise;
- *updAnalysis:* alteração de uma análise existente;
- *removeAnalysis:* remoção de uma análise existente;
- *existsAnalysis:* retorna se uma dada análise informada existe em um dado conjunto de análises;
- *getAnalysis:* retorna uma análise de um dado conjunto;
- *addMeasurementGoal:* adição de um novo objetivo de medição a uma análise específica;
- *removeMeasurementGoal:* remoção de um objetivo de medição de uma análise específica.

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.3.7. ATO Evaluations

O ATO *Evaluations* representa a definição das avaliações a serem realizadas após as mensurações. A figura 5.22 apresenta a classe do respectivo ATO, onde foram utilizados os tipos mapeamento, registro e união disjuntiva.

Assim, as características de uma avaliação são definidas através de um mapeamento, onde o domínio da função é um identificador da avaliação e sua imagem como sendo um registro, contendo os atributos, nome, descrição e os resultados obtidos com as avaliações realizadas, sendo este último necessário para provimento de informações históricas e formando base de conhecimento acerca das mensurações realizadas. Além destes, o atributo type_id representa o tipo da avaliação sendo definida, que pode ser das métricas, das atividades, do processo, ou do plano de mensuração, sendo este atributo pertencente a um dos tipos definidos na hierarquia de tipos do APSEE.

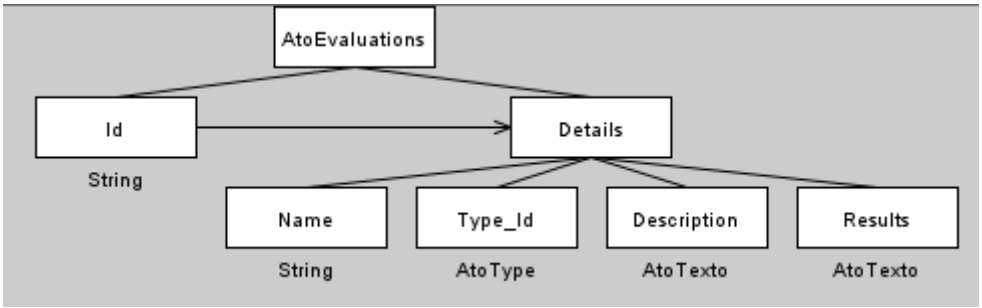


Figura 5.22: Representação gráfica do ATO *Evaluations*

O Ato *Evaluation* possui as seguintes operações:

- *newEvaluation:* criação de um conjunto vazio de avaliações;
- *addEvaluation:* adição de uma nova avaliação;

- *updEvaluation:* alteração de uma avaliação existente;
- *removeEvaluation:* remoção de uma avaliação existente;
- *existsEvaluation:* retorna se uma dada avaliação informada existe em um dado conjunto de avaliações;
- *getEvaluation:* retorna uma avaliação de um dado conjunto;

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.4 ATOS do Processo de Mensuração

Para representar o processo de mensuração foram definidos os atos *MeasurementProcess* (Processo de Mensuração) e utilizados alguns atos já existentes no ambiente APSEE, como por exemplo o ATO *Processes*. Estes ATOs serão apresentados à seguir.

5.2.4.1. ATO MeasurementProcess

O ATO *MeasurementProcess* representa o processo de mensuração a ser utilizado, na qual são definidas cada uma das atividades necessárias para realização de cada etapa de um programa de mensuração.

A figura 5.23 apresenta a classe do respectivo ATO. Para construção do ATO *MeasurementProcess* foram utilizados os tipos mapeamento e registro. Assim, as características de um processo de mensuração são definidas através de um mapeamento, onde o domínio da função é um identificador do processo, e sua imagem como sendo um registro contendo dois atributos, o nome do processo e uma instância de um processo de *software* definido pelo ambiente APSEE, representado pelo ATO *Processes*.

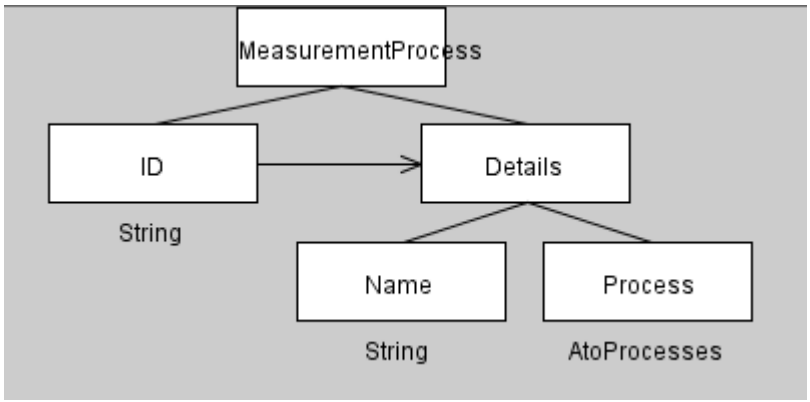


Figura 5.23: Representação gráfica do ato *Measurement Process*

O Ato *MeasurementProcess* possui as seguintes operações:

- *newMeasurementProcess:* criação de um conjunto vazio de processos de mensuração;
- *addMeasurementProcess:* adição de um novo processo de mensuração;

- *updMeasurementProcess:* alteração de um processo de mensuração existente;
- *removeMeasurementProcess:* remoção de um processo de mensuração existente;
- *existsMeasurementProcess:* retorna se um dad processo de mensuração informado existe em um dado conjunto de processos;
- *getMeasurementProcess:* retorna um processo de mensuração de um dado conjunto;

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.4.2. ATO Processes

O ATO *Processes* representa o processo em si, conforme sua definição original proposta em (REIS, 2003).

A figura 5.24 apresenta a classe do respectivo ATO. Para construção do ATO *Processes* foram utilizados os tipos mapeamento, registro e união disjuntiva. Assim, as características de um processo são definidas através de um mapeamento, onde o domínio da função é um identificador do processo, e sua imagem como sendo um registro contendo os atributos:

Tipo (*type_id)*: que corresponde a um tipo na hierarquia de atividades;

Estado (*Gen_state*): que corresponde ao estado do processo, podendo ser *Not_started* (não iniciado), *Enacting* (o processo está em execução) ou *finished* (processo está concluído);

Modelo de processo (*ProcessModel*): que corresponde ao modelo do processo. A distinção entre processo e modelo de processo foi proposta para determinar a situação geral do nodo raiz da rede de atividades, pois modelos de processo são definidos de forma recursiva (são compostos de atividades que podem ser decompostas em novos modelos de processo).

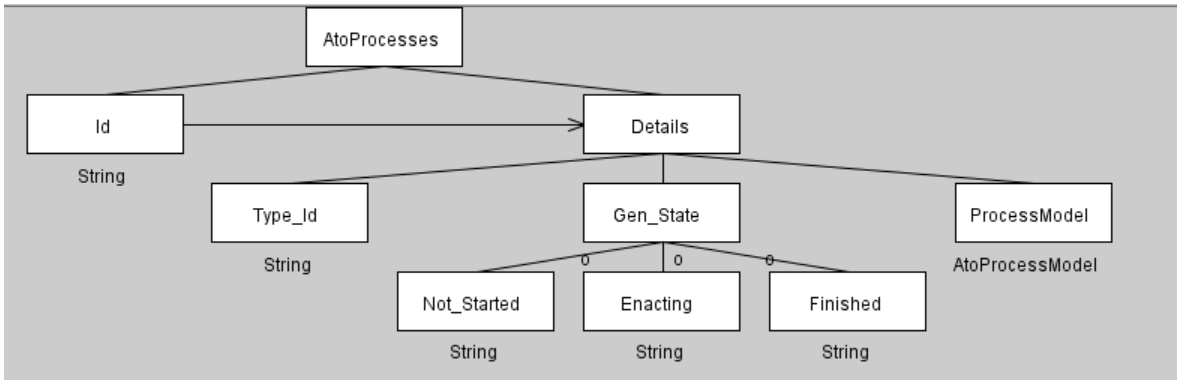


Figura 5.24: Representação gráfica do ATO *Processes*

A especificação das operações foi da classe Process foi preservada e pode ser encontrada em (REIS, 2003).

5.2.5 ATOS do Programa de Mensuração

Para representar um programa de mensuração foram definidos os atos *MeasurementProgram* (Programa de Mensuração) e *MeasurementPlan* (Plano de Mensuração). Estes ATOs serão apresentados à seguir.

5.2.5.1. ATO MeasurementProgram

O ATO *MeasurementProgram* representa o programa de mensuração definido. A figura 5.25 apresenta a classe do respectivo ATO. Para construção do ATO *MeasurementProgram* foram utilizados os tipos mapeamento e registro. Assim, as características de um programa de mensuração são definidas através de um mapeamento, onde o domínio da função é um identificador do programa de mensuração, e sua imagem como sendo um registro contendo os atributos, *name* (nome), *description* (descrição), o modelo de informação (*InformationModel*) definido para o programa em questão, bem como seu processo de mensuração (*MeasurementProcess*).

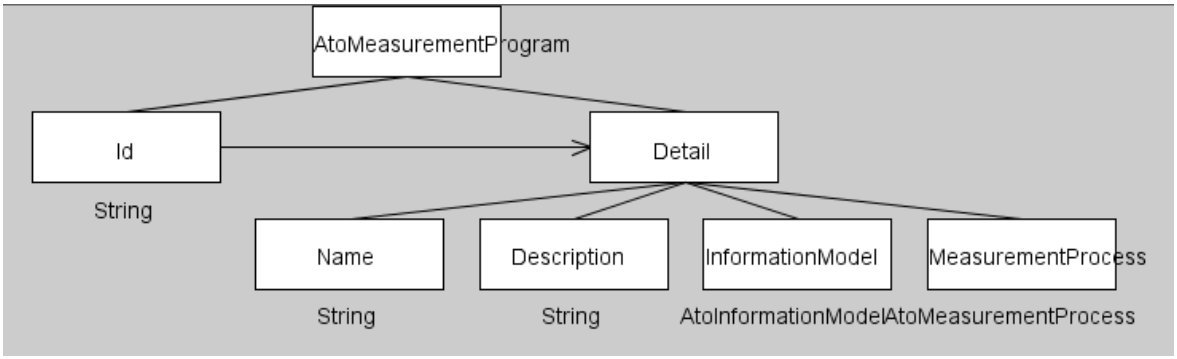


Figura 5.25: Representação gráfica do ATO *Measurement Program*

O Ato *MeasurementProgram* possui as seguintes operações:

- *newMeasProgram:* criação de um conjunto vazio de programas de mensuração;
- *addMeasProgram:* adição de um novo programa de mensuração;
- *updMeasProgram:* alteração de um programa de mensuração existente;
- *removeMeasProgram:* remoção de um programa de mensuração existente;
- *existsMeasProgram:* retorna se um dad programa de mensuração informado existe em um dado conjunto de processos;
- *getMeasProgram:* retorna um programa de mensuração de um dado conjunto;

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.2.5.2. ATO MeasurementPlan

O ATO *MeasurementPlan* representa o plano de mensuração utilizado em um projeto. A figura 5.26 apresenta a classe do respectivo ATO. Para construção do

ATO *MeasurementPlan* foram utilizados os tipos mapeamento e registro. Assim, as características de uma plano de mensuração são definidas através de um mapeamento, onde o domínio da função é um identificador do plano de mensuração, e sua imagem como sendo um registro contendo os atributos, *name* (nome), *description* (descrição), o programa de mensuração utilizado (*Program_id*), o contexto (*Context*) ao qual o plano de mensuração está sendo aplicado (organização, tipo de processo de desenvolvimento, tipo de *software*, etc.), além de informações adicionais (*Adic_informations*) relevantes para a caracterização do plano de mensuração (como por exemplo, objetivos das mensurações, informações sobre os resultados obtidos, ou demais informações históricas relevantes para os gerentes e desenvolvedores).

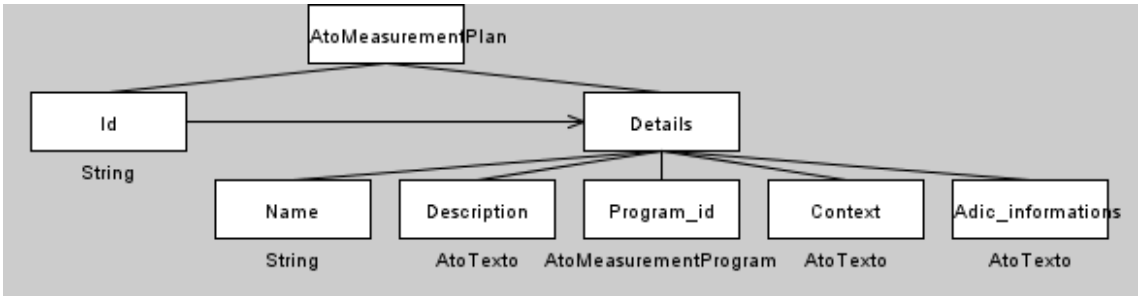


Figura 5.26: Representação gráfica do ATO *MeasurementPlan*

O Ato *MeasurementPlan* possui as seguintes operações:

- *newMeasPlan:* criação de um conjunto vazio de planos de mensuração;
- *addMeasPlan:* adição de um novo plano de mensuração;
- *updMeasPlan:* alteração de um plano de mensuração existente;
- *removeMeasPlan:* remoção de um plano de mensuração existente;
- *existsMeasPlan:* retorna se um dado plano de mensuração informado existe em um dado conjunto de planos de mensuração;
- *getMeasPlan:* retorna um plano de mensuração de um dado conjunto;

No apêndice A é apresentada a especificação algébrica do ATO, através de suas interfaces, variáveis formais e também de suas operações.

5.3 Considerações sobre o capítulo

Neste capítulo foi apresentada a especificação algébrica do modelo APSEE-*Metrics*, onde foi utilizado um formalismo chamado Prosoft-Algébrico, desenvolvido pelo grupo Prosoft, sendo amplamente utilizado na grande maioria dos trabalhos do grupo.

A especificação formal do modelo permitiu não somente a definição precisa dos elementos que compõe o modelo, mas também a derivação parcial desta especificação ao nível de implementação, através do ambiente Prosoft-Java, visto que há uma correspondência semântica entre os elementos usados na especificação e a implementação dos componentes de *software* descritos nesse paradigma. Assim, na seção seguinte será apresentada uma visão geral de um protótipo de uma

ferramenta desenvolvida com o objetivo de verificar a viabilidade do modelo, como veremos à seguir.

6 PROTÓTIPO DA FERRAMENTA

O presente capítulo apresenta o protótipo de uma ferramenta desenvolvida com base no modelo proposto neste trabalho, que consiste em uma ferramenta de apoio à abordagem proposta, provendo apoio computacional a mesma, além de servir como uma maneira de verificar a viabilidade do modelo.

A implementação do protótipo é restrita aos componentes principais do modelo, se limitando a algumas funcionalidades básicas que envolvem principalmente a definição do plano de mensuração, definição de métricas, formulários para coleta de dados das medições, além da visualização destas informações pelos envolvidos no processo de desenvolvimento.

Este protótipo visa guiar a execução das atividades previstas na abordagem proposta, possibilitando o registro das informações em cada fase do processo de mensuração. Um aspecto importante é que a aplicação do modelo foi realizada através de um ambiente de execução de processos de *software*, um PSEE, no caso o APSEE-Prosoft, uma vez que a ferramenta foi integrada ao ambiente de modelagem e execução de processos do APSEE.

Na seção seguinte veremos uma breve introdução ao ambiente utilizado como base para a implementação do protótipo.

6.1 Prosoft Java

De forma similar aos demais trabalhos do grupo de pesquisa ao qual este trabalho foi desenvolvido, apresentamos uma breve descrição do ambiente Prosoft-Java, com base na descrição apresentada em (REIS, 2003) e (SOUZA, 2004).

PROSOFT-Java é a denominação da mais recente da implementação do paradigma PROSOFT, na forma de um ambiente homogêneo e integrado de desenvolvimento de *software* escrito na linguagem Java (SCHLEBBE, 1997). O desenvolvimento atual do PROSOFT-Java é resultado do esforço cooperativo de estudantes e pesquisadores do PPGC-UFRGS e da *Fakultät Informatik da Universität Stuttgart* (Alemanha), sob orientação do Prof. Dr. Daltro José Nunes.

O principal objetivo do ambiente é estabelecer uma infra-estrutura que apoie o desenvolvimento de *software* de alta complexidade através da integração de ferramentas escritas em um paradigma próprio. A seguir são apresentadas algumas de suas principais características (REIS, 2003):

Portabilidade: o PROSOFT-Java portável no sentido em que o sistema pode ser executado nas plataformas para as quais estão disponíveis o Java *Runtime Enviromment* (SUN, 2005);

Distribuição: a distribuição entre ATOs fornecida atualmente através do mecanismo *ad-hoc* de comunicação denominado ICS-Distribuído que está implementado, atualmente, sobre o protocolo Java-RMI (SUN, 2005);

Cooperação: a funcionalidade para o trabalho cooperativo está disponível a partir da extensão denominada PROSOFT-Cooperativo, descrita em (REIS, 1998);

Geração de código: através do PROSOFT-Java uma classe do PROSOFT-Algébrico é diretamente mapeada para implementação em PROSOFT-Java, a qual inclui funções para criação e remoção de objetos e para obtenção e alteração dos valores armazenados nos nodos-folhas do tipo defnido. Na implementação atual, as funções adicionais especificadas algebricamente devem ser traduzidas manualmente para métodos escritos de acordo com a sintaxe da linguagem Java.

6.2 O ambiente APSEE

Nesta seção é apresentado o ambiente APSEE, bem como algumas das características do ambiente relacionadas com o protótipo do modelo desenvolvido. O APSEE foi desenvolvido através do paradigma PROSOFT. Portanto, utiliza os serviços de apoio ao trabalho distribuído e cooperativo fornecidos pelo PROSOFT-Java (REIS, 2003).

6.2.1 Linguagem de Modelagem – APSEE PML

Como citado no capítulo 4, o APSEE possui uma linguagem própria para modelagem de processos baseada em redes de atividades. Os modelos de processos construídos no APSEE são compostos de quatro partes principais: os requisitos textuais; a parte abstrata (com tipos), a parte instanciada (com instâncias de elementos do processo) e as informações sobre a execução (LIMA et al., 2002).

Basicamente, a PML oferece a representação gráfica para atividades, conexões de artefato, conexões simples e múltiplas. Os elementos sintáticos da linguagem APSEE-PML e sua representação gráfica utilizada no ambiente APSEE são apresentados resumidamente na figura 6.1.

O acesso as informações sobre os demais elementos do modelo (como agentes, seus papéis, data de início e término da execução, etc.) são dispostos através de formulários ou descrições textuais.

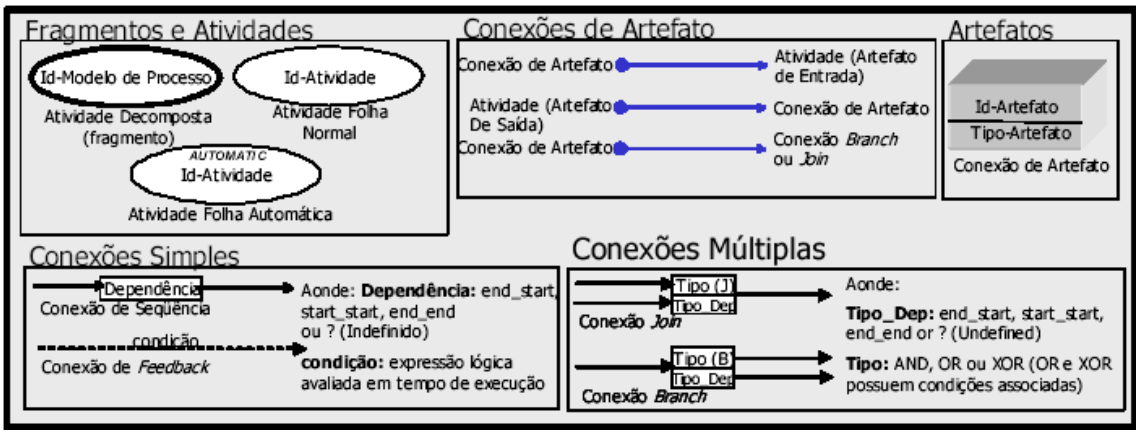


Figura 6.1: Representação gráfica do alfabeto da APSEE-PML

(REIS, 2003)

6.3 O protótipo APSEE-*Metrics*

Nesta seção é apresentado o protótipo da ferramenta APSEE-*Metrics,* integrada ao ambiente APSEE. A ferramenta desenvolvida constitui-se de uma implementação limitada de algumas das funcionalidades propostas no modelo proposto, e concentra-se principalmente nas funções de planejamento do modelo. Assim, nesta seção serão apresentadas de forma sucinta estas funcionalidades.

A figura 6.2 apresenta a interface inicial do APSEE. Na parte inferior desta tela encontram-se os principais módulos do APSEE, entre os quais, o APSEE-*Metrics.*

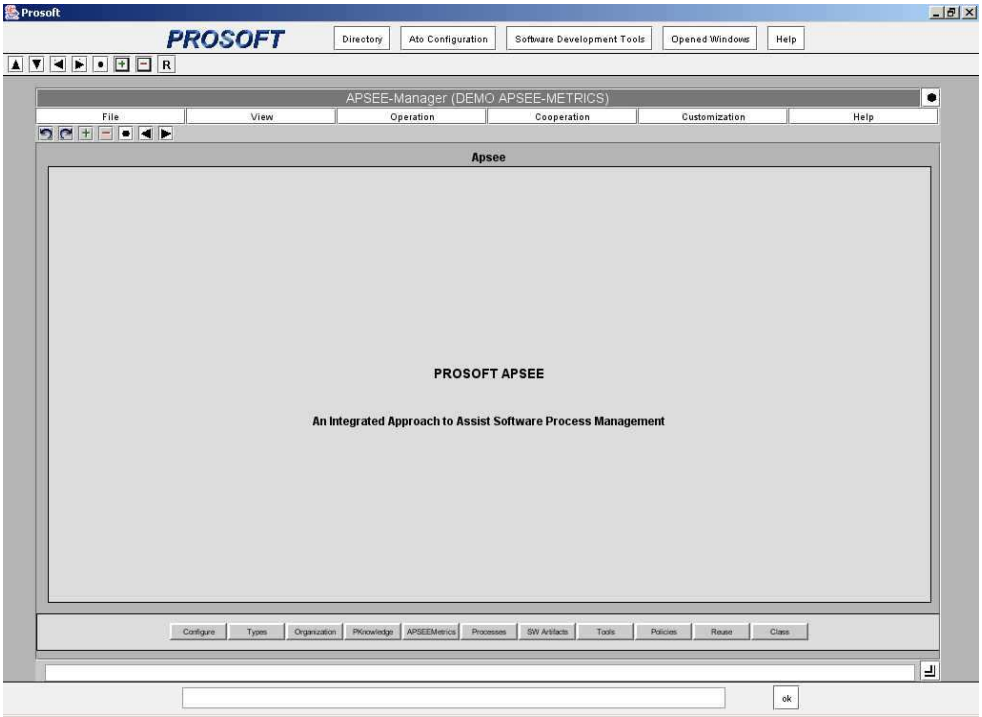
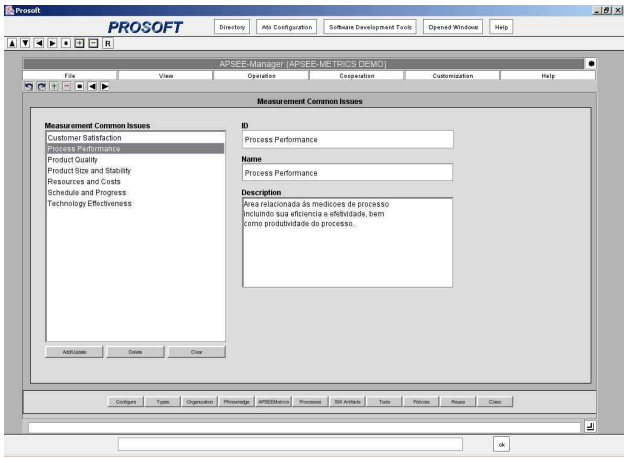


Figura 6.2: Tela inicial do APSEE

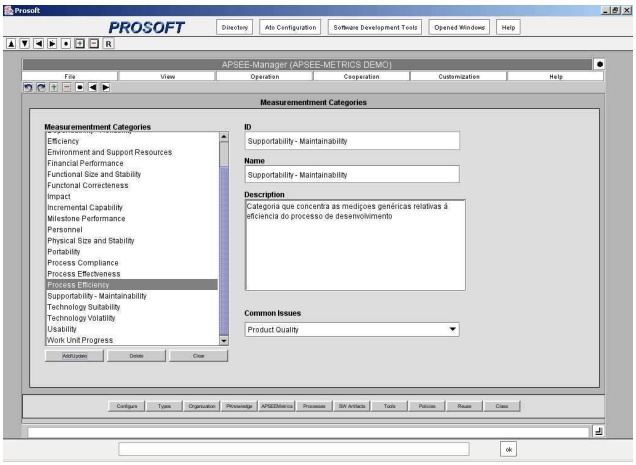
Nas sub-seções à seguir serão apresentadas as funcionalidades relacionadas ao APSEE-*Metrics.*

6.3.1 Funcionalidades do Modelo ICM

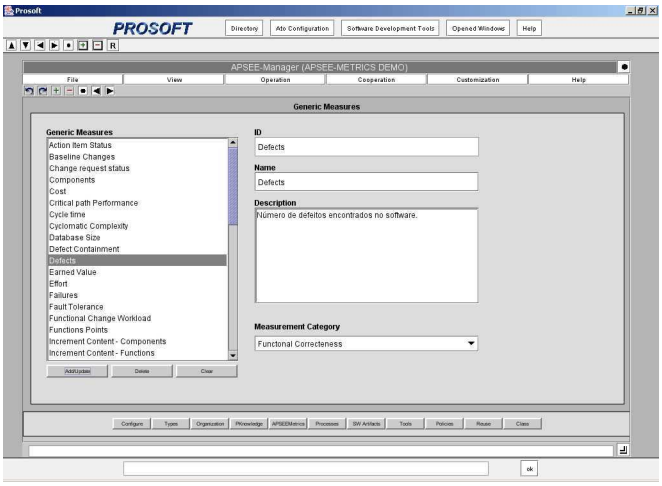
As funcionalidades ligadas aos registros das informações referentes ao modelo ICM foram implementadas, sendo apresentadas na figura 6.3.



Common Issues



Measurement Categories



Generic Measures

Figura 6.3: Telas do modelo ICM

A imagem da esquerda da figura 6.3 apresenta a interface de registro das áreas de medições do modelo ICM (*common issues*), permitindo o registro de cada área e das suas respectivas informações de contexto, orientando o usuário sobre o significado de cada uma das áreas. A interface permite a inclusão de novas áreas, bem como a alteração e exclusão das mesmas.

Na imagem do centro temos a interface de registro das categorias de medição do modelo ICM (*measurement categories*), permitindo o registro de cada categoria de medição, suas respectivas informações de contexto, bem como a área na qual está associada. A interface permite a inclusão de novas categorias, a associação das mesmas às áreas do modelo ICM, bem como a alteração e exclusão das mesmas.

Já na imagem da direita da figura, temos a interface de registro das medições genéricas do modelo ICM (*generic measures*), que permite o registro de cada medida genérica, de suas respectivas informações de contexto, bem como a categoria na qual está associada. A interface permite a inclusão de novas medidas genéricas, a associação das mesmas às categorias do modelo ICM, bem como a alteração e exclusão das mesmas.

6.3.2 Registro dos objetivos de mensuração

O registro dos objetivos de mensuração definidos para os projetos é realizado através das funcionalidades apresentadas na figura 6.4.

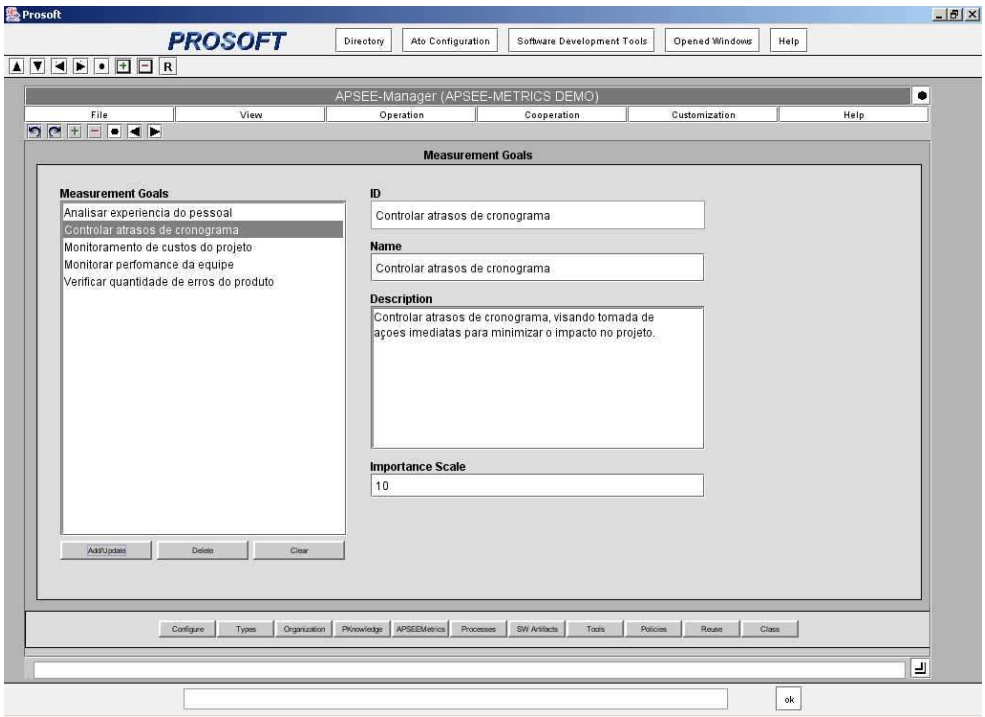


Figura 6.4: Definição dos objetivos de medição

As funcionalidades oferecidas permitem o registro de cada objetivo de mensuração, e de suas respectivas informações de contexto. A interface permite a inclusão de novos objetivos de mensuração, bem como a alteração e exclusão dos mesmos. Estes objetivos servirão futuramente de base para a elaboração do plano de mensuração.

6.3.3 Registro das informações críticas de projeto

Uma vez definidos os objetivos de mensuração, é necessário a definição das informações críticas de projeto, tais como problemas, riscos associados, necessidades de informação do projeto, etc. O registro das informações críticas definidos para os projetos é realizado através das funcionalidades apresentadas na figura 6.5.

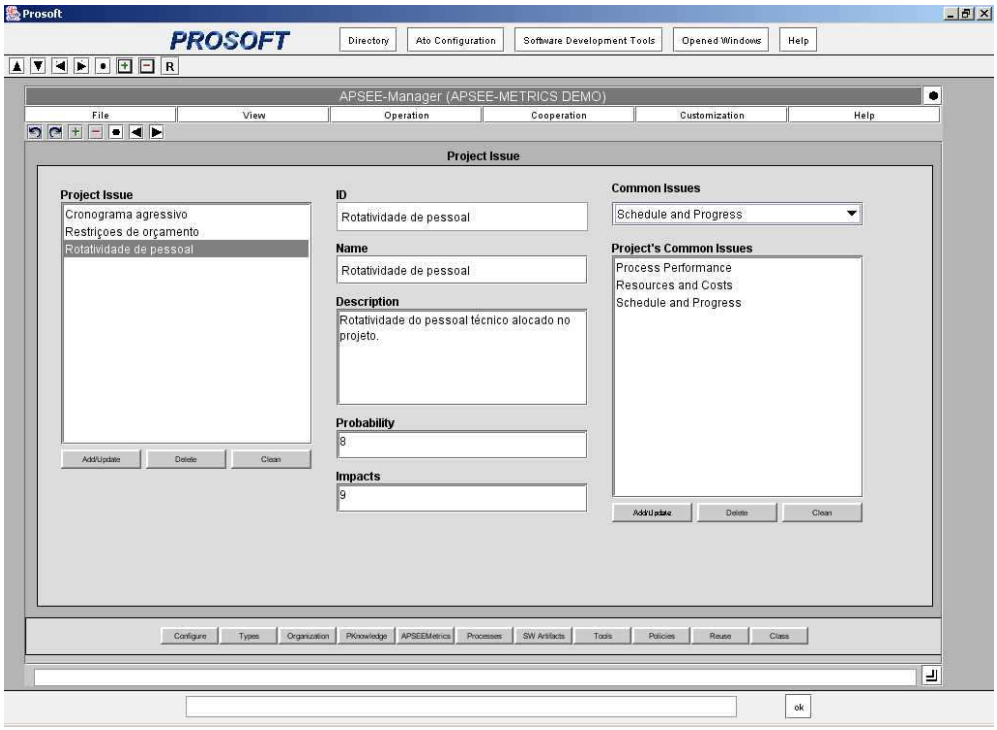


Figura 6.5: Definição das informações críticas de projeto

As funcionalidades oferecidas permitem o registro de cada informação crítica de projeto, e de suas respectivas informações de contexto. Além disto, existe a possibilidade de definição da probabilidade de ocorrência e o seu impacto para o projeto. A interface permite a inclusão de novas informações específicas de projeto, bem como a alteração e exclusão das mesmas. Outra funcionalidade é a associação de cada um dos registros às áreas comuns do modelo ICM, de forma a contextualizá-las, facilitando posteriormente as escolhas das métricas apropriadas para satisfazer estas necessidades de informação. Estas informações servirão futuramente de base para a elaboração do plano de mensuração.

6.3.4 Definição de métricas

A definição das métricas gerais do APSEE é realizada através das funcionalidades da interface apresentada na figura 6.6.

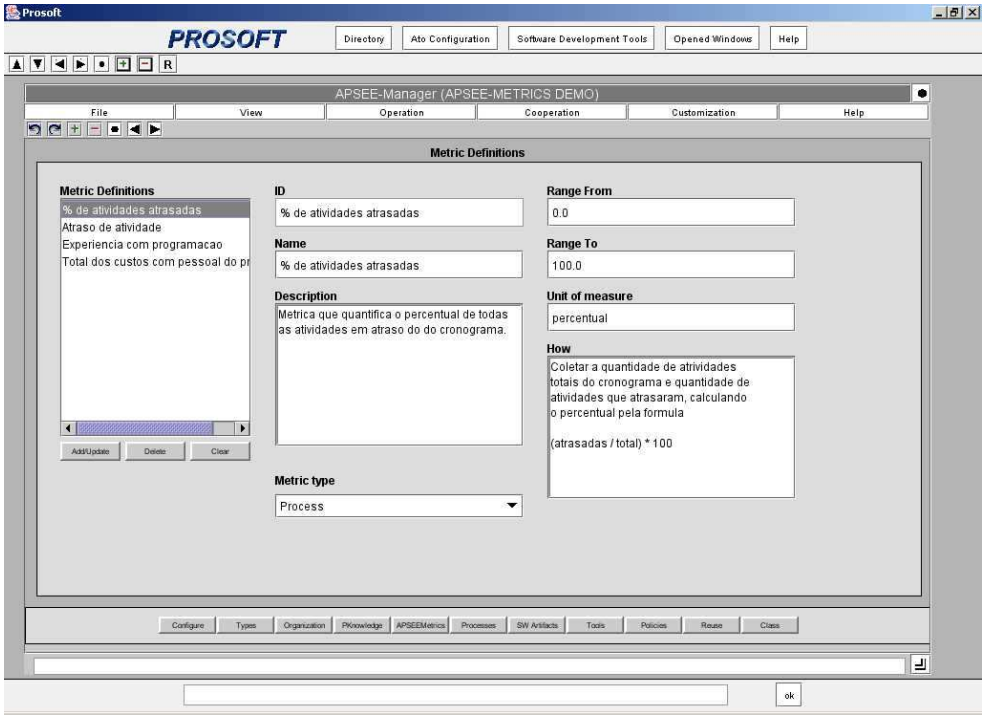


Figura 6.6: Definição das métricas

As funcionalidades oferecidas permitem o registro de cada métrica, onde são registradas além da identificação, nome e descrição da métrica, o seu tipo (que é definido segundo a hierarquia de tipos do APSEE, podendo ser parametrizado a qualquer momento), além das informações relativas à especificação como unidades de medida e a forma de coleta de cada métrica. A interface permite a inclusão de novas métricas, bem como a alteração e exclusão das mesmas.

6.3.5 Definição de análises

A definição das análises é realizada através das funcionalidades da interface apresentada na figura 6.7.

As funcionalidades oferecidas permitem o registro de cada análise a ser realizada, onde são registradas além da identificação e nome da análise, a sua especificação, que permite ao especialista um registro detalhado do que deve ser monitorado ou analisado na execução desta etapa. Além disto, para cada análise são associados os objetivos de mensuração, que devem orientar as análises realizadas, de forma que as mesmas satisfaçam os objetivos pré-estabelecidos. Por fim, os resultados obtidos com cada análise podem ser registrados, visando dar um *feedback* sobre a utilização da mesma no projeto. A interface permite a inclusão de novas análises, bem como a alteração e exclusão das mesmas.

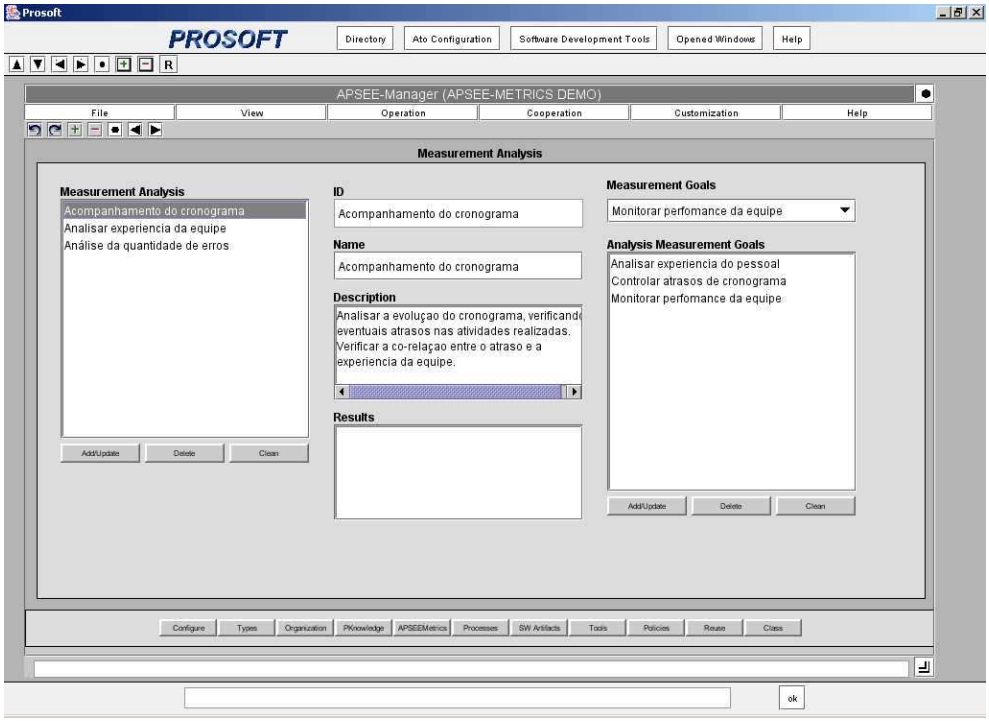


Figura 6.7: Definição das análises

6.3.6 Definição das avaliações

A definição das avaliações das métricas, das atividades, do processo de mensuração utilizado é realizada através das funcionalidades da interface apresentada na figura 6.8.

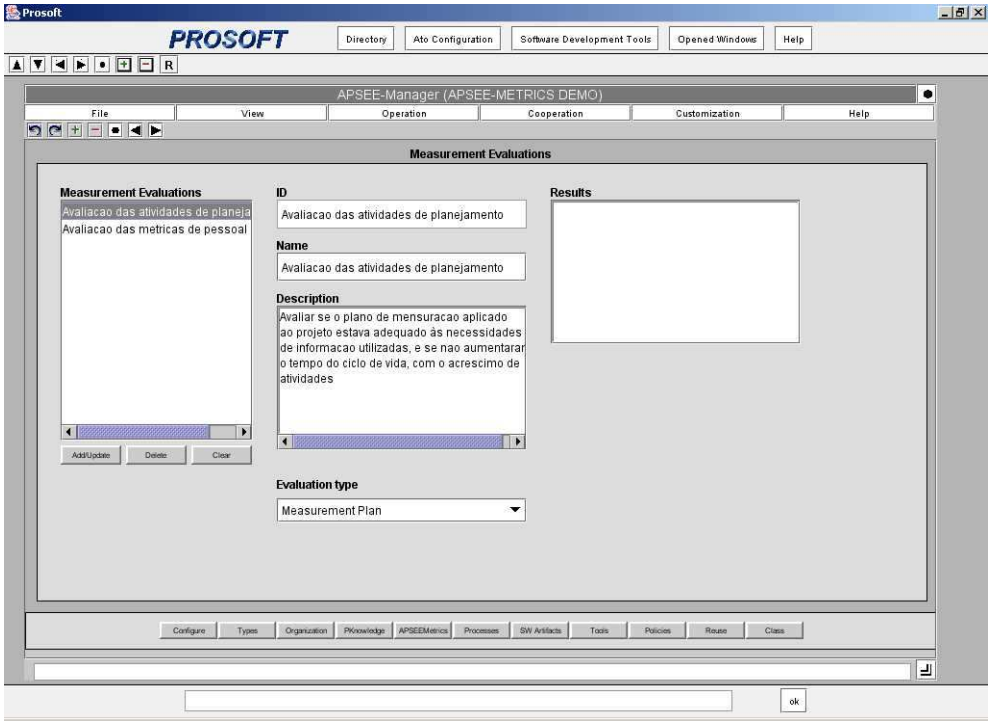


Figura 6.8: Definição das avaliações

As funcionalidades oferecidas permitem o registro de cada avaliação a ser realizada, onde são registradas além da identificação e nome da avaliação, a sua

especificação, que permite ao especialista um registro detalhado do que deve ser avaliado. Para cada avaliação é definido o seu tipo (que é definido segundo a hierarquia de tipos do APSEE, podendo ser parametrizado a qualquer momento). Inicialmente são definidos três tipos: Avaliação do plano de mensuração, das métricas e das atividades de mensuração. Por fim, os resultados obtidos com cada avaliação podem ser registrados, visando dar um *feedback* sobre a utilização das mesmas no projeto. A interface permite a inclusão de novas avaliações, bem como a alteração e exclusão das mesmas.

6.3.7 Definição do modelo de informação

O modelo de informação basicamente reúne a associação de todos os elementos definidos anteriormente, viabilizando as informações necessárias para a mensuração. Os registros relativos ao modelo de informação são realizados através das funcionalidades da interface apresentada na figura 6.9.

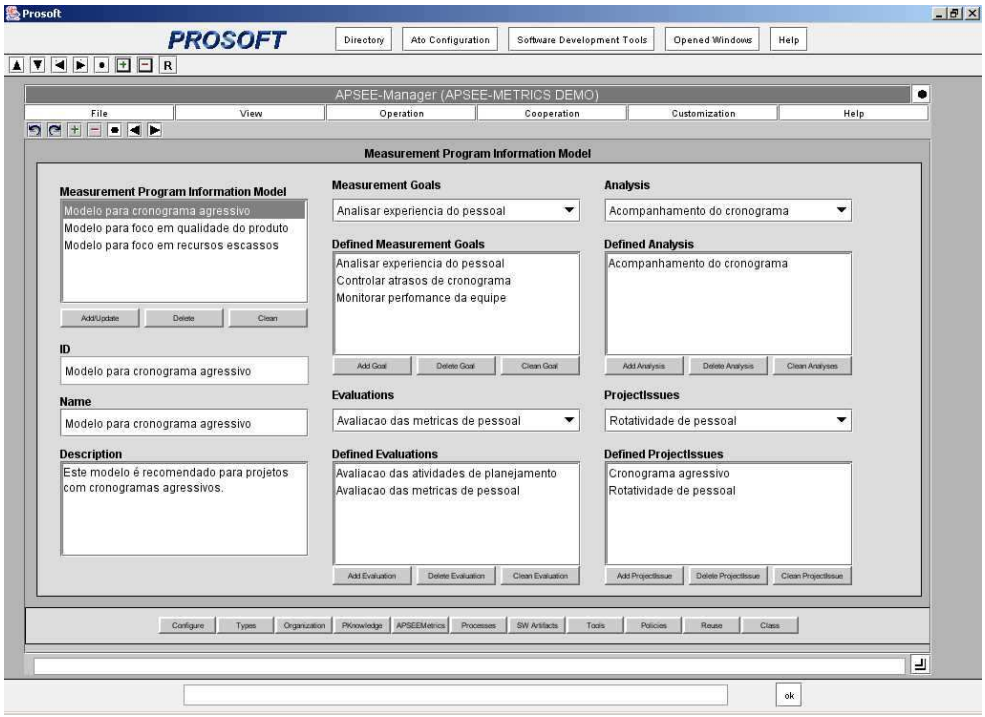


Figura 6.9: Definição do modelo de informação

As funcionalidades oferecidas permitem o registro de cada modelo de informação, cada qual permitindo ao usuário direcionar um foco específico para cada contexto de projeto, além do registro da identificação e nome do modelo, e uma descrição textual, que permite ao especialista um registro detalhado da aplicabilidade de cada modelo. Para cada modelo são definidos os alvos de mensuração, informações críticas do projeto análises e avaliações necessárias. A interface permite a inclusão de novos modelos, bem como a alteração e exclusão dos mesmos, podendo inclusive modificar os seus elementos a qualquer momento.

6.3.8 Definição do processo de mensuração

O processo de mensuração é definido utilizando a linguagem de modelagem de processos do APSEE. A definição de um processo de mensuração é realizada

através das funcionalidades da interface apresentada na figura 6.10, onde o processo é criado.

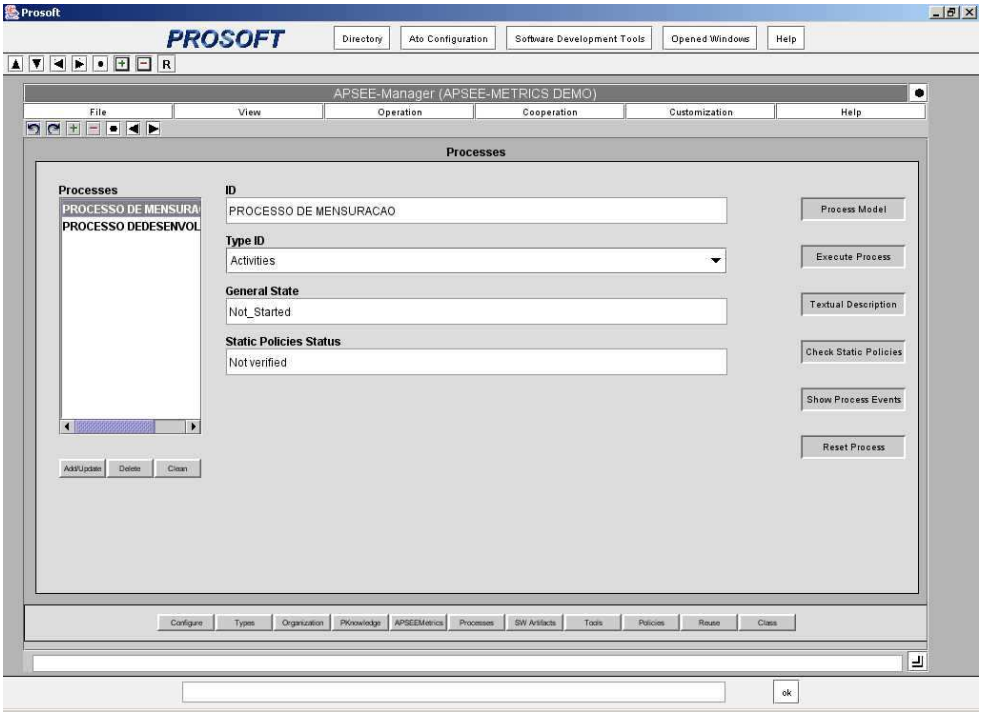


Figura 6.10: Definição do processo de mensuração

As funcionalidades oferecidas permitem a modelagem de um processo, bem como sua execução. Na figura 6.11 apresentamos um exemplo de modelagem de um processo de mensuração, com suas principais etapas (planejamento, coleta, análise e avaliação), bem como os artefatos gerados por cada atividade.

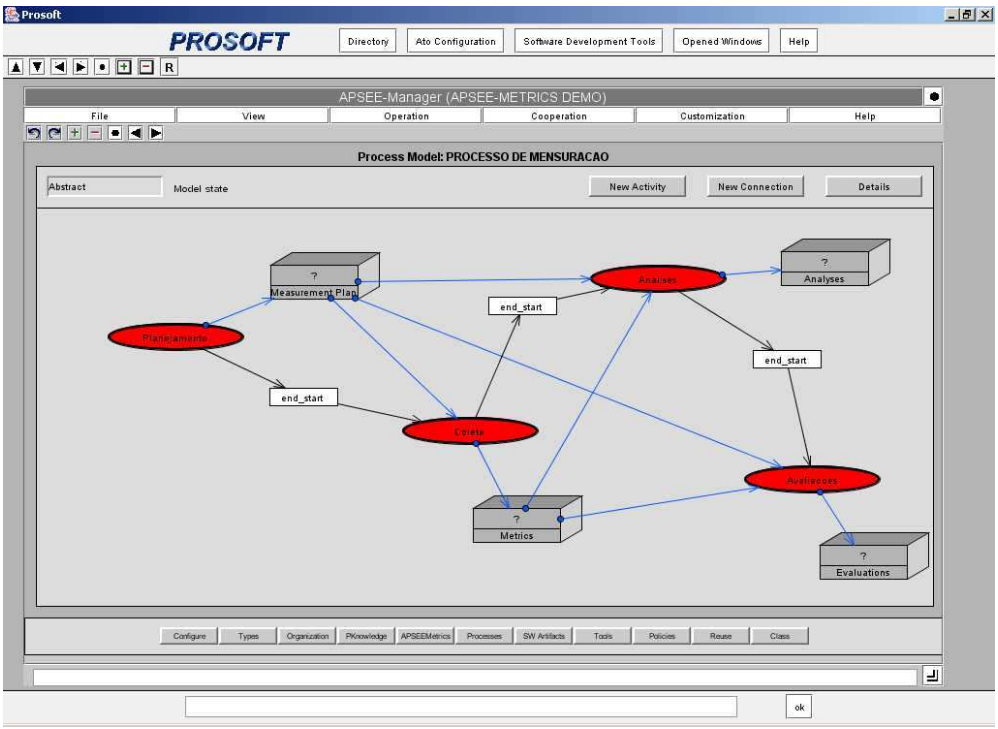


Figura 6.11: Modelagem do processo de mensuração

Cada etapa destas pode ser configurada como um atividade simples (*plain)* ou como um sub-processo (*fragment*), que pode ser definido de forma independente, como veremos na configuração das atividades apresentadas na figura 6.12. Além disto, cada atividade possui um tipo, que define se as atividades são de planejamento, coleta, análise ou avaliação. Estes tipos são pré-definidos na hierarquia de tipos do APSEE (*Apsee-types*), podendo ser incluídos novos tipos se necessário.

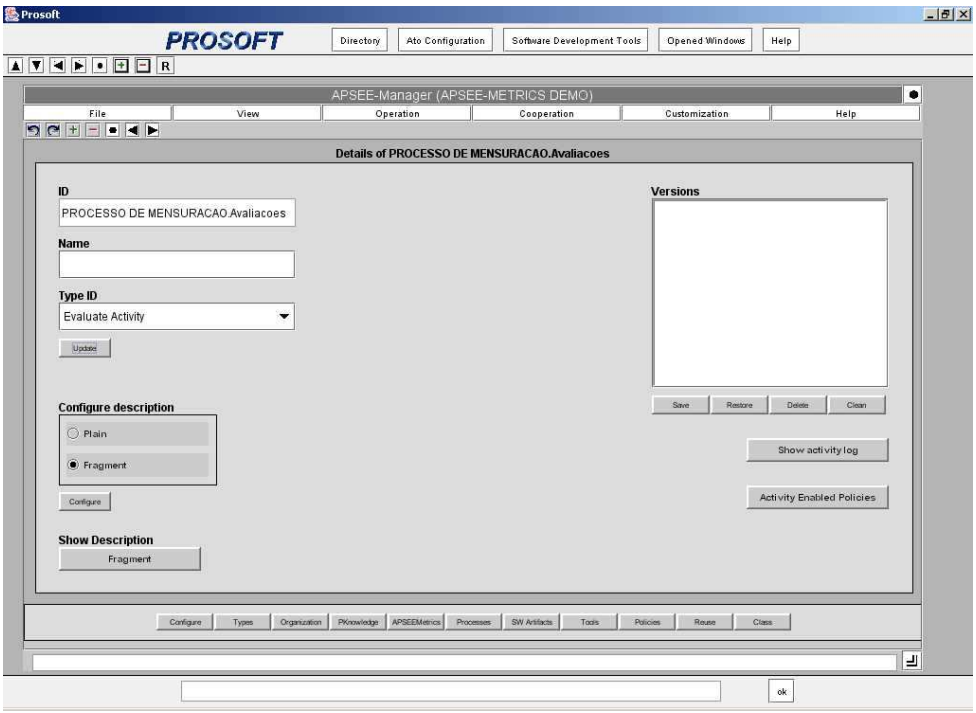


Figura 6.12: Configuração das atividades

Na figura 6.13 temos um exemplo de modelagem do sub-processo de planejamento, apresentando cada uma de suas atividades, bem como os artefatos que são produzidos e/ou utilizados em cada atividade, demonstrando as funcionalidades da ferramenta para modelagem das atividades. As elipses denotam as atividades ou sub-processos e os cubos denotam os artefatos produzidos e/ou consumidos pelas atividades. Ligações entre as atividades indicam a relação de conexões existente entre as mesmas, indicando o grau de dependência entre as mesmas (por exemplo: *end-start, start-start, end-end*).

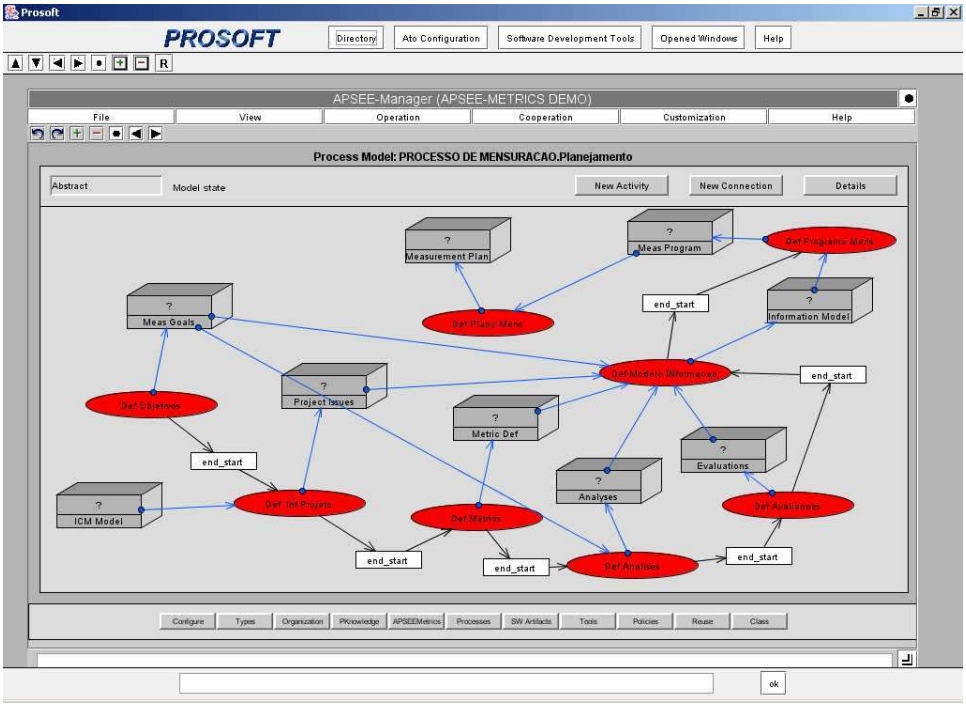


Figura 6.13: Modelagem das atividades de planejamento

De forma integrada ao processo de desenvolvimento, o especialista poderá definir o processo de mensuração na ferramenta de forma a combinar atividades habituais do ciclo de vida de desenvolvimento, com as atividades e sub-processos relativos à mensuração. Na figura 6.14 apresentamos um exemplo simples desta interação, na qual são definidas atividades de análise, projeto, desenvolvimento e testes, bem como as atividades relativas à mensuração, de forma integrada. As elipses em destaque denotam as atividades e sub-processos relativas ao processo de mensuração e as demais representam atividades do processo de desenvolvimento.

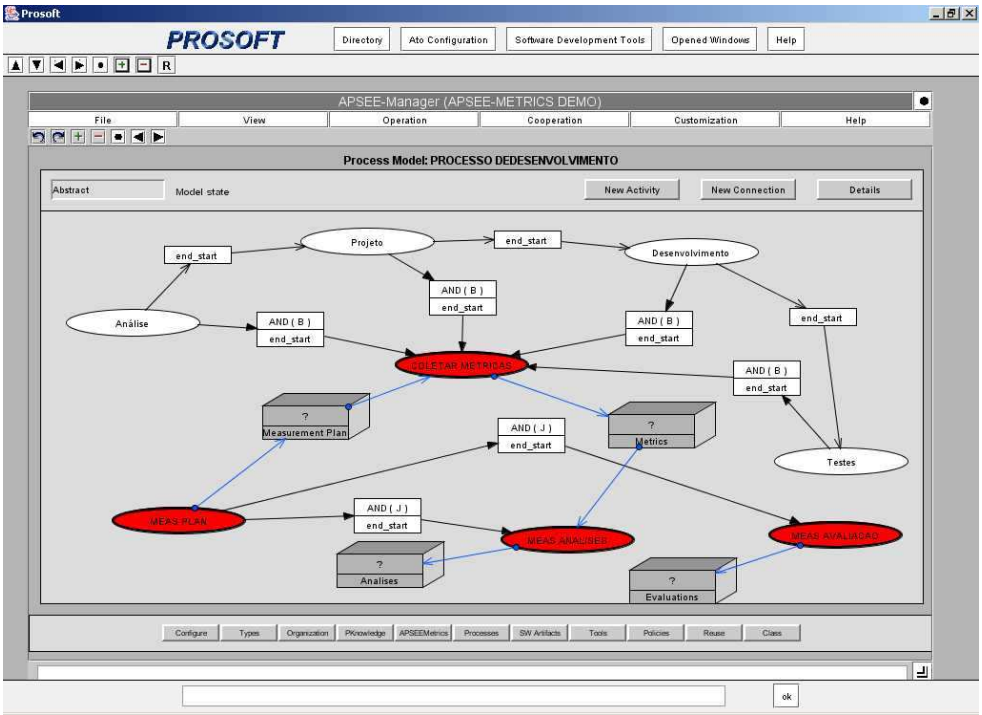


Figura 6.14: Modelagem integrada das atividades de desenvolvimento

6.3.9 Definição do programa de mensuração

O programa de mensuração faz a associação de um modelo de informação com um processo de mensuração definidos anteriormente, permitindo, por exemplo, que determinada organização possa, para cada plano de mensuração de um projeto, escolher o nível de mensuração ao qual ela deseja aplicar a cada contexto. Os registros relativos ao programa de mensuração são realizados através das funcionalidades da interface apresentada na figura 6.15.

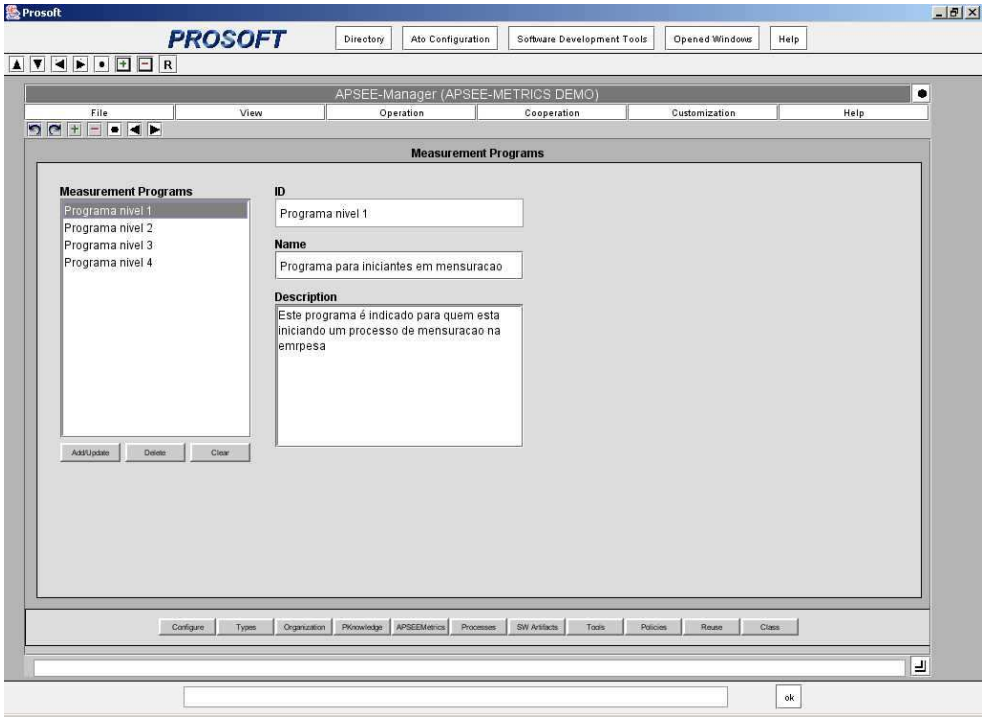


Figura 6.15: Definição do programa de mensuração

As funcionalidades oferecidas permitem o registro de cada programa de mensuração, cada qual permitindo ao usuário indicar o foco específico para diferentes contextos. Permitem o registro da identificação e nome do programa, e uma descrição textual, que possibilita ao especialista um registro detalhado da aplicabilidade de cada programa. Para cada programa são definidos o modelo de informação e o processo de mensuração a ser utilizado. A interface permite a inclusão de novos programas, bem como a alteração e exclusão dos mesmos.

6.3.10 Definição do plano de mensuração

O plano de mensuração faz a aplicação de um programa de mensuração (modelo de informação e de um processo de mensuração), definido anteriormente, para cada projeto. Os registros relativos ao plano de mensuração são realizados através das funcionalidades da interface apresentada na figura 6.16.

As funcionalidades oferecidas permitem o registro do plano de mensuração de um projeto. Permitem o registro da identificação e nome do plano, e uma descrição textual, que permite ao especialista um registro detalhado da aplicabilidade do plano. Para cada plano são definidos o contexto onde está sendo aplicado, bem como informações relevantes relativas ao projeto e a aplicabilidade do plano de mensuração, sendo também definido o programa de mensuração que está sendo

utilizado no projeto. A interface permite a inclusão de novos planos, bem como a alteração e exclusão dos mesmos.

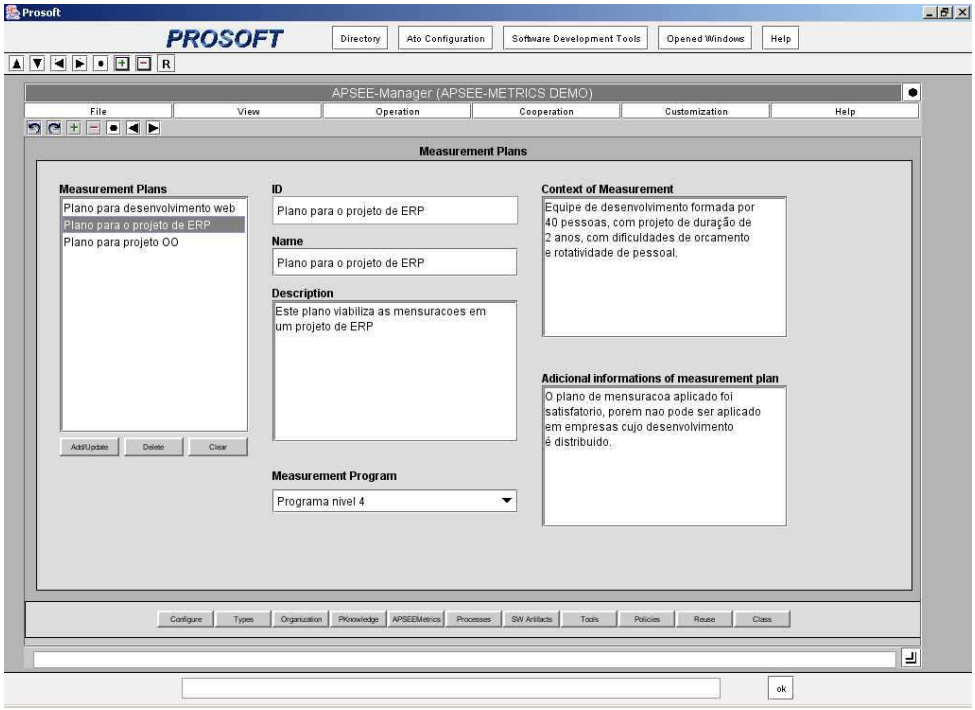


Figura 6.16: Definição do plano de mensuração

6.4 Considerações sobre o capítulo

Nesta seção apresentamos o protótipo da ferramenta APSEE-*Metrics,* integrada ao ambiente APSEE. A ferramenta desenvolvida proporcionou a materialização de algumas das funcionalidades propostas no modelo, demonstrando sua viabilidade. Apesar de ser uma implementação limitada de seus componentes, o protótipo desenvolvido permitiu visualizar a aplicabilidade da abordagem proposta, demonstrando também que a integração com o ambiente de desenvolvimento de *software* APSEE-Prosoft é viável, e permitiu agregar novas funcionalidades importantes ao ambiente, complementando-o em uma das áreas importantes da Engenharia de *Software*.

Na próxima seção será apresentado um exemplo prático de utilização do modelo, demonstrando sua utilização com base em um contexto real de uma organização desenvolvedora de *software.*

7 EXEMPLO DE UTILIZAÇÃO

Este capítulo apresenta um exemplo de utilização da abordagem proposta neste trabalho, utilizando como base um cenário real de uma organização desenvolvedora de *software*. Nesta seção serão apresentados primeiramente os objetivos deste estudo de caso, as características do cenário de aplicação, posteriormente a aplicação da abordagem e por fim as conclusões e resultados obtidos.

7.1 Objetivos

O objetivo geral deste exemplo consiste basicamente em verificar a aplicabilidade da abordagem proposta, através de um exemplo de utilização aplicado a um cenário real de utilização. Para isto são objetivos específicos:

- Executar as etapas de planejamento previstas na abordagem, analisando potenciais facilidades e dificuldades existentes em sua aplicação;
- Identificar pré-condições necessárias para sua implantação.
- Analisar a aplicabilidade da abordagem;

Desta forma, a sistemática de aplicação foi dividida nas seguintes etapas:

- Identificação do cenário de aplicação, através da caracterização da organização, do ambiente e do processo de desenvolvimento utilizados;
- Identificação das necessidades de informação desta organização e sistemática atual de coleta das informações;
- Aplicação do APSEE-*Metrics* a um projeto específico desta organização;
- Análise geral da aplicação da abordagem.

Nas seções seguintes apresentaremos cada uma destas etapas em maiores detalhes.

7.2 Cenário de aplicação

A organização utilizada para este exemplo consiste no departamento de Sistemas de Informação de um Centro Universitário, pertencente a uma rede privada de ensino. O departamento em questão é responsável pelo desenvolvimento de um sistema para gestão acadêmica de escolas e universidades

integrantes da rede de ensino a qual faz parte. A escolha da organização foi baseada em questões de acessibilidade e disponibilidade das informações necessárias para a aplicação da abordagem.

O departamento caracteriza-se pelo desenvolvimento de aplicações *Web,* utilizando como principais recursos tecnológicos para o desenvolvimento as linguagens PHP, Javascript e HTML, bem como um *framework* próprio desenvolvido pela equipe (baseado em classes e utilizando conceitos de orientação a objetos (GARCIA JUNIOR, 2005)), além de um Sistema Gerenciador de Banco de Dados Relacional (SGBD) Oracle. As aplicações desenvolvidas pela equipe são voltadas ao atendimento das demandas internas dos diversos setores das instituições usuárias de suas aplicações, sendo utilizadas por uma escola e dois Centros Universitários da rede, totalizando mais de 10.000 usuários internos e externos.

Em relação à equipe de desenvolvimento, a mesma está estruturada com uma coordenação, responsável pela gestão do processo de desenvolvimento e dos projetos, analistas e programadores, totalizando sete pessoas envolvidas no desenvolvimento.

Com relação ao processo de desenvolvimento, antes da aplicação deste estudo não existia um modelo de processo definido para a organização em questão. Desta forma um modelo de processo de software teve de ser definido para que este exemplo pudesse ser aplicado, sendo descrito posteriormente.

7.2.1 Necessidades de informação e sistemática atual de coleta

Atualmente são realizadas algumas medições durante o processo de desenvolvimento. Estas medições são realizadas de forma não sistemática e sem um foco específico. Também não é utilizada nenhuma ferramenta específica para o processo de mensuração. Alguns dados coletados pela equipe referem-se principalmente ao número de atendimentos de suporte realizados, número de erros contabilizados por mês e principais setores que demandam tarefas para desenvolvimento. Estas informações são registradas em um sistema de controle de atendimentos (utilizado para fins de contabilização do número de atividades realizadas), mas que não contempla o gerenciamento de métricas ou de um programa de mensuração.

Há uma carência específica quanto a metodologia de mensuração, bem como do padrão de coleta e análise dos dados obtidos.

Em termos das necessidades de informação, de maneira geral, possuem importante relevância para tomada de decisão pela coordenação da equipe os seguintes aspectos (principais):

- Produtividade da equipe;
- Qualidade do produto desenvolvido;
- Prazos de entrega das *releases* planejadas.

Estas necessidades foram utilizadas como base para a realização deste exemplo de uso, como veremos nas seções que seguem.

7.3 Aplicação da abordagem de mensuração

Com base nas necessidades de informação sobre o processo de desenvolvimento apresentadas anteriormente, a abordagem proposta foi aplicada, sendo apresentadas nas sub-seções à seguir as etapas relativas ao planejamento.

7.3.1 Definição do plano de mensuração

Inicialmente, foi definido um novo plano de mensuração, onde foram especificadas as informações relativas ao contexto do projeto, do processo de desenvolvimento utilizado e informações relativas à organização onde o processo está inserido. A tabela abaixo apresenta as informações referentes a este plano.

Tabela 7.1 : Informações gerais do plano de mensuração

Informações gerais do plano de mensuração	
Objeto	**Contextualização**
Projeto	O projeto refere-se ao desenvolvimento de um módulo para controle de caixa (tesouraria) para um sistema de gestão acadêmica, envolvendo aplicativos para recebimento de mensalidades, cálculo de juros, recebimento de cheques, controle geral de caixas, entre outras informações.
Processo de desenvolvimento	O processo de desenvolvimento utilizado caracteriza-se por utilizar o ciclo de vida clássico de desenvolvimento, com pequenas adaptações particulares à organização.
Organização	A organização desenvolvedora é a unidade de sistemas de informação de um Centro Universitário, com equipe de desenvolvedores composta por 7 pessoas, sendo 2 analistas e 5 programadores, onde 1 dos analistas é o coordenador técnico da equipe.
Pontos críticos	Prazos de entrega do aplicação relativamente reduzidos em função do tamanho da equipe. Outras atividades paralelas que demandam o compartilhamento do tempo de alguns desenvolvedores com outras atividades.

A figura 7.1 ilustra o registro do plano no protótipo do APSEE-*Metrics*.

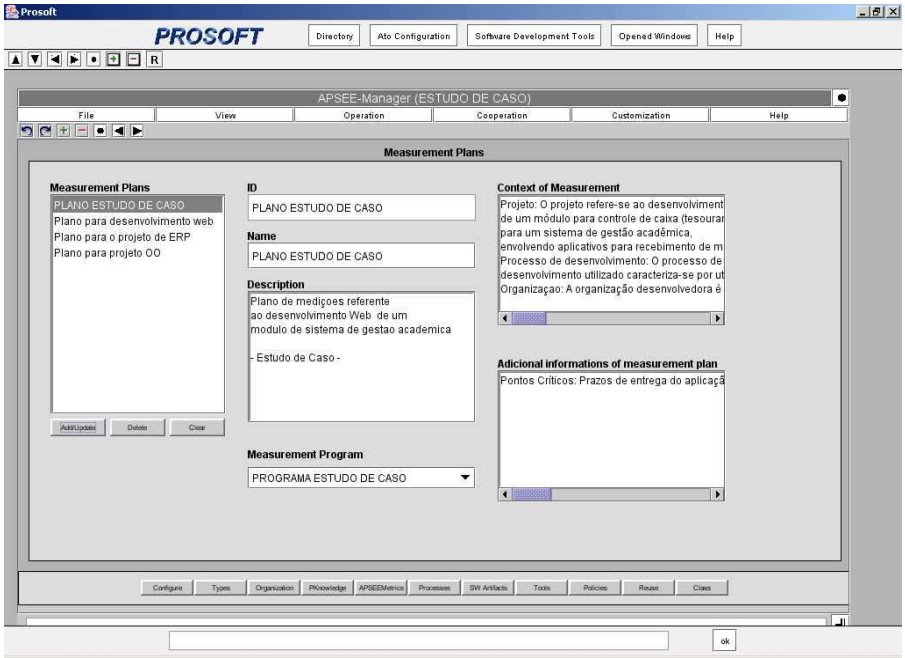


Figura 7.1: Definição do plano de mensuração no APSEE

Definidas as informações iniciais do plano de mensuração, a próxima etapa consiste na definição do programa de mensuração, envolvendo a criação do modelo de informação e o processo de mensuração.

7.3.2 Definição do modelo de informação

A definição do modelo de informação discorre com a execução das seguintes etapas:

- identificação dos objetivos de mensuração do projeto;
- identificação das informações críticas do projeto (tais como riscos e problemas associados) e priorização das mesmas;
- seleção das medições genéricas;
- definição e especificação das métricas;
- definição e especificação das análises a serem realizadas;
- definição e especificação das avaliações a serem realizadas.

A execução de cada uma das etapas apresentadas acima, são descritas de forma sucinta como segue:

a) *Identificação dos objetivos de mensuração do projeto:*

Para o modelo de informação há necessidade de definição dos objetivos de mensuração, que para este projeto foram identificados pelo coordenador da equipe de desenvolvimento, sendo apresentados na tabela 7.2:

Tabela 7.2: Objetivos de mensuração do projeto

Objetivos de mensuração do projeto	Escala de Importância
Monitorar o atingimento dos prazos dos marcos de projeto	10
Acompanhar o desempenho da equipe em termos de produtividade	9
Controlar a qualidade do produto desenvolvido, minimizando o número de erros dos programas entregues.	7
Monitorar o tempo gasto com atividades de suporte paralelas ao projeto	6

A figura 7.2 ilustra o registro dos objetivos de mensuração no protótipo do APSEE-*Metrics.*

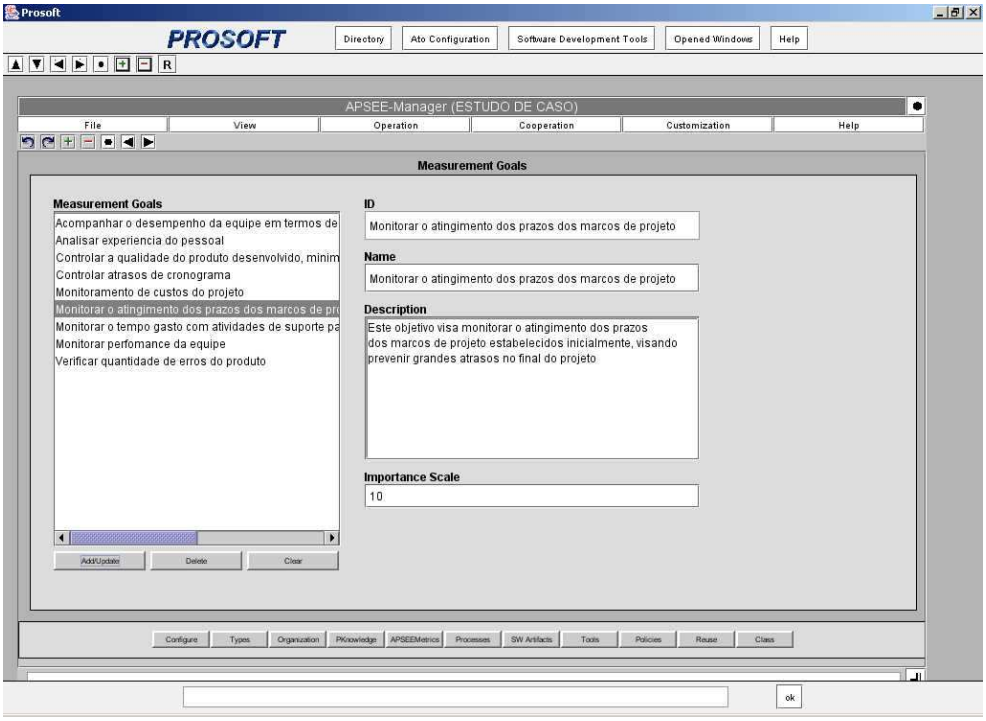


Figura 7.2: Definição dos objetivos de mensuração no APSEE

b) *Identificar informações críticas do projeto:*

Com base nos objetivos de medição definidos, foram identificados os itens relativos aos problemas, riscos e informações necessárias a serem obtidas com a mensuração (conhecidos como *issues* no PSM). Após mapeadas as informações críticas, as mesmas foram priorizadas, segundo uma probabilidade de ocorrência (que observa uma escala de 0 a 1), bem como o impacto no projeto (escala de 1 a 10), obtendo-se a exposição do projeto em relação a cada uma das informações críticas do projeto. Estas informações são apresentadas na tabela 7.3:

Tabela 7.3: Informações críticas do projeto

Informações críticas do projeto	Probabilidade de ocorrência	Impacto no projeto	Exposição do projeto
Cronograma agressivo	1,0	10	10
Falta de produtividade da equipe	0,8	9	7,2
Requisitos instáveis	0,7	9	6,3
Atividades de suporte paralelas ao projeto	1	10	10

Com base no modelo ICM (*Issue-Category-Measures*), foi realizado o mapeamento das informações críticas do projeto para as áreas comuns do modelo (*Common Issues*), de forma a auxiliar na identificação das medições apropriadas para as necessidades de informação apresentadas como sendo necessárias pela organização em questão. A tabela 7.4 apresenta as informações mapeadas.

Tabela 7.4: Mapeamento das informações críticas do projeto para o modelo ICM

Informações críticas do projeto	Áreas comuns do modelo ICM
Cronograma agressivo	Cronograma e progresso
Falta de produtividade da equipe	Desempenho do processo
Requisitos instáveis	Tamanho e estabilidade do produto
Atividades de suporte paralelas ao projeto	Desempenho do processo

A figura 7.3 ilustra o registro dos objetivos de mensuração no protótipo do APSEE-*Metrics*.

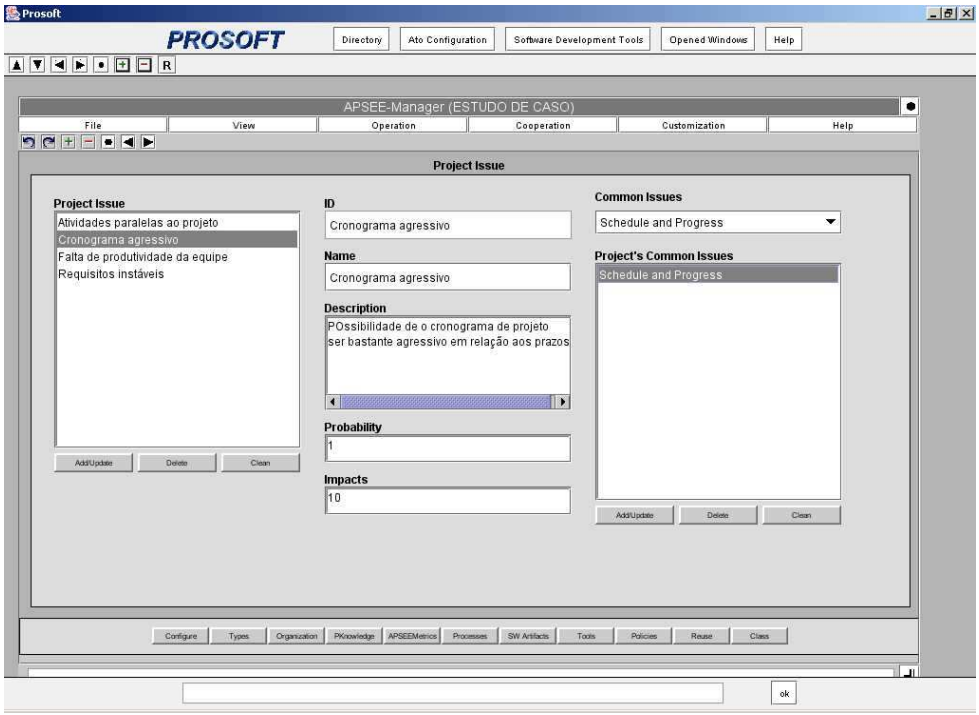


Figura 7.3: Definição das informações críticas do projeto no APSEE

c) *Seleção das medições genéricas*

Uma vez identificadas as necessidades específicas de medição do projeto, à partir do mapeamento realizado ao modelo ICM foi possível derivar as medições necessárias para coleta dos dados necessários. Desta forma, primeiramente foram derivadas as categorias de cada área (*Measurement Category*), e por fim selecionadas as medições genéricas (*Generic Measures*) a serem utilizadas. Na tabela 7.5 são apresentadas as derivações realizadas.

Tabela 7.5: Definição das medições do projeto

Informações críticas do projeto	Áreas comuns do modelo ICM	Categorias de medição	Medições
Cronograma agressivo	Cronograma e progresso	Progresso das unidades de trabalho	Datas dos marcos de projeto
Falta de produtividade da equipe	Recursos e custos	Pessoal	Esforço Experiência da equipe
Requisitos instáveis	Tamanho e estabilidade do produto	Tamanho e estabilidade funcional	Requisitos
Atividades de suporte paralelas ao projeto	Desempenho do processo	Eficiência do projeto	Produtividade Tempo de ciclo

d) *Definição e especificação das métricas:*

Após obtidas as medições genéricas, foram definidas as métricas a serem utilizadas, sendo estas métricas associadas a cada medição genérica do modelo ICM. Na tabela 7.6 são apresentadas as métricas definidas para cada uma das medições genéricas derivadas do modelo ICM.

Tabela 7.6: Definição das métricas

Medições Genéricas do modelo ICM	Métricas
Datas dos marcos de projeto	- Dias de atraso de cada atividade - % de atividades concluídas após o prazo previsto
Esforço	- Número de funcionários do projeto - Número total de horas trabalhadas no projeto por funcionário
Experiência da equipe	- Anos de experiência na equipe de cada funcionário - Anos de experiência externa de cada funcionário
Requisitos	- Número de requisitos iniciais - Número de requisitos modificados - Número de novos requisitos
Produtividade	- Número de horas de trabalho por atividade - Número de horas previstas por atividade
Tempo de ciclo	- Dias adicionais de projeto - Número de atividades de suporte executadas - Total de horas gastas em atividades fora do projeto

A tabela 7.7 apresenta a especificação das métricas definidas.

Tabela 7.7: Especificação das métricas

Nome	Descrição	Tipo	Unidade de medida	Valor mín.	Valor máx.	Modo de coleta
Dias de atraso de cada atividade	Identificar os dias de atraso de cada atividade prevista no projeto	Atividade	Dias	0	999	Coletar datas de início e término de cada atividade (previsto e realizado)
% de atividades concluídas fora do prazo previsto	Identificar quantas atividades foram concluídas fora do prazo previsto	Atividade	Percentual	0	100	Coletar datas de início e término de cada atividade (previsto e realizado)
Número de funcionários do projeto	Quantidade de funcionários da área técnica envolvidos no processo	Recursos	Número de pessoas	0	999	Coletar número de funcionários envolvidos em atividades do projeto
Número total de horas trabalhadas no projeto por funcionário	Somatório de horas trabalhadas por funcionários em atividades do projeto	Processo	Número de horas	0	999	Coletar número de horas de cada atividade executada no projeto
Anos de experiência na equipe de cada funcionário	Experiência de trabalho na equipe (em anos) de cada funcionário	Pessoas	Anos	0	999	Coletar anos de experiência na equipe de cada funcionário
Número de requisitos iniciais	Número total de requisitos no início do projeto	Artefatos	Quantidade	0	999	Coletar quantidade de requisitos definidos no início do projeto
Número de requisitos modificados	Número total de requisitos modificados durante o projeto	Artefatos	Quantidade	0	999	Coletar quantidade de requisitos alterados durante o projeto
Número de novos requisitos	Número total de requisitos incluídos durante o projeto	Artefatos	Quantidade	0	999	Coletar quantidade de requisitos incluídos durante o projeto
Número de horas de trabalho por atividade	Número total de horas de trabalho despendidas em cada atividade	Atividade	Número de horas	0	999	Coletar número de horas efetivas de trabalho em cada atividade
Número de horas previstas por atividade	Número total de horas de trabalho previstas para cada atividade no início do projeto	Atividade	Número de horas	0	999	Coletar número de horas previstas para cada atividade
Dias adicionais de projeto	Total de dias adicionais utilizados para conclusão do projeto	Processo	Dias	0	999	Coletar data de início e término do projeto (previsto e realizado)
Número de atividades de suporte executadas	Total de atividades de suporte paralelas ao projeto que foram executadas durante o mesmo período do projeto	Processo	Número de atividades	0	999	Coletar número de atividades de suporte realizadas que não são do projeto, no período do mesmo
Total de horas gastas em atividades de suporte	Total de horas gastas em atividades de suporte paralelas ao projeto que foram executadas durante o mesmo período do projeto	Processo	Horas	0	999	Coletar número de horas gastas em atividades de suporte realizadas que não são do projeto, no período do mesmo

A figura 7.4 ilustra o registro das métricas e suas especificações no protótipo do APSEE-*Metrics.*

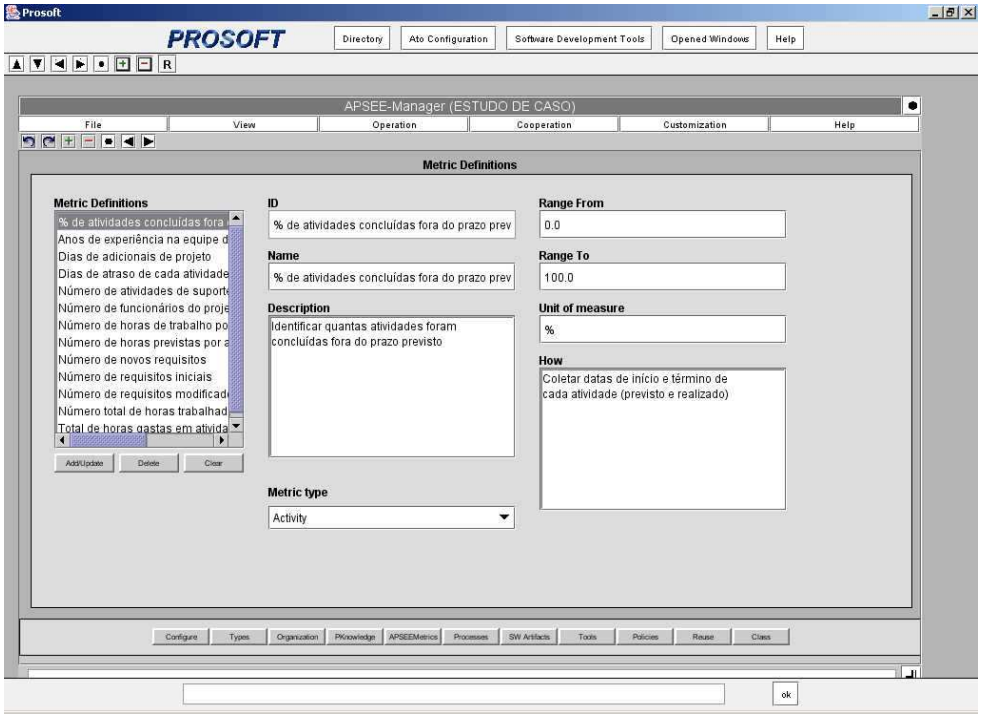


Figura 7.4: Definição e especificação das métricas no APSEE

e) *Definição e especificação das análises:*

Uma vez definidas as métricas, o modelo proposto prevê a definição das análises a serem realizadas, de forma a promover a utilização efetiva dos dados coletados no processo de mensuração, bem como a co-relação com as informações críticas do projeto e os objetivos de mensuração, definidos anteriormente. Assim, algumas análises necessárias foram especificadas para o referido projeto, sendo apresentadas na tabela 7.8.

Tabela 7.8: Definição e especificação das análises

Análises	Objetivos de mensuração associados	Informações críticas do projeto relacionadas	Métricas associadas	Descrição da análise
Acompanhamento das atividades	Monitorar o atingimento dos prazos dos marcos de projeto	- Cronograma agressivo - Falta de produtividade da equipe - Requisitos instáveis - Atividades de suporte paralelas ao projeto	- Dias de atraso de cada atividade - % de atividades concluídas após o prazo previsto - Número de requisitos modificados - Número de novos requisitos	Analisar o percentual de atividades concluídas após o prazo, verificando os dias de atraso de cada atividade. Nos casos onde houve atraso, verificar se houveram mudanças nos requisitos ou atividades paralelas para o responsável pela atividade. Além disto, verificar o número de novos requisitos do projeto.
Monitoramento da equipe	Acompanhar o desempenho da equipe em termos de produtividade	- Falta de produtividade da equipe - Requisitos instáveis	- Dias de atraso de cada atividade - Número de requisitos modificados - Anos de experiência na equipe de cada	Correlacionar os dias de atraso de cada atividade com as mudanças de requisitos, com a experiência de trabalho dos funcionários, bem como o total de horas gastas com atividades paralelas ao projeto, analisando

			funcionário - Anos de experiência externa de cada funcionário - Total de horas gastas em atividades fora do projeto	a causa de falta de produtividade, quando houver.
Analisar o tempo gasto com atividades paralelas	Monitorar o tempo gasto com atividades paralelas ao projeto	- Atividades de suporte paralelas ao projeto	- Dias de projeto não previstos - Número de atividades fora do projeto executadas - Total de horas gastas em atividades fora do projeto	Analisar o impacto das atividades paralelas executadas durante o projeto, relacionando eventuais atrasos no cronograma em função das atividades paralelas

A figura 7.5 ilustra o registro das análises e suas especificações no protótipo do APSEE-*Metrics*.

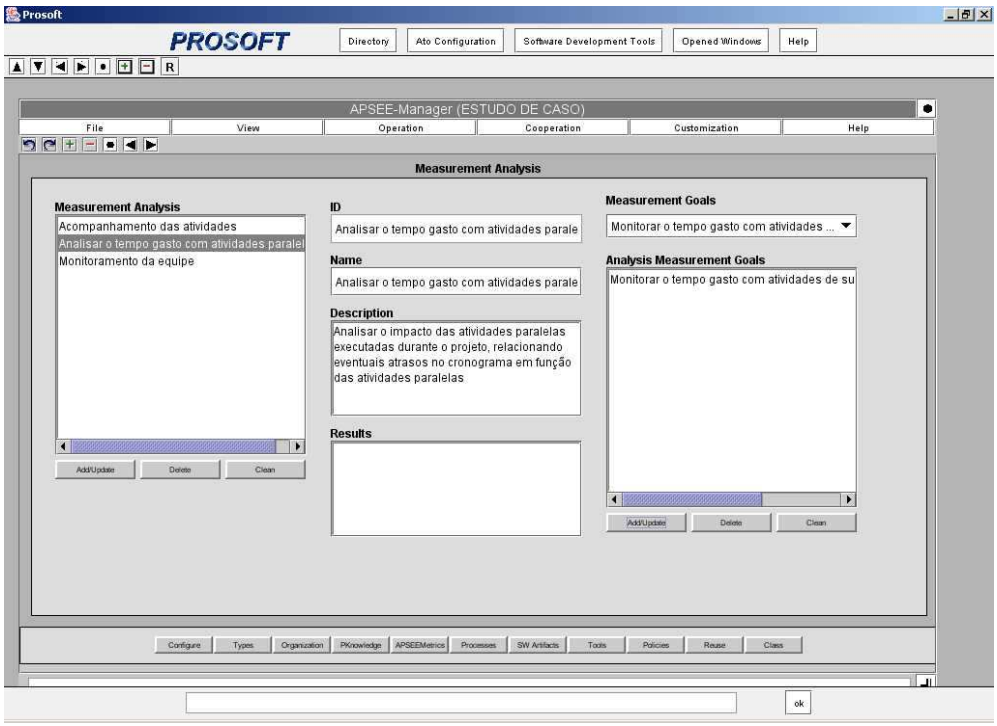


Figura 7.5: Definição e especificação das análises no APSEE

f) *Definição e especificação das avaliações:*

Nesta etapa foram definidas as avaliações a serem realizadas, de forma a promover a avaliação das métricas, do processo de mensuração e do programa de mensuração como um todo. Assim, as avaliações a serem realizadas foram especificadas para o referido projeto, sendo apresentadas na tabela 7.9.

Tabela 7.9: Definição e especificação das avaliações

Avaliação	Tipo de avaliação	Descrição
Avaliação do processo de mensuração	Atividades de mensuração	Avaliar a execução do processo de mensuração, apontando dificuldades existentes em relação às atividades de mensuração definidas no processo.
Avaliação das métricas	Métricas	Avaliar se a utilização de cada métrica foi satisfatória, apresentando dificuldades existentes na coleta e utilização de cada uma das métricas, visando definir a continuidade de sua utilização em projetos similares.
Avaliação do plano de mensuração	Plano de mensuração	Avaliar como um todo, o plano de mensuração utilizado, em relação a sua aplicabilidade, correlacionando o modelo de informação, o processo de mensuração, e as características do projeto em questão, visando avaliar a aplicabilidade do programa de mensuração para projetos da mesma característica.

A figura 7.6 ilustra o registro das avaliações e suas especificações no protótipo do APSEE-*Metrics.*

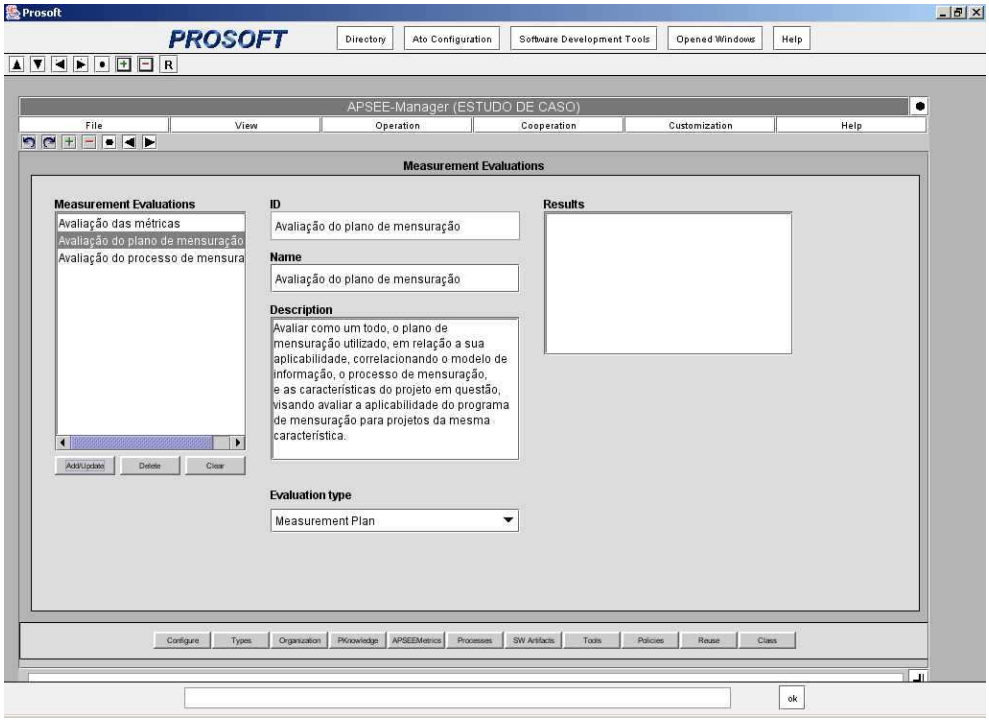


Figura 7.6: Definição e especificação das avaliações no APSEE

g) *Criação do plano de mensuração:*

Tendo sido definidos todos os elementos do modelo de informação, estes elementos foram então associados, conforme mostra a figura 7.7.

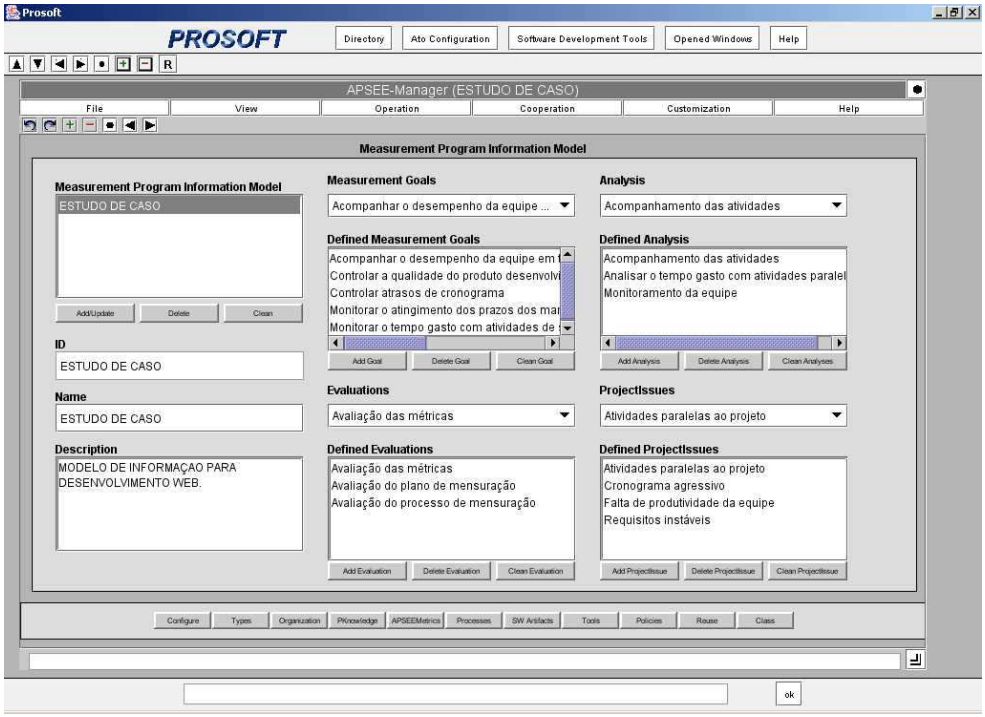


Figura 7.7: Definição do modelo de informação no APSEE

7.3.3 Definição do processo de mensuração

A definição do processo de mensuração discorre com a execução das seguintes etapas:

- Definição das atividades de coleta dos dados;
- Definição das atividades de execução das análises previstas;
- Definição das atividades das avaliações previstas;

Assim, as atividades do processo de mensuração foram modeladas, sendo definidos quatro sub-processos, planejamento, coleta, análise e avaliação, cada qual tendo suas atividades específicas detalhadas posteriormente. Na figura 7.8 é apresentado a modelagem das atividades principais, utilizando o APSEE, na qual apresentamos as atividades e sub-processos, bem como os artefatos produzidos pelos mesmos.

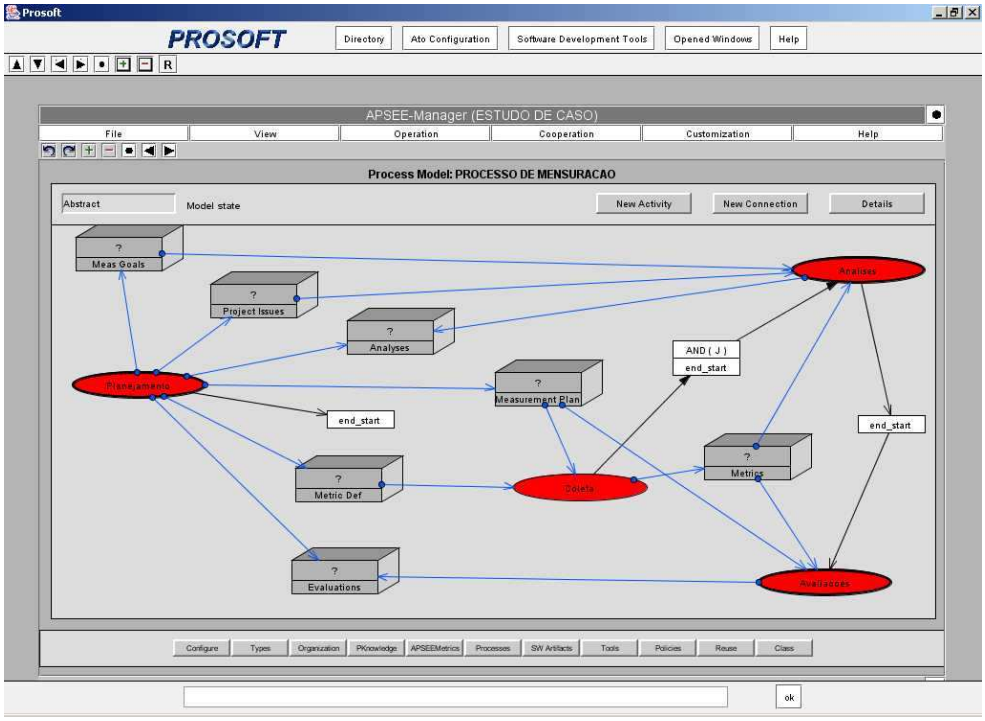


Figura 7.8: Definição do processo de mensuração

Detalhando a modelagem acima, na figura 7.9 é apresentado a modelagem das atividades referentes ao sub-processo de planejamento, onde foram definidas as etapas previstas na abordagem proposta neste trabalho. São acrescentados os artefatos a serem produzidos, detalhando cada uma das etapas a serem executadas.

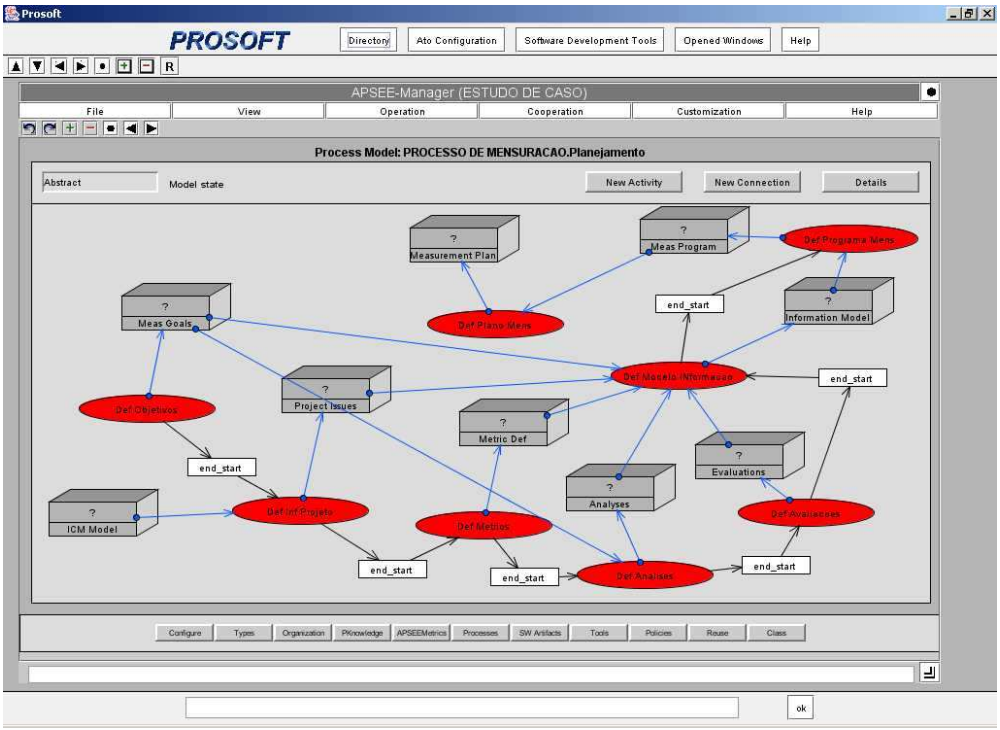


Figura 7.9: Definição do processo de planejamento

Como antes da aplicação deste estudo não existia um modelo de processo definido para a organização em questão, um um modelo de processo de software

teve de ser definido para que este exemplo pudesse ser aplicado. Ainda que de forma simples, o processo definido permitiu que as etapas do processo de mensuração fosse integrado, através dos sub-processos e atividades de mensuração apresentados anteriormente. A figura 7.10 apresenta esta integração utilizando o APSEE.

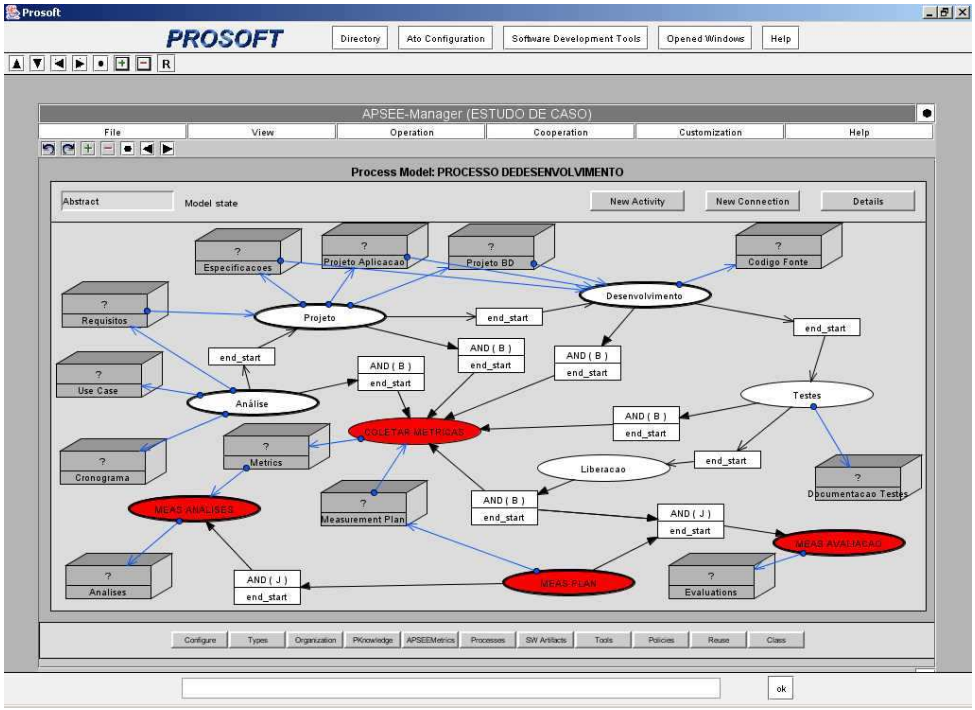


Figura 7.10: Definição e modelagem do processo de desenvolvimento

Os detalhamento dos demais sub-processos modelados encontram-se no apêndice B deste trabalho.

7.4 Análise dos resultados

O exemplo apresentado neste capítulo permitiu demonstrar a aplicabilidade do modelo e da ferramenta proposta, atendendo aos objetivos propostos inicialmente, possibilitando a realização de algumas análises em relação às suas características. Cabe ressaltar que o mesmo não permitiu a obtenção de uma avaliação muito criteriosa quanto a eficiência do modelo e da ferramenta, o que, mesmo não estando nos objetivos deste exemplo, entende-se como necessário futuramente.

No exemplo apresentado, foi possível observar que a organização pôde definir, de forma sistemática, a condução da definição de métricas e das atividades necessárias para proporcionar os dados relevantes para tomada de decisão. Uma vez que a organização não possuia nenhum tipo de padronização, metodologia ou critério para definição dos dados a serem coletados acerca do processo de desenvolvimento, a simples definição dos passos necessários para realização das mensurações já proporcionou um importante ganho para a organização em termos de padronização e foco nas mensurações. Além disto, a possibilidade de registro das informações relativas ao plano de mensuração também é um importante avanço em termos de controle. A pré-definição das análises a serem realizadas pode

proporcionar maior orientação e controle em relação a utilização dos dados coletados, uma vez que há um direcionamento mais específico das análises, que antes eram realizadas sem planejamento e deforma não periódica.

Em função da integração das atividades de mensuração com o processo de desenvolvimento, uma característica importante desta abordagem é a necessidade de definição prévia de um processo de desenvolvimento, que posteriormente irá suportar as atividades de mensuração. No exemplo apresentado, ainda que não tenha sido definido um processo ideal, o fato de ter sido definido e modelado um processo já pode ser considerado como outro avanço para a organização, que antes não possuía um processo definido (situação comum em muitas organizações que possuem equipes pequenas de desenvolvimento). A definição de um processo de desenvolvimento é um passo importante para a melhoria na qualidade do desenvolvimento.

As atividades de coleta, análise e avaliação do processo de mensuração não foram simuladas neste exemplo, em função de se tratarem de etapas apenas de registro de informações, sendo necessário aplicá-las em um ambiente de produção para proporcionar uma melhor avaliação de sua aplicabilidade.

Assim, o exemplo acima serviu como uma maneira de verificar a aplicabilidade do modelo, tendo sido cumprido este objetivo. Na seção seguinte são apresentadas as conclusões deste trabalho bem como a projeção de trabalhos futuros vislumbrados à partir da pesquisa realizada.

8 CONCLUSÕES

Cada vez mais as empresas que desenvolvem *software* precisam qualificar e melhorar seus processos, bem como os produtos desenvolvidos. Como vimos ao longo deste estudo, uma das maneiras de podermos melhorar a qualidade de processos e produtos, e de permitir uma avaliação mais confiável e segura dos elementos que compõe o desenvolvimento de um *software*, passa, sem dúvida, pela utilização de métricas. Vimos também que é extremamente importante que a coleta e utilização de métricas em um processo de desenvolvimento deve estar amparada por metodologias ou padrões que permitam uma sistematização deste processo.

Assim, este trabalho apresentou um modelo para mensuração em processos de *software*, o qual possibilita a condução, de forma organizada, de planos de mensuração para organizações desenvolvedoras de *software*, através da definição de programas de mensuração. O modelo proposto, denominado APSEE-*Metrics*, foi integrado ao ambiente APSEE de modelagem e execução de processos. Este modelo fornece mecanismos para a definição dos programas de mensuração para os processos de *software* definidos no ambiente APSEE, com o objetivo de facilitar a condução das atividades relativas à mensuração em projetos de desenvolvimento de software.

O modelo APSEE-*Metrics* foi especificado formalmente, com a utilização do formalismo Prosoft-Algébrico. A especificação formal deu origem à implementação de um protótipo, no qual foi desenvolvido um exemplo de uso do modelo. Embora o trabalho aqui apresentado não forneça uma solução completa para o problema investigado, acredita-se que o modelo proposto constitui uma contribuição importante para a área de mensuração de *software*, em virtude da especificação apresentada e da prototipação realizada. Além disto, a integração promovida com o ambiente APSEE proporcionou também um avanço importante do projeto em relação ao ambiente de desenvolvimento de *software* ao qual diversas pesquisas relacionadas às áreas de Engenharia de *Software* e Processos de *Software* vêm sido desenvolvidas pelo grupo.

Este capítulo contém as conclusões obtidas com a realização do trabalho, e está estruturado como segue: a seção 8.1 uma análise do APSEE-*Metrics* frente aos requisitos definidos neste trabalho, a sção 8.2 apresenta as contribuições do trabalho; na seção 8.3 são abordadas as limitações do modelo proposto; e a seção 8.4 identifica alguns trabalhos futuros. Por fim, a seção 8.5 apresenta as considerações finais do capítulo.

8.1 Análise do APSEE-*Metrics* frente aos requisitos

Após concluídas as etapas realizadas nesta pesquisa, entendeu-se como sendo importante realizar uma análise crítica do modelo desenvolvido em relação aos requisitos definidos na seção 2.7. A tabela 8.1 apresenta uma rápida análise para cada requisito definido, indicando o atendimento dos mesmos pelo modelo.

Tabela 8.1: Análise do modelo frente aos requisitos

Atendimento	**Análise do atendimento do requisito**
Categoria: Requisitos relativos ao planejamento e definição das medições	
R1.1 - Suporte ao planejamento das medições	
Sim	O modelo suporta o planejamento através da utilização de objetivos de mensuração, que orientam a definição das métricas apropriadas a cada contexto, abrangendo o plano de mensuração que contempla as necessidades de processo e de informação para mensuração.
R1.2 - Suporte ao planejamento e definição do processo de medição:	
Sim	Através do mecanismo de modelagem de processos do APSEE, o processo de mensuração pode ser definido, indicando as atividades, responsáveis, dependências e artefatos produzidos por cada atividade. A especificação de como deve ser coletada cada métrica também é permitida pelo modelo.
R1.3 - Seleção e classificação dos objetivos da medição:	
Sim	Utilizando o modelo ICM permitiu a classificação das métricas e sua derivação à partir de objetivos de mensuração e necessidades de informação do projeto.
R1.4 - Compartilhamento das medições:	
Sim	Todas as métricas são armazenadas em um repositório que pode ser compartilhado entre diferentes necessidades de informação e entre as ferrramentas disponíeis no APSEE.
R1.5 - Adequação aos diferentes contextos e níveis de maturidade das empresas:	
Sim	O modelo permite que o especialista defina os seus próprios programas de mensuração, indicando o nível de maturidade necessário à cada contexto, uma vez que o mesmo tem a liberdade de definir suas próprias métricas e o processo de coleta, análise e avaliação, de acordo com sua necessidade.
R1.6 - Possibilidade de criação de nova métricas:	
Sim	O modelo permite criar as métricas próprias para cada organização ou projeto e associá-las às categorias do modelo ICM.
R1.7 - Reuso e adaptação dos planos de medição:	
Parcial	Não foram especificadas funcionalidades de reuso, porém pela arquitetura proposta, as informações do modelo de mensuração podem ser reutilizadas em outros projetos com pequenas adaptações. Um mecanismo para reuso de processos existente no APSEE permite que o processo de mensuaração possa ser reutilizado.
R1.8 Catálogos de medições:	
Sim	Todas as métricas ficam disponíveis em um repositório para futuros projetos, bem como as avaliações realizadadas das memas.
Categoria: Requisitos relativos ao processo de medição	
R2.1 - Integração das atividades de mensuração ao processo de produção do *software*:	
Sim	O processo de mensuração é modelado de forma integrada ao processo de desenvolvimento, através dos recursos de uma linguagem de modelagem de processos disponível pelo APSEE.
R2.2 – Suporte a coleta de dados das medições:	
Parcial	O modelo não especifica funcionalidades para coleta automática de métricas, o que deve ser aperfeiçoado.
R2.3 - Histórico das medições:	
Sim	O histórico das métricas é possível, uma vez que cada métrica utilizada é armazenada no banco de dados. A visualização das métricas pode se dar durante o planejamento de novas medições quanto durante o processo, para consultas.
R2.4 – Avaliação e monitoramento do custo/esforço relativo ao processo de medição:	
Parcial	Uma vez que o processo de mensuração é integrado ao processo de desenvolvimento, é possível visualizar a introdução destas atividades, e, à partir da execução de processos no APSEE é passível de cálculo deste custo. Há

	necessidade de especificação e implementação de funcionalidades que permitam a visualização e cálculo deste custo.
R2.5 Extração de dados de outras ferramentas/bases de dados:	
Não	A ferramenta não provê funcionalidades para extração de dados de outras ferramentas, apenas do próprio ambiente. O processo de registro destas informações pode ser feito manualmente para cada métrica.
Categoria: Requisitos relativos a análise, avaliação e divulgação dos resultados	
R3.1 - Divulgação das Medições:	
Parcial	As funcionalidades propostas são bastante simples e não contemplam gráficos e relatórios elaborados para visualizaçÃo da medições, uma vez que o principal foco do modelo é o planejamento e a modelagem do processo de mensuração.
R3.2 - Suporte às análises dos dados coletados:	
Parcial	O modelo prevê a realização de análises durante o processo. Neste sentido foram propostas algumas funcinalidades simples para registro das análises realizadas foram definidos. Processos automatizados para busca de padrões e relacionamento de métricas com outras características do processo de desenvolvimento (tal como variação dos requisistos) não foram propostas, mas poderiam ser incorporadas ao modelo futuramente.
R3.3 - Registro das decisões:	
Parcial	O modelo prevê registro das decisões realizadas com base nas análises durante o processo. Neste sentido foram propostas algumas funcinalidades simples para registro.
R3.4 - Permitir a avaliação do plano de medição:	
Sim	O modelo disponibiliza os recursos para a avaliação das métricas, do processo de mensuração, do plano de medição e demais elementos que necessitarem ser avaliados no processo de mensuração.

Como vimos na tabela acima, boa parte dos requisitos definidos inicalmente foi contemplada pelo modelo, o que demonstra sua validade em relação às recomendações apresentadas na literatura como sendo importantes nesta área. Após esta análise, na próxima seção são apresentdas as principais contribuições deste trabalho.

8.2 Contribuições

As principais contribuições apresentadas por este trabalho são listadas a seguir:

- O texto apresenta, no capítulo 2, um estudo sobre diversos aspectos relacionados à mensuração, bem como um conjunto relevante de requisitos elaborados com base em um estudo detalhado sobre as principais abordagens existentes e consolidadas na literatura especializada da área. Acredita-se que o estudo apresentado, bem como o conjunto de requisitos levantado, fornecem uma base importante para avaliação do trabalho aqui apresentado, bem como para futuras pesquisas relacionadas ao tema desta pesquisa;
- O modelo de mensuração viabiliza o planejamento dos elementos necessários para que se possa realizar mensuração em processos de *software*. Além disto, contempla as principais etapas necessárias para a condução de programas de mensuração, abrangendo planejamento, coleta, análise e avaliação do programa;
- O mecanismo de modelagem do processo de mensuração permite a documentação e facilita a análise e a gerência das atividades necessárias para cada uma das etapas deste processo. Além disto, a integração destas

atividades com o processo de desenvolvimento é um fator de importante relevância, uma vez que permite a interação de diferentes aspectos relacionados à gestão de um processo de desenvolvimento de *software*, o que inclui necessariamente a mensuração;

- Os diferentes componentes envolvidos na definição do modelo foram especificados formalmente, constituindo uma base semântica de alto nível de abstração, o que traz muitos benefícios quando se considera a obtenção de uma especificação independente de implementação, tendo também o potencial de permitir que as idéias do trabalho possam ser implementadas em outros ambientes de desenvolvimento de *software*;
- Um protótipo que implementa o modelo de mensuração foi construído no ambiente Prosoft-Java e integrado ao ambiente APSEE. Essa integração complementa ainda mais o APSEE, e permite que o APSEE-*Metrics* utilize os recursos disponíveis no ambiente. Um exemplo de utilização do protótipo foi apresentado com base em um processo de desenvolvimento real, visando demonstrar as características do modelo proposto;
- Uma das características importantes desta abordagem é a integração das atividades de mensuração à um ambiente de desenvolvimento de *software* (ADS), demonstrado através do ambiente APSEE. Entende-se que tal integração proporcionou algumas vantagens importantes, como a construção do modelo de mensuração à partir dos meta-modelos do APSEE, usufruindo de todo o conjunto de ferramentas já projetado no ambiente; a complementação da efetividade do ambiente, no que diz respeito a coleta de métricas relativas ao processo (sendo esta área um dos trabalhos futuros sinalizados na definição original do APSEE); os dados providos pela utilização deste modelo de mensuração poderão servir de base para outras ferramentas integradas ao ambiente, como as ferramentas de reuso e adaptação de processos de *software*, além de possibilitar base para a avaliação dos processos utilizados; a possibilidade de visualização do processo de desenvolvimento de forma integrada ao processo de mensuração, permitindo, ainda que de forma indireta, a análise do impacto da introdução de atividades de mensuração em relação aos custos e tempo total de cada projeto, entre outras vantagens futuras.

8.3 Limitações

Devido à complexidade do problema tratado, alguns tópicos não foram abordados no modelo proposto, por apresentarem, por si só, um grau de complexidade elevado, ou por não estarem contidos no escopo do trabalho. Esses tópicos constituem, em alguns casos, limitações na abrangência do modelo proposto, e podem ser alvo de melhorias futuras a serem incorporadas ao modelo. Dentre as limitações encontradas, é importante mencionar as seguintes:

- O APSEE-*Metrics* não provê suporte às estimativas de projeto. Desta forma, não foram desenvolvidas funcionalidades bem como a infra-estrutura

necessária para suportar o uso de estimativas, o que pode ser importante futuramente para realização de análises comparativas em relação às métricas coletadas. Vale salientar que essa limitação foi assumida como premissa do trabalho, em função escopo estabelecido para o mesmo;

- O mecanismo de coleta das métricas foi definido de modo bastante simplificado, apenas com caráter demonstrativo, visando apresentar as funcionalidades necessárias para viabilização do modelo. Para seu uso efetivo em produção, acredita-se ser muito importante a evolução do mesmo;
- Outros aspectos do modelo que merecem um estudo mais detalhado são os referentes às atividades de análise e avaliação. Estes itens são de importante relevância em programas de mensuração, e podem ter mecanismos específicos que auxiliem na execução destas atividades, de forma mais automática e orientada.

8.4 Trabalhos futuros

O trabalho aqui apresentado trouxe importantes contribuições, apresentadas anteriormente, sendo que em relação ao meta-modelo APSEE, procurou-se definir uma estrutura de base para novas pesquisas relacionadas no ambiente APSEE. À partir deste trabalho, visualiza-se o desenvolvimento de outros trabalhos que abordem aspectos complementares ao tema abrangido:

- Um dos tópicos que necessita de exploração é a avaliação empírica da influência do modelo proposto na gerência de projetos de *software*. A realização desse trabalho, por si só, não garante um aumento da qualidade, produtividade ou melhoria na gerência de projetos, processos ou para o software resultante. Assim, uma importante atividade a ser realizada é a condução de uma avaliação empírica que determine, através de estudos de casos aplicados em projetos reais de desenvolvimento, os benefícios reais obtidos quando o modelo proposto é utilizado;
- Um aspecto relacionado que não foi considerado neste trabalho é a utilização de estimativas de projeto. Estudos podem ser realizados para permitir a definição e cálculo de estimativas relacionadas às métricas definidas no modelo de mensuração. Um mecanismo para definição, cálculo e comparação entre estimativas e métricas pode ser necessário;
- Em relação ao protótipo, aspectos de usabilidade e facilidades em relação à relatórios e demais funcionalidades necessitam ser aperfeiçoadas. Como pontos de melhoria, alguns dos aspectos que poderiam ser melhor trabalhados seriam a coleta das métricas, bem como a apresentação das mesmas de forma mais interativa. Como o propósito deste trabalho era fundamentar uma infra-estrutura de base para a realização de mensuração (foco no modelo), as questões relacionadas à implementação de funcionalidades e facilidades de uso da ferramenta foram simplificadas (tendo o propósito apenas de demonstração da viabilidade), porém, são

consideradas muito importantes para uma futura utilização da ferramenta em escala de produção.

- Embora o uso de métodos de especificação formal possibilite a realização de provas e verificações matemáticas acerca de propriedades do modelo, tal aspecto não foi considerado como objetivo do trabalho aqui apresentado. A realização de provas, entretanto, é um recurso importante que pode ser aproveitado em trabalhos futuros. Um ponto que poderia ser provado é que as atividades ligadas ao processo de mensuração sempre estarão em estados consistentes, sem que a execução das mesmas viole as dependências entre atividades definidas nos modelos de processo de desenvolvimento;
- Por fim, outro tópico para trabalhos futuros é a viabilização da coleta automática de métricas através do relacionamento de atributos de cada um dos componentes do ambiente APSEE (tais como atividades, pessoas, artefatos, etc.), de forma integrada com as métricas definidas no modelo.

8.5 Considerações finais

Este trabalho foi desenvolvido com o propósito de contribuir com a evolução da área de mensuração de *software*, propondo um modelo de mensuração para ambientes de desenvolvimento de *software* orientado à processos (PSEE).

As idéias contidas neste trabalho e suas motivações foram embasadas em especial pelas abordagens existentes em relação à mensuração, onde foram utilizados como inspiração alguns dos princípios apresentados pelo PSM, apresentado no capítulo 3, bem como as características e princípios relativos ao meta-modelo APSEE, que constituiu importante base para este trabalho.

Por fim, este trabalho possibilitou a combinação de vários aspectos relacionados à importantes áreas da Engenharia de *Software*, como mensuração, processos de *software*, ambientes de engenharia de *software* centrados no processos (PSEE), bem como a utilização de métodos formais para a definição e especificação de um modelo de mensuração integrado a um PSEE.

REFERÊNCIAS

ABIB, J.C.; KIRNER, T.G. A GQM-based tool to support the development of software quality measurement plans. **ACM SIGSOFT Software Engineering Notes**, New York, v.24, n.4, p. 75-80, July 1999.

ABREU, F.B. As Métricas na Gestão de Projectos de Desenvolvimento de Sistemas de Informação. In: JORNADAS PARA A QUALIDADE NO SOFTWARE, 6., 1992. **Anais...** Lisboa: APQ, 1992.

AGUIAR, M. **PSM - O CMM da Mensuração de Software**. 2002. Disponível em: <http://www.metricas.com.br/Downloads/PSM_CMM_Mensuracao_Software.pdf> Acesso em: 15 out. 2005.

ANACLETO, A.; VON WANGENHEIN, C.G. Aplicando Mensuração em Microempresas de Software para suporte da Gerência de Projetos. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 1., 2002. **Anais...** Gramado: SBC, 2002.

ANDRADE, E.L.P. de et al. Aplicando Mensuração em Microempresas de Software para suporte da Gerência de Projetos. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 2., 2003. **Anais...** Fortaleza: SBC, 2003.

ANDRADE, E.L.P. de; et al. Gestão da estimativa de tamanho de projeto de software orientado a objetos**.** In: WORKSHOP DE TESES E DISSERTAÇÕES DE QUALIDADE DE SOFTWARE, 1., 2003. **Anais...** Fortaleza: SBC, 2003.

ANDRADE, J.M.S. Medição, Avaliação e Melhoria de Processos em Ambientes de Desenvolvimento Orientados a Organização**.** In: WORKSHOP DE TESES E DISSERTAÇÕES DE QUALIDADE DE SOFTWARE, 1., 2003. **Anais...** Fortaleza: SBC, 2003.

BASILI, V.R.; ROMBACH, H.D. Tailoring the software process to project goals and environments. In: INTERNATIONAL CONFERENCE ON SOFTWARE

ENGINEERING, 9., 1987, Monterey, California, United States. **Proceedings**… Los Alamitos, CA: IEEE Computer Society Press, 1987. p.345-357.

BASILI, V. et al. SEL's Software Process Improvement Program. **IEEE Software**, Los Alamitos, CA, v.12, n.6, p. 83-87, Nov., 1995.

BASILI, V. R. et al. Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 24., 2002, Orlando, Florida. **Proceedings…** New York: ACM Press, 2002. p. 69-79.

BORGES, E.P. **Um modelo de medição para processos de desenvolvimento de software**. 2003. 154 f. Dissertação (Mestrado) - UFMG, Belo Horizonte.

BRIAND, L.; MORASCA, S. ; BASILI, V.R. **A goal-driven definition process for product metrics based on properties**. [S.l.]: Univ. of Maryland, Department of Computer Science, 1994. (Tech. Report CS-TR-3346, UMIACS-TR-94-106).

BRIAND, L. C.; MORASCA, S.; BASILI, V. R. An Operational Process for Goal-Driven Definition of Measures. **IEEE Transactions on Software Engineering**, Los Alamitos, CA, v.28, n.12, p.1106-1125, Dec. 2002.

CARD, D.N. Practical software measurement. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 25., 2003. Portland, Oregon. **Proceedings…**. Washington, DC: IEEE Computer Society, 2003. p. 738-739.

CARVALHO, G.L. de. Métricas de Modelação de *Software***.** In: SEMINÁRIO DE PFSI, 2004. **Anais...** Ramada, Portugal: Instituto Superior Técnico, 2004.

CANOSSA, S. **Gerenciando o Sistema de Medição.** Disponível em: <http://internal.dstm.com.ar/sites/mmnew/bib/notas/Medição.pdf>. Acesso em: 22 maio 2005.

DAWSON, R.; O'NEILL, B. Simple Metrics for Improving Software Process Performance and Capability: A Case Study**. Software Quality Control,** Netherlands, v.11, n.3, p.243-258, July 2003.

DIAS, R. **Métricas para avaliação de Sistemas de Informação**. 2002. Disponível em: < http://www.inf.ufsc.br/resi/edicao01/artigo02.pdf>. Acesso em: 22 set. 2005.

FENTON, N. Software Measurement: A Necessary Scientific Basis. **IEEE Transactions on Software Engineering**, Los Alamitos, CA**,** v.20, n.3, p.199-206, March 1994.

FRANCA, L.P.A.; VON STAA, A.; LUCENA, C.J.P. de. **Medição de Software para Pequenas Empresas:** uma Solução Baseada na Web. Rio de Janeiro: PUC-RJ, 1998. 24 p.

FRANÇA, L.P.A. et al. Um Modelo de Classes para Ambiente de Geração de Programas de Medição de Software Baseado na Web. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 13., 1999, Florianópolis. **Anais...** Florianópolis: UFSC, 1999. p.225-237.

FUENTES, A.A.J. **Necesidad de Sistemas Formales de Métricas para Proyectos de Software OO**. Oviedo: Universidade de Oviedo, 1997. 12 p.

FUGGETTA, A. et al. Applying GQM in an industrial software factory**. ACM Transactions on Software Engineering and Methodology**, New York, v.7, n.4, p.411-448, Oct. 1998.

GARCIA, F.. **Gestión Integrada del Modelado y de la Medición del Processo Software**. Ciudad Real: Escuela Superior de Informática, 2003. 8 p.

GARCIA JÚNIOR, P.R.; DIAS, R.P. Utilizando a Tecnologia da Informação como base para a melhoria de processos institucionais através do SGL - Sistema de Gestão La Salle. In: WORKSHOP DE ENGENHARIA E TECNOLOGIA, 1., 2005, Lajeado. **Anais...** Lajeado: UNIVATES, 2005.

GOETHERT, W.; FISHER, M. **Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques**. Pittsburgh: Carnegie Mellon University, 2003.

GOMES, Augusto. Avaliação de Processo de Software Baseada em Medições. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, 2001.

HERBSLEB, J.D.; GRINTER, R.E. Conceptual simplicity meets organizational complexity: case study of a corporate metrics program**.** In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 20., Kyoto, Japan. **Proceedings…** Washington, DC: IEEE Computer Society, 1998. p. 271-280.

JACKSON, M. **System Development**. New York: Prenctice-Hall International, 1993.

KRISHNAN, M.S.; KELLNER, M.I. Measuring Process Consistency: Implications for Reducing Software Defects. **IEEE Transactions on Software Engineering**, Los Alamitos, CA, v.25, n.6, p.800-815, Nov. 1999.

KÖRBES, F. **Implementação de um mecanismo de herança no PROSOFT.** 1996. Projeto de Diplomação (Curso de Ciência da Computação) - UFRGS, Porto Alegre.

LANE, J.A; ZUBROW, D. Integrating Measurement with Improvement: An Action-Oriented Approach Experience Report. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 19., 1997, Boston, MA, United States. **Proceedings…** Los Alamitos, CA: IEEE Computer Society Press, 1997. p.380-389.

MENESES, J. **Inspector:** Um processo de Avaliação de Progresso para Projetos de Software. 2001. 163 f. Dissertação (Mestrado) - UFPE, Recife.

MENESES, J. Inspector: Um processo de Avaliação de Progresso para Projetos de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, 2001.

MENS, T.; DEMEYER, S. Future trends in software evolution metrics. In: INTERNATIONAL WORKSHOP ON PRINCIPLES OF SOFTWARE EVOLUTION IWPSE, 4., 2001, Vienna, Austria, **Proceedings…** New York, NY: ACM Press, 2001. p.83-86.

MUNSON, J.C. Software Engineering Measurement. **ACM SIGSOFT *Software* Engineering Notes,** New York, v. 29, n. 4, p.30, May 2004.

DÁVILA NICATOR, L.; MEJÍA ALVAREZ, P. **Evaluacíon de la Calidade de Software en Sistemas de Informacíon en Internet**. Zacatenco - México: CINVESTAV-IPN, 2003. 11p.

NUNES, D.J. Estratégia Data-driven no Desenvolvimento de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 6., 1992, Gramado. **Anais...** Porto Alegre: UFRGS, 1992. p.81-95.

NUNES, D.J. **PROSOFT:** Um ambiente de Desenvolvimento de Software baseado no metodo algébrico**.** Porto Alegre: UFRGS, 1995. 20 p.

OGASAWARA, H.; YAMADA, A.; KOJO, M. Experiences of software quality management using metrics through the life-cycle**.** In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 18., Berlin, Germany. **Proceedings…** Washington, DC: IEEE Computer Society, 1996. p. 179-188.

OLIVEIRA JÚNIOR, E.A. de; GIMENES, I.M. de S.. Um modelo para Avaliação da Qualidade da Gerência de Projetos de Software. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 1., 2002, Gramado. **Anais...** Gramado: UFRGS, 2002.

OLSINA, L.A. et al. Un Marco Conceptual para la Definición y Explotacíon de Métricas de Calidad. In: JORNADAS DE INGENIERÍA DEL SOFTWARE Y BASES DE DATOS, 7., 2001, Málaga, Espanha. **Anais...** Málaga: University of Málaga, 2001.

OMG (Object Management Group). **Unified Modeling Language:** UML 2 Superstructure. Needham, USA, 2004.

PARK, R.E.; GOETHERT, W.B.; FLORAC, W.A. **Goal-Driven Software Measurement:** a Guidebook. Carnegie Mellon University, 1996. Disponível em: <http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.002.html>. Acesso em: jul. 2005.

PERKINS, T.; PETERSON R., SMITH, L. **Back to the Basics:** Measurement and Metrics**.** USA: Departament of Defense and USA Army, 2003. Disponível em: <www.stsc.af.mil/crosstalk> Acesso em: out. 2005.

PERRELLI, H. Gerência de Projetos: O modelo PMBOK**.** In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 3., 2004, Brasilia. **Anais...** Brasilia: UCB, 2004.

PRACTICAL SOFTWARE MEASUREMENT SUPPORT CENTER. **PSM Insight Tool.** Disponível em: <http://www.psmsc.com/insight.htm>. Acesso em: set. 2005.

PRACTICAL SOFTWARE MEASUREMENT SUPPORT CENTER. **PSM Insight Tool**. Disponível em: <http://www.psmsc.com/insight.htm> Acesso em: jan. 2006.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill,2006. 720 p.

QUEIROZ, E. **Ambiente de Mensuração de Software Orientado a Objetos**. 1999. 186 f. Dissertação (Mestrado) - UnB, Brasilia.

RAMOS, C.S. Avaliação da Qualidade de Sistemas Legados. In: WORKSHOP DE TESES E DISSERTAÇÕES DE QUALIDADE DE SOFTWARE, 1., 2003, Fortaleza. **Anais...** Fortaleza: UniFor, 2003.

REIS, C.A.de L. **Um gerenciador de processos de software para o ambiente Prosoft**. 1998. 197 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

REIS, C.A.de L.; REIS, R.Q.; NUNES, D.J. APSEE: uma abordagem integrada para automação de processos de software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, 2001.

REIS, C.A.de L.; REIS, R.Q.; NUNES, D.J.; ABREU, M.M. APSEE: um modelo formal e flexível para execução de processos de software**.** In: WORKSHOP IBEROAMERICANO DE INGENIERÍA DE REQUISITOS Y DESARROLLO DE AMBIENTES SOFTWARE, 5., 2002, La Habana. **Memorias...** La Habana: Universidad de la Habana, 2002. p.201-213

REIS, C.A.de L.; NUNES, D.J.; REIS, R.Q.; SCHLEBBE, H. A Abordagem APSEE para Modelagem e Gerência de Recursos em Ambientes de Processos de Software**.** In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, 2001. p.68-83

REIS, C.A.de L. **Uma Abordagem Flexível para Execução de Processos de Software Evolutivos.** 2003. 215 f. Tese (Doutorado em Ciência da Computação) – Instituito de Informática, UFRGS, Porto Alegre.

REIS, R. Q. **Uma Proposta de Suporte ao Desenvolvimento Cooperativo de Software no Ambiente PROSOFT.** 1998. 177 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Infomática, UFRGS, Porto Alegre.

REIS, R.Q. **APSEE-REUSE - Um Meta-Modelo para Apoiar a Reutilização de processos de Software.** 2002. 215 f. Tese (Doutorado em Ciência da Computação) - Instituto de Infomática, UFRGS, Porto Alegre.

RODRIGUES, J.G.; OLSINA, L. Hacia la Medición de Calidad en Uso Web. In: JORNADAS DE INGENIERÍA DEL SOFTWARE Y BASES DE DATOS, 6., 2001, Ciudad Real. Disponível em: <http://www.dlsi.ua.es/~jaime/webe/articulos/s222.pdf>. Acesso em: mar. 2006.

SALVIANO, C.F. **Contribuições da Melhoria de Processo e Gerência de Projetos:** Transformando Boas Idéias em Resultados. Belo Horizonte: UFMG, 2002. 4 p.

SCHLEBBE, H.; SCHIMPF, S. **Reenginering of PROSOFT in Java**. Stuttgart: Stuttgart University, 1997. 15 p.

SOUSA, A.L.R. de. **APSEE-Monitor:** um mecanismo de apoio a Visualização de modelos de processos de software**.** 2004. 112 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Infomática, UFRGS, Porto Alegre.

STATZ, J. Practical software measurement. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 21., Los Angeles, California, United States. **Proceedings…** Los Alamitos, CA: IEEE Computer Society Press, 1999. p.667-668.

SOMMERVILLE, I. **Engenharia de *Software***. São Paulo: Addison Wesley, 2003. 592 p.

SUN MICROSYSTEMS. **Java Runtime Environment**. Disponível em: <http://www.java.sun.com>. Acesso em: set. 2005.

TANAKA, T et. al Software quality analysis and measurement service activity in the company. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 20., Kyoto, Japan. **Proceedings…** Washington, DC: IEEE Computer Society, 1998. p. 426-429.

USA. Department of Defense and USA Army. **Practical Software and Systems Measurement**: A Foundation for Objective Project Management. Washington, D.C.: Department of Defense and US Army, 2003. Disponível em: <http://www.psmsc.com>. Acesso em: 05 ago. 2005.

VAVASSORI, F. B. Ferramenta CASE para Gerenciamento de projetos e métricas de software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15., 2001, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ, 2001.

VAVASSORI, F. B. Integrando Métrica de Software ao Gerenciamento de Projetos. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 15., 2001, Itajaí. **Anais...** Itajaí: UNIVALI, 2001.

VAVASSORI, F. B. **Metodologia para o Gerenciamento Distribuído de Projetos e Métrica de Software**. 2002. 211 f. Tese (Doutorado) - UFSC, Florianópolis.

WEBER, K.C. et al. Modelo de Referência para Melhoria do Proceso de Software: uma abordagem brasileira. In: CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA, 30.,2004, Arequipa. **Articulos...** La Paz: Universidad de San Andrés, 2004.

YEO, K.T. Critical failure factors in information system projects. **International Journal of Project Management**, St Louis, v.20, n.3, p. 241–246, Apr. 2002.

ZANONI, R.; AUDY, J.L.N. Um Modelo de Gerência de Projetos Voltado para um Ambiente de Desenvolvimento de Software Fisicamente Distribuído. In: CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA, 28., 2002, Montevideo. **Articulos...** Montevideo: Universidad de la Republica, 2002.

ZANONI, R. **Modelo de Gerência de Projeto Baseado no PMI para Ambiente de Desenvolvimento de Software Fisicamente Distribuído**. 2002. 131 f. Tese (Doutorado) – PUC-RS, Porto Alegre.

APÊNDICE A ESPECIFICAÇÃO FORMAL DAS OPERAÇÕES DO MODELO APSEE-METRICS

Neste apêndice é apresentado o detalhamento da especificação formal do modelo APSEE-Metrics, apresentado na seção 5.2 deste trabalho. Para cada classe do modelo serão descritas suas as interfaces, variáveis formais e operações, utilizando a notação do Prosoft Algébrico.

ATO CommonIssue

INTERFACE

newCommonIssue		→ CommonIssue
addCommonIssue (_,_,_,_)	CommonIssue, String, String, Text	→ CommonIssue
updCommonIssue (_,_,_,_)	CommonIssue, String, String, Text	→ CommonIssue
removeCommonIssue (_,_)	CommonIssue, String	→ CommonIssue
existsCommonIssue (_,_)	CommonIssue, String	→ Boolean
getCommonIssue(_,_)	CommonIssue, String	→ CommonIssue

VARIÁVEIS FORMAIS

cIssue	COMMONISSUE
id, name	STRING
description	TEXTO
new_name	STRING
new_description	TEXTO
s_id	STRING

OPERAÇÕES

newCommonIssue = empty-mapping

addCommonIssue(cIssue, id, name, description)
= modify (id, (Name name, Description description), cIssue)

updCommonIssue(modify(id, (Name name, Description description), cIssue), s_id, new_name, new_description)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description), cIssue)
else modify(id, (Name name, Description description), updCommonIssue(cIssue, s_id, new_name new_description))

updCommonIssue(empty-mapping,_,_,_) = empty-mapping

removeCommonIssue(cIssue, s_id) = restrict_with (cIssue, add(empty-set, s_id))

removeCommonIssue(empty-mapping, _) = empty-mapping

existsCommonIssue(modify(id, (Name name, Description description), cIssue), s_id)
= **if** id = s_id
then true
else existsCommonIssue(cIssue, s_id)

existsCommonIssue (empty-mapping,_) = false

getCommonIssue(modify(id, (Name name, Description description), cIssue), s_id)
= **if** id = s_id
then modify(id, (Name name, Description description), cIssue)
else getCommonIssue(cIssue), s_id)

getCommonIssue (empty-mapping,_) = empty-mapping

ATO MeasurementCategory

INTERFACE

newMeasurementCategory		→ MeasurementCategory
addMeasurementCategory (_,_,_,_,_)	MeasurementCategory, String, String, Text, CommonIssue	→ MeasurementCategory
updMeasurementCategory (_,_,_,_,_)	MeasurementCategory, String, String, Text, CommonIssue	→ MeasurementCategory
removeMeasurementCategory (_,_)	MeasurementCategory, String	→ MeasurementCategory
existsMeasurementCategory (_,_)	MeasurementCategory, String	→ Boolean
getMeasurementCategory (_,_)	MeasurementCategory, String	→ MeasurementCategory

VARIÁVEIS FORMAIS

measCat	MEASUREMENTCATEGORY
id, name	STRING
description	TEXTO
issue	COMMONISSUE
s_id	STRING
new_id, new_name	STRING
new_description	TEXTO
new_issue	COMMONISSUE

OPERAÇÕES

newMeasurementCategory = empty-mapping

addMeasurementCategory (measCat, id, name, description, issue)
= modify (id, (Name name, Description description, Issue issue), measCat)

updMeasurementCategory (modify(id, (Name name, Description description, Issue issue), measCat), s_id, new_name, new_description, new_issue)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, Issue new_issue), measCat)
else modify(id, (Name name, Description description, Issue issue), updMeasurementCategory (measCat, s_id, new_name new_description, new_issue))

updMeasurementCategory (empty-mapping,_,_,_) = empty-mapping

removeMeasurementCategory (measCat, s_id) = restrict_with (measCat, add(empty-set, s_id))

removeMeasurementCategory (empty-mapping, _) = empty-mapping

existsMeasurementCategory (modify(id, (Name name, Description description, Issue issue), measCat), s_id)
= **if** id = s_id
then true
else existsMeasurementCategory (measCat, s_id)

existsMeasurementCategory (empty-mapping,_) = false

getMeasurementCategory (modify(id, (Name name, Description description, Issue issue), measCat), s_id)
= **if** id = s_id
then modify(id, (Name name, Description description, Issue issue), measCat)
else getMeasurementCategory (measCat, s_id)

getMeasurementCategory (empty-mapping,_) = empty-mapping

ATO GenericMeasure

INTERFACE

newGenericMeasure		→ GenericMeasure
addGenericMeasure (_,_,_,_,_)	GenericMeasure, String, String, Text, MeasurementCategory	→ GenericMeasure
updGenericMeasure (_,_,_,_,_)	GenericMeasure, String, String, Text, MeasurementCategory	→ GenericMeasure
removeGenericMeasure (_,_)	GenericMeasure, String	→ GenericMeasure
existsGenericMeasure (_,_)	GenericMeasure, String	→ Boolean
getGenericMeasure (_,_)	GenericMeasure, String	→ GenericMeasure

VARIÁVEIS FORMAIS

genMeas	GENERICMEASURE
id, name	STRING
description	TEXTO
category	MEASUREMENTCATEGORY
s_id	STRING
new_id, new_name	STRING
new_description	TEXTO
new_category	MEASUREMENTCATEGORY

OPERAÇÕES

newGenericMeasure = empty-mapping

addGenericMeasure (genMeas, id, name, description, category)
= modify (id, (Name name, Description description, Category category), genMeas)

updGenericMeasure (modify(id, (Name name, Description description, Category category), genMeas), s_id, new_name, new_description, new_category)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, Category new_category), genMeas)
else modify(id, (Name name, Description description, Category category), updGenericMeasure (genMeas, s_id, new_name new_description, new_category))

updGenericMeasure (empty-mapping,_,_,_) = empty-mapping

removeGenericMeasure (genMeas, s_id) = restrict_with (genMeas, add(empty-set, s_id))

removeGenericMeasure (empty-mapping, _) = empty-mapping

existsGenericMeasure (modify(id, (Name name, Description description, Category category), genMeas), s_id)
= **if** id = s_id
then true
else existsGenericMeasure (genMeas, s_id)

existsGenericMeasure (empty-mapping,_) = false

getGenericMeasure (modify(id, (Name name, Description description, Category category), genMeas), s_id)
= **if** id = s_id
then modify(id, (Name name, Description description, Category category), genMeas)
else getGenericMeasure (genMeas, s_id)

getGenericMeasure (empty-mapping,_) = empty-mapping

ATO InformationModel

INTERFACE

newInformationModel		→ InformationModel
addInformationModel (_,_,_,_)	InformationModel, String, String, Text	→ InformationModel
updInformationModel (_,_,_,_)	InformationModel, String, String, Text	→ InformationModel
removeInformationModel (_,_)	InformationModel, String	→ InformationModel
existsInformationModel (_,_)	InformationModel, String	→ Boolean
getInformationModel (_,_)	InformationModel, String	→ InformationModel
addGoal (_,_,_)	InformationModel, String, MeasurementGoal	→ InformationModel
addAnalysis (_,_,_)	InformationModel, String, Analysis	→ InformationModel
addEvaluation (_,_,_)	InformationModel, String, Evaluation	→ InformationModel
addProjectIssue (_,_,_)	InformationModel, String, ProjectIssue	→ InformationModel
addMetricDef (_,_,_)	InformationModel, String, MetricDef	→ InformationModel
addMetricCollected (_,_,_)	InformationModel, String, MetricCollected	→ InformationModel
removeGoal (_,_,_)	InformationModel, String, MeasurementGoal	→ InformationModel
removeAnalysis (_,_,_)	InformationModel, String, Analysis	→ InformationModel
removeEvaluation (_,_,_)	InformationModel, String, Evaluation	→ InformationModel
removeProjectIssue (_,_,_)	InformationModel, String, ProjectIssue	→ InformationModel
removeMetricDef (_,_,_)	InformationModel, String, MetricDef	→ InformationModel
removeMetricCollected (_,_,_)	InformationModel, String, MetricCollected	→ InformationModel

VARIÁVEIS FORMAIS

model	INFORMATIONMODEL
id, name	STRING
description	TEXTO
goal	MEASUREMENTGOAL
analysis	ANALYSIS
evaluation	EVALUATION
projectissue	PROJECTISSUE
metricdef	METRICDEF
metriccollected	METRICCOLLECTED
goals	SETOFMEASUREMENTGOAL
analyses	SETOFANALYSIS
evaluations	SETOFEVALUATION
projectissues	SETOFPROJECTISSUE
metricsdef	SETOFMETRICDEF
metricscollected	SETOFMETRICCOLLECTED
s_id, new_name	STRING
new_description	TEXTO

OPERAÇÕES

newModel = empty-mapping

addModel (model, id, name, description)
= modify (id, (Name name, Description description, Goals add(empty-set), Analyses add(empty-set), Evaluations add(empty-set), ProjectIssues add(empty-set), Metricdef add(empty-set), MetricsCollected add(empty-set)), model)

updModel (modify(id, (Name name, Description description, _, _, _, _, _, _), model), s_id, new_name, new_description)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, _, _, _, _, _, _), model)
else modify(id, (Name name, Description description, _, _, _, _, _, _), updModel (model, s_id, new_name new_description))

updModel(empty-mapping,_,_,_) = empty-mapping

removeModel(model, s_id) = restrict_with (model, add(empty-set, s_id))

removeModel (empty-mapping, _) = empty-mapping

existsModel(modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model), s_id)
= **if** id = s_id
then true
else existsModel (model, s_id)

existsModel (empty-mapping,_) = false

getModel(modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model), s_id)
= **if** id = s_id
then modify(id, (Name name, Description description, Goals goals, Analyses analyses, Evaluations evaluations, ProjectIssues projectissues, Metricdef metricsdef, MetricsCollected metricscollected), model)
else getModel (model, s_id)

getModel (empty-mapping,_) = empty-mapping

addGoal (modify (id, (_,_, Goals goals, _, _, _, _, _), model), s_id, goal)
= **if** id = s_id
then modify (id, (_,_,Goals add(goals, goal), _, _, _, _, _), model)
else modify (id, (_,_, Goals goals, _, _, _, _, _), addGoals (model, s_id, goal))

addGoal (empty-mapping, _, _) = empty-mapping

removeGoal (modify(id, (_,_, Goals. goal, _, _, _, _, _), model), s_id, goal)
= if id = s_id
then modify(id, (_,_, Goals. delete(goals, goal), _,_,_,_,_), model)
else modify(id, (_,_, Goals. goals, _,_,_,_,_), removeGoal(model, s_id, goal))

removeGoal (empty-mapping,_,_) = empty-mapping

addAnalysis (modify (id, (_,_,_, Analyses analysis, _, _, _, _), model), s_id, analysis)
= **if** id = s_id
then modify (id, (_,_,_, Analyses add(analyses, analysis), _, _, _, _), model)
else modify (id, (_,_,_, Analyses analyses, _, _, _, _), addAnalysis (model, s_id, analysis))

addAnalysis (empty-mapping, _, _) = empty-mapping

removeAnalysis (modify(id, (_,_,_, Analyses. analysis, _, _, _, _), model), s_id, analysis)
= if id = s_id
then modify(id, (_,_,_, Analyses. delete(analyses, analysis), _,_,_,_), model)
else modify(id, (_,_,_, Analyses. analyses, _,_,_,_), removeAnalysis (model, s_id, analysis))

removeAnalysis (empty-mapping,_,_) = empty-mapping

addEvaluation (modify (id, (_,_,_,_, Evaluations evaluation, _, _, _), model), s_id, evaluation)
= **if** id = s_id
then modify (id, (_,_,_,_, Evaluations add(evaluations, evaluation), _, _, _), model)
else modify (id, (_,_,_,_, Evaluations evaluations, _, _, _), addEvaluation (model, s_id, evaluation))

addEvaluation (empty-mapping, _, _) = empty-mapping

removeEvaluation (modify(id, (_,_,_,_, Evaluations. evaluations, _, _, _), model), s_id, evaluation)
= if id = s_id
then modify(id, (_,_,_,_, Evaluations. delete(evaluations, evaluation), _,_,_), model)
else modify(id, (_,_,_,_, Evaluations. evaluations, _,_,_), removeEvaluation (model, s_id, evaluation))

removeEvaluation (empty-mapping,_,_) = empty-mapping

addProjectIssue (modify (id, (_,_,_,_,_,_, Issues issues, _, _), model), s_id, issue)
= **if** id = s_id
then modify (id, (_,_,_,_,_,_, Issues add(issues, issue), _, _), model)
else modify (id, (_,_,_,_,_,_, Issues issues, _, _), addProjectIssue (model, s_id, issue))

addProjectIssue (empty-mapping, _, _) = empty-mapping

removeProjectIssue (modify(id, (_,_,_,_,_,_,Issues. issues, _, _), model), s_id, issue)
= if id = s_id

then modify(id, (_,_,_,_,_, Issues. delete(issues, issue), _,_), model)
else modify(id, (_,_,_,_,_, Issues. issues, _,_), removeProjectIssue (model, s_id, issue))

removeProjectIssue (empty-mapping,_,_) = empty-mapping

addMetricDef (modify (id, (_,_,_,_,_,_,MetricsDef metricsdef, _), model), s_id, metricdef)
= **if** id = s_id
  then modify (id, (_,_,_,_,_,_,MetricsDef add(metricsdef, metricdef), _), model)
  else modify (id, (_,_,_,_,_,_,MetricsDef metricsdef,_), addMetricDef (model, s_id, metricdef))

addMetricDef (empty-mapping, _, _) = empty-mapping

removeMetricDef (modify(id, (_,_,_,_,_,MetricsDef. metricsdef, _), model), s_id, metricdef)
= if id = s_id
then modify(id, (_,_,_,_,_,_,MetricsDef. delete(metricsdef, metricdef), _), model)
else modify(id, (_,_,_,_,_,_,MetricsDef. metricsdef, _), removeMetricDef (model, s_id, metricdef))

removeMetricDef (empty-mapping,_,_) = empty-mapping

addMetricCollected (modify (id, (_,_,_,_,_,_,MetricsCollected metricscollected, _), model), s_id, metriccollected)
= **if** id = s_id
  then modify (id, (_,_,_,_,_,_,MetricsCollected add(metricscollected, metriccollected), _), model)
  else modify (id, (_,_,_,_,_,_,MetricsCollected metricscollected,_), addMetricCollected (model, s_id, metriccollected))

addMetricCollected (empty-mapping, _, _) = empty-mapping

removeMetricCollected (modify(id, (_,_,_,_,_,MetricsCollected. metricscollected, _), model), s_id, metriccollected)
= if id = s_id
then modify(id, (_,_,_,_,_,_,MetricsCollected. delete(metricscollected, metriccollected), _), model)
else modify(id, (_,_,_,_,_,_,MetricsCollected. metricscollected, _), removeMetricCollected (model, s_id, metriccollected))

removeMetricCollected (empty-mapping,_,_) = empty-mapping

ATO MeasurementGoal

INTERFACE

newMeasurementGoal		→ MeasurementGoal
addMeasurementGoal (_,_,_,_,_)	MeasurementGoal, String, String, Text, Integer	→ MeasurementGoal
updMeasurementGoal (_,_,_,_,_)	MeasurementGoal, String, String, Text, Integer	→ MeasurementGoal
removeMeasurementGoal (_,_)	MeasurementGoal, String	→ MeasurementGoal
existsMeasurementGoal (_,_)	MeasurementGoal, String	→ Boolean
getMeasurementGoal (_,_)	MeasurementGoal, String	→ MeasurementGoal

VARIÁVEIS FORMAIS

measGoals	MEASUREMENTGOAL
id, name	STRING
description	TEXTO
priority	INTEGER
s_id	STRING
new_id, new_name	STRING
new_description	TEXTO
new_priority	INTEGER

OPERAÇÕES

newMeasurementGoal = empty-mapping

addMeasurementGoal (measGoals, id, name, description, priority)
 = modify (id, (Name name, Description description, Priority priority), measGoals)

updMeasurementGoal (modify(id, (Name name, Description description, Priority priority), measGoals), s_id, new_name, new_description, new_priority)
= **if** id = s_id
  then modify(id, (Name new_name, Description new_description, Priority new_priority), measGoals)
  else modify(id, (Name name, Description description, Priority priority), updMeasurementGoal (measGoals, s_id, new_name new_description, new_priority))

updMeasurementGoal (empty-mapping,_,_,_) = empty-mapping

removeMeasurementGoal (measGoals, s_id) = restrict_with (measGoals, add(empty-set, s_id))

removeMeasurementGoal (empty-mapping, _) = empty-mapping

existsMeasurementGoal (modify(id, (Name name, Description description, Priority priority), measGoals), s_id)
= **if** id = s_id
  then true
  else existsMeasurementGoal (measGoals, s_id)

existsMeasurementGoal (empty-mapping,_) = false
getMeasurementGoal (modify(id, (Name name, Description description, Priority priority), measGoals), s_id)

= **if** id = s_id
  then modify(id, (Name name, Description description, Priority priority), measGoals)
  else getMeasurementGoal (measGoals, s_id)

getMeasurementGoal (empty-mapping,_) = empty-mapping

ATO ProjectIssue

INTERFACE

newProjectIssue		→ ProjectIssue
addProjectIssue (_,_,_,_,_,_)	ProjectIssue, String, String, Text, Double, Double	→ ProjectIssue
updProjectIssue (_,_,_,_,_,_)	ProjectIssue, String, String, Text, Double, Double	→ ProjectIssue
removeProjectIssue (_,_)	ProjectIssue, String	→ ProjectIssue
existsProjectIssue (_,_)	ProjectIssue, String	→ Boolean
getProjectIssue (_,_)	ProjectIssue, String	→ ProjectIssue
addCommonIssue (_,_,_)	ProjectIssue, String, CommonIssue	→ ProjectIssue
removeCommonIssue (_,_,_)	ProjectIssue, String, CommonIssue	→ ProjectIssue

VARIÁVEIS FORMAIS

projissue	PROJECTISSUE
id, name	STRING
description	TEXTO
probability, impact	DOUBLE
issue	COMMONISSUE
issues	SETOFCOMMONISSUE
s_id, new_name	STRING
new_description	TEXTO
new_probability, new_impact	DOUBLE

OPERAÇÕES

newProjectIssue = empty-mapping

addProjectIssue (projissue, id, name, description, probability, impact)
 = modify (id, (Name name, Description description, Probability probability, Impact impact, CommonIssues add(empty-set)), projissue)

updProjectIssue (modify(id, (Name name, Description description, Probability probability, Impact impact,_), projissue), s_id, new_name, new_description, new_probability, new_impact)
= **if** id = s_id
  then modify(id, (Name new_name, Description new_description, Probability new_probability, Impact new_impact, _), projissue)
  else modify(id, (Name name, Description description, Probability probability, Impact impact, _), updProjectIssue (projissue, s_id, new_name new_description, new_probability, new_impact))

updProjectIssue(empty-mapping,_,_,_,_,_) = empty-mapping

removeProjectIssue(projissue, s_id) = restrict_with (projissue, add(empty-set, s_id))

removeProjectIssue (empty-mapping, _) = empty-mapping

existsProjectIssue(modify(id, (Name name, Description description, Probability probability, Impact impact, CommonIssues issues, projissue), s_id)
= **if** id = s_id
  then true
  else existsProjectIssue (projissue, s_id)

existsProjectIssue (empty-mapping,_) = false

getProjectIssue(modify(id, (Name name, Description description, Probability probability, Impact impact, CommonIssues issues, projissue), s_id)
= **if** id = s_id
  then modify(id, (Name name, Description description, Probability probability, Impact impact, CommonIssues issues, projissue)
  else getProjectIssue (projissue, s_id)

getProjectIssue (empty-mapping,_) = empty-mapping

addCommonIssue (modify (id, (_,_, _, _, Issues issues), projissue), s_id, issue)
= **if** id = s_id
  then modify (id, (_,_,_,_Issues add(issues, issue)), projissue)
  else modify (id, (_,_,_,_ Issues issues), addCommonIssues (projissue, s_id, issue))

addCommonIssue (empty-mapping, _, _) = empty-mapping

removeCommonIssue (modify(id, (_,_,_,_ Issues. issue), projissue), s_id, issue)
= if id = s_id
then modify(id, (_,_,_,_ Issues. delete(issues, issue)), projissue)
else modify(id, (_,_,_,_ Issues. issues), removeCommonIssue (projissue, s_id, issue))

removeCommonIssue (empty-mapping,_,_) = empty-mapping

ATO MetricsDef

INTERFACE

newMetricsDef		→ MetricsDef
addMetricsDef (_,_,_,_,_,_,_,_,_)	MetricsDef, String, String, Text, String, String, Double, Double, String	→ MetricsDef
updMetricsDef (_,_,_,_,_,_,_,_,_)	MetricsDef, String, String, Text, String, String, Double, Double, String	→ MetricsDef
removeMetricsDef (_,_)	MetricsDef, String	→ MetricsDef
existsMetricsDef (_,_)	MetricsDef, String	→ Boolean
getMetricsDef (_,_)	MetricsDef, String	→ MetricsDef
addUnit (_,_,_)	MetricsDef, String, String	→ MetricsDef
removeUnit (_,_,_)	MetricsDef, String, String	→ MetricsDef

VARIÁVEIS FORMAIS

metricsdef	PROJECTISSUE
id, name, type, kind, how	STRING
description	TEXTO
range_from, range_to	DOUBLE
unit	STRING
units	SETOFSTRING
s_id, new_name, new_type,	
new_kind, new_how	STRING
new_description	TEXTO
new_range_from,	
new_range_to	DOUBLE

OPERAÇÕES

newMetricsDef = empty-mapping

addMetricsDef (metricsdef, id, name, description, type, range_from, range_to, how)
= modify (id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo range_to), How how, Units add(empty-set)), metricsdef)

updMetricsDef (modify(id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo) range_to, How how,_), metricsdef), s_id, new_name, new_description, new_type, new_kind, new_range_from, new_range_to, new_how)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, Type new_type, Kind new_kind, (RangeFrom new_range_from, RangeTo new_range_to), How new_how,_), metricsdef)
else modify(id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo range_to), How how, _), updMetricsDef (metricsdef, s_id, new_name, new_description, new_type, new_kind, new_range_from, new_range_to, new_how))

updMetricsDef(empty-mapping,_,_,_,_,_,_,_,_) = empty-mapping

removeMetricsDef(metricsdef, s_id) = restrict_with (metricsdef, add(empty-set, s_id))

removeMetricsDef (empty-mapping, _) = empty-mapping

existsMetricsDef(modify(id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo range_to), How how, Units units, metricsdef), s_id)
= **if** id = s_id
then true
else existsMetricsDef (metricsdef, s_id)

existsMetricsDef (empty-mapping,_) = false

getMetricsDef(modify(id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo range_to), How how, Units units, metricsdef), s_id)
= **if** id = s_id
then modify(id, (Name name, Description description, Type type, Kind kind, (RangeFrom range_from, RangeTo range_to), How how, Units units, metricsdef)
else getMetricsDef (metricsdef, s_id)

getMetricsDef (empty-mapping,_) = empty-mapping

addUnit (modify (id, (_,_, _, _,(_,_), _, Units units), metricsdef), s_id, unit)
= **if** id = s_id
then modify (id, (_,_, _, _,(_,_),_, Units add(units, unit)), metricsdef)
else modify (id, (_,_, _, _,(_, _),_, Units units), addUnits (metricsdef, s_id, unit))

addUnit (empty-mapping, _, _) = empty-mapping

removeUnit (modify(id, (_,_, _, _,(_,_), Units unit), metricsdef), s_id, unit)
= if id = s_id
then modify(id, (_,_, _, _,(_,_), Units. delete(units, unit)), metricsdef)
else modify(id, (_,_, _, _,(_,_), Units. units), removeUnit (metricsdef, s_id, unit))

removeUnit (empty-mapping,_,_) = empty-mapping

ATO MetricsCollected

INTERFACE

newMetricsCollected		→ MetricCollected
newMetric (_,_,_,_,_,_)	MetricCollected, String, String, Double, String, Date, Date	→ MetricCollected
removeMetric (_,_)	MetricCollected, String	→ MetricCollected
existsMetric (_,_)	MetricCollected, String	→ Boolean
getMetric (_,_)	MetricCollected, String	→ MetricCollected

VARIÁVEIS FORMAIS

metricscol	METRICCOLLECTED
artifactid, processid, activityid, resourceid, agenteid, groupid, allorganization	STRING
metricid	STRING
value	DOUBLE
unit	STRING
period_bg, period_end	DATE
metrics	SETOFMETRIC

OPERAÇÕES

newMetricsCollected = empty-set

newArtifactMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Artifact artifacid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

newProcessMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Process processid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

newActivityMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Activity activityid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

newResourceMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Resource resourceid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

newAgentMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Agent agentid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

newGroupMetric (metricscol, (artifactid, metricid, value, unit, period_bg, period_end))
= add(metricscol, (Group groupid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end))

getArtifactMetric (add((Artifact artifacid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Artifact artifacid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getArtifactMetric (metricscol, s_id)

getProcessMetric (add((Process processid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Process processid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getProcessMetric (metricscol, s_id)

getActivityMetric (add((Activity activityid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Activity activityid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getActivityMetric (metricscol, s_id)

getResourceMetric (add((Resource resourceid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Resource resourceid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getResourceMetric (metricscol, s_id)

getAgentMetric (add((Agent agentid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Agent agentid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getAgentMetric (metricscol, s_id)

getGroupMetric (add((Group groupid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then (Group groupid, MetricID metricid, Value value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end)
  else getGroupMetric (metricscol, s_id)

removeMetric(metricscol, s_id) = delete (metricscol, add(empty-set, getMetric(metricscol, s_id)))

existsMetric (add((_, MetricID metricid, Value_ value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then true
  else existsMetric (metricscol, s_id)

existsMetric (empty-set,_) = false

getMetric (add((_, MetricID metricid, Value_ value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol), s_id)
= **if** s_id = metricid
  then add((_, MetricID metricid, Value_ value, Unit unit, PeriodBegin period_bg, PeriodEnd period_end), metricscol)
  else getMetric (metricscol, s_id)

getMetric (empty-set,_) = empty-set

ATO Analysis

INTERFACE

newAnalysis		→ Analysis
addAnalysis (_,_,_,_,_)	Analysis, String, String, Text, Text	→ Analysis
updAnalysis (_,_,_,_,_)	Analysis, String, String, Text, Text	→ Analysis
removeAnalysis (_,_)	Analysis, String	→ Analysis
existsAnalysis (_,_)	Analysis, String	→ Boolean
getAnalysis (_,_)	Analysis, String	→ Analysis
addMeasurementGoal (_,_,_)	Analysis, String, MeasurementGoal	→ Analysis
removeMeasurementGoal (_,_,_)	Analysis, String, MeasurementGoal	→ Analysis

VARIÁVEIS FORMAIS

analyses	ANALYSIS
id, name	STRING
description, result	TEXTO
measurementgoal	MEASUREMENTGOAL
measurementgoals	SETOFMEASUREMENTGOAL
s_id, new_name	STRING
new_description, new_result	TEXTO

OPERAÇÕES

newAnalysis = empty-mapping

addAnalysis (analyses, id, name, description, result)
 = modify (id, (Name name, Description description, Result result, MeasurementGoals add(empty-set)), analyses)

updAnalysis (modify(id, (Name name, Description description, Result result, _), analyses), s_id, new_name, new_description, new_result)
= **if** id = s_id
  then modify(id, (Name new_name, Description new_description, Result new_result, _), analyses)
  else modify(id, (Name name, Description description, Result result, _), updAnalysis (analyses, s_id, new_name new_description, new_result))

updAnalysis(empty-mapping,_,_,_,_) = empty-mapping

removeAnalysis(analyses, s_id) = restrict_with (analyses, add(empty-set, s_id))

removeAnalysis (empty-mapping, _) = empty-mapping

existsAnalysis(modify(id, (Name name, Description description, Result result, MeasurementGoals measurementgoals), analyses), s_id)
= **if** id = s_id
  then true
  else existsAnalysis (analyses, s_id)

existsAnalysis (empty-mapping,_) = false

getAnalysis(modify(id, (Name name, Description description, Result result, MeasurementGoals measurementgoals), analyses), s_id)
= **if** id = s_id
  then modify(id, (Name name, Description description, Result result, MeasurementGoals measurementgoals), analyses)
  else existsAnalysis (analyses, s_id)

getAnalysis (empty-mapping,_) = empty-mapping

addMeasurementGoal (modify (id, (_,_, _, MeasurementGoals measurementgoals), analyses), s_id, measurementgoal)
= **if** id = s_id
  then modify (id, (_,_,_MeasurementGoals add(measurementgoals, measurementgoal)), analyses)
  else modify (id, (_,_,_,MeasurementGoals measurementgoals), addMeasurementGoals (analyses, s_id, measurementgoal))

addMeasurementGoal (empty-mapping, _, _) = empty-mapping

removeMeasurementGoal (modify(id, (_,_,_ MeasurementGoals. measurementgoal), analyses), s_id, measurementgoal)
= if id = s_id
then modify(id, (_,_,_ MeasurementGoals. delete(measurementgoals, measurementgoal)), analyses)
else modify(id, (_,_,_ MeasurementGoals. measurementgoals), removeMeasurementGoal (analyses, s_id, measurementgoal))

removeMeasurementGoal (empty-mapping,_,_) = empty-mapping

ATO Evaluations

INTERFACE

newEvaluation		→ Evaluation
addEvaluation (_,_,_,_,_,_)	Evaluation, String, String, Type, Text, Text	→ Evaluation
updEvaluation (_,_,_,_,_,_)	Evaluation, String, String, Type, Text, Text	→ Evaluation
removeEvaluation (_,_)	Evaluation, String	→ Evaluation
existsEvaluation (_,_)	Evaluation, String	→ Boolean
getEvaluation (_,_)	Evaluation, String	→ Boolean

VARIÁVEIS FORMAIS

evaluations	EVALUATION
id, name	STRING
description, results	TEXTO
s_id, new_name	STRING
new_description, new_results	TEXTO

OPERAÇÕES

newEvaluation = empty-mapping

addEvaluation (evaluations, id, name, type, description, results)
= modify (id, (Name name, Type, type, Description description, Results results), evaluations)

updEvaluation (modify(id, (Name name, Type, type, Description description, Results results), evaluations), s_id, new_name, new_type, new_description, new_results)
= **if** id = s_id
then modify(id, (Name new_name, Type, type, Description new_description, Results new_results), evaluations)
else modify(id, (Name name, Type, type, Description description, Results results), updEvaluation (evaluations, s_id, new_name new_type, new_description, new_results))

updEvaluation(empty-mapping,_,_,_,_,_) = empty-mapping

removeEvaluation(evaluations, s_id) = restrict_with (evaluations, add(empty-set, s_id))

removeEvaluation (empty-mapping, _) = empty-mapping

existsEvaluation(modify(id, (Name name, Type, type, Description description, Results results), evaluations), s_id)
= **if** id = s_id
then true
else existsEvaluation (evaluations, s_id)

existsEvaluation (empty-mapping,_) = false

getEvaluation(modify(id, (Name name, Type, type, Description description, Results results), evaluations), s_id)
= **if** id = s_id
then modify(id, (Name name, Type, type, Description description, Results results), evaluations)
else getEvaluation (evaluations, s_id)

getEvaluation (empty-mapping,_) = empty-mapping

ATO MeasurementProcess

INTERFACE

newMeasurementProcess		→ MeasurementProcess
addMeasurementProcess (_,_,_,_)	MeasurementProcess, String, String, Processes	→ MeasurementProcess
updMeasurementProcess (_,_,_,_)	MeasurementProcess, String, String, Processes	→ MeasurementProcess
removeMeasurementProcess (_,_)	MeasurementProcess, String	→ MeasurementProcess
existsMeasurementProcess (_,_)	MeasurementProcess, String	→ Boolean
getMeasurementProcess (_,_)	MeasurementProcess, String	→ MeasurementProcess

VARIÁVEIS FORMAIS

measurementprocesss	MEASUREMENTPROCESS
id, name	STRING

process	PROCESSES
s_id, new_name	STRING
new_process	PROCESSES

OPERAÇÕES

newMeasurementProcess = empty-mapping

addMeasurementProcess (measurementprocesses, id, name, process)
= modify (id, (Name name, Process process), measurementprocesses)

updMeasurementProcess (modify(id, (Name name, Process process), measurementprocesses), s_id, new_name, new_process)
= **if** id = s_id
then modify(id, (Name new_name, Process new_process), measurementprocesses)
else modify(id, (Name name, Process process), updMeasurementProcess (measurementprocesses, s_id, new_name, new_process))

updMeasurementProcess(empty-mapping,_,_,_) = empty-mapping

removeMeasurementProcess(measurementprocesses, s_id) = restrict_with (measurementprocesses, add(empty-set, s_id))

removeMeasurementProcess (empty-mapping, _) = empty-mapping

existsMeasurementProcess(modify(id, (Name name, Process process), measurementprocesses), s_id)
= **if** id = s_id
then true
else existsMeasurementProcess (measurementprocesses, s_id)

existsMeasurementProcess (empty-mapping,_) = false

getMeasurementProcess(modify(id, (Name name, Process process), measurementprocesses), s_id)
= **if** id = s_id
then modify(id, (Name name, Process process), measurementprocesses)
else getMeasurementProcess (measurementprocesses, s_id)

getMeasurementProcess (empty-mapping,_) = empty-mapping

ATO MeasurementProgram

INTERFACE

newMeasProgram		→ MeasurementProgram
addMeasProgram (_,_,_,_,_,_)	MeasurementProgram, String, String, String, InformationModel, MeasurementProcess	→MeasurementProgram
updMeasProgram (_,_,_,_,_,_)	MeasurementProgram, String, String, MasurementProcess, Text, Text	→MeasurementProgram
removeMeasProgram (_,_)	MeasurementProgram, String	→MeasurementProgram
existsMeasProgram (_,_)	MeasurementProgram, String	→ Boolean
getMeasProgram (_,_)	MeasurementProgram, String	→ MeasurementProgram

VARIÁVEIS FORMAIS

measprograms	MEASUREMENTPROGRAM
id, name, description	STRING
informationmodel	INFORMATIONMODEL
measurementprocess	MEASUREMENTPROCESS
s_id, new_name, new_description	STRING
new_informationmodel	INFORMATIONMODEL
new_measurementprocess	MEASUREMENTPROCESS

OPERAÇÕES

newMeasProgram = empty-mapping

addMeasProgram (measprograms, id, name, description, informationmodel, measurementprocess)
= modify (id, (Name name, Description description, InformationModel informationmodel, MasurementProcess, measurementprocess), measprograms)

updMeasProgram (modify(id, (Name name, Description description, InformationModel informationmodel, MasurementProcess, measurementprocess), measprograms), s_id, new_name, new_description, new_informationmodel, new_measurementprocess)
= **if** id = s_id
then modify(id, (Name new_name, Description new_description, InformationModel new_informationmodel, MasurementProcess measurementprocess), measprograms)
else modify(id, (Name name, Description description, InformationModel informationmodel, MasurementProcess, measurementprocess), updMeasProgram (measprograms, s_id, new_name, new_description, new_informationmodel, new_measurementprocess))

updMeasProgram(empty-mapping,_,_,_,_,_) = empty-mapping

removeMeasProgram(measprograms, s_id) = restrict_with (measprograms, add(empty-set, s_id))

removeMeasProgram (empty-mapping, _) = empty-mapping

existsMeasProgram(modify(id, (Name name, , Description description, InformationModel informationmodel, MasurementProcess measurementprocess), measprograms), s_id)
= **if** id = s_id
  then true
  else existsMeasProgram (measprograms, s_id)

existsMeasProgram (empty-mapping,_) = false

getMeasProgram(modify(id, (Name name, , Description description, InformationModel informationmodel, MasurementProcess measurementprocess), measprograms), s_id)
= **if** id = s_id
  then modify(id, (Name name, , Description description, InformationModel informationmodel, MasurementProcess measurementprocess), measprograms)
  else getMeasProgram (measprograms, s_id)

getMeasProgram (empty-mapping,_) = empty-mapping

ATO MeasurementPlan

INTERFACE

newMeasPlan		→ MeasurementPlan
addMeasPlan (_,_,_,_,_,_,_)	MeasurementPlan, String, String, Text, Program, Text, Text	→MeasurementPlan
updMeasPlan (_,_,_,_,_,_,_)	MeasurementPlan, String, String, Text, Program, Text, Text	→MeasurementPlan
removeMeasPlan (_,_)	MeasurementPlan, String	→MeasurementPlan
existsMeasPlan (_,_)	MeasurementPlan, String	→ Boolean
getMeasPlan (_,_)	MeasurementPlan, String	→ MeasurementPlan

VARIÁVEIS FORMAIS

measplans	MEASUREMENTPLAN
id, name	STRING
description, context, inform	TEXT
program	MEASUREMENTPROGRAM
s_id, new_name	STRING
new_description, new_context, new_inform	TEXT
new_program	MEASUREMENTPROGRAM

OPERAÇÕES

newMeasPlan = empty-mapping

addMeasPlan (measplans, id, name, description, program, context, inform)
 = modify (id, (Name name, Description description, Program program, Context, context, Informations inform), measplans)

updMeasPlan (modify(id, (Name name, Description description, Program program, Context context, Informations inform), measplans), s_id, new_name, new_description, new_program, new_context, new_inform)
= **if** id = s_id
  then modify(id, (Name new_name, Description new_description, Program new_program, Context context, Informations inform), measplans)
  else modify(id, (Name name, Description description, Program program, Context context, Informations inform), updMeasPlan (measplans, s_id, new_name, new_description, new_program, new_context, new_inform))

updMeasPlan(empty-mapping,_,_,_,_,_,_) = empty-mapping

removeMeasPlan(measplans, s_id) = restrict_with (measplans, add(empty-set, s_id))

removeMeasPlan (empty-mapping, _) = empty-mapping

existsMeasPlan(modify(id, (Name name, Description description, Program program, Context context, Informations inform), measplans), s_id)
= **if** id = s_id
  then true
  else existsMeasPlan (measplans, s_id)

existsMeasPlan (empty-mapping,_) = false

getMeasPlan(modify(id, (Name name, Description description, Program program, Context context, Informations inform), measplans), s_id)
= **if** id = s_id
  then (id, (Name name, Description description, Program program, Context context, Informations inform), measplans)
  else getMeasPlan (measplans, s_id)

getMeasPlan (empty-mapping,_) = empty-mapping

APÊNDICE B MODELAGEM DOS SUB-PROCESSOS DO EXEMPLO DE UTITILIZAÇÃO

Neste apêndice é apresentada a modelagem, utilizando o protótipo da ferramenta desenvolvida, dos sub-processos das atividades de coleta, análise, avaliação do processo de mensuração definido no exemplo de utilização do modelo. Também é apresentada a modelagem dos sub-processos de desenvolvimento de *software*, definidos para a organização em questão.

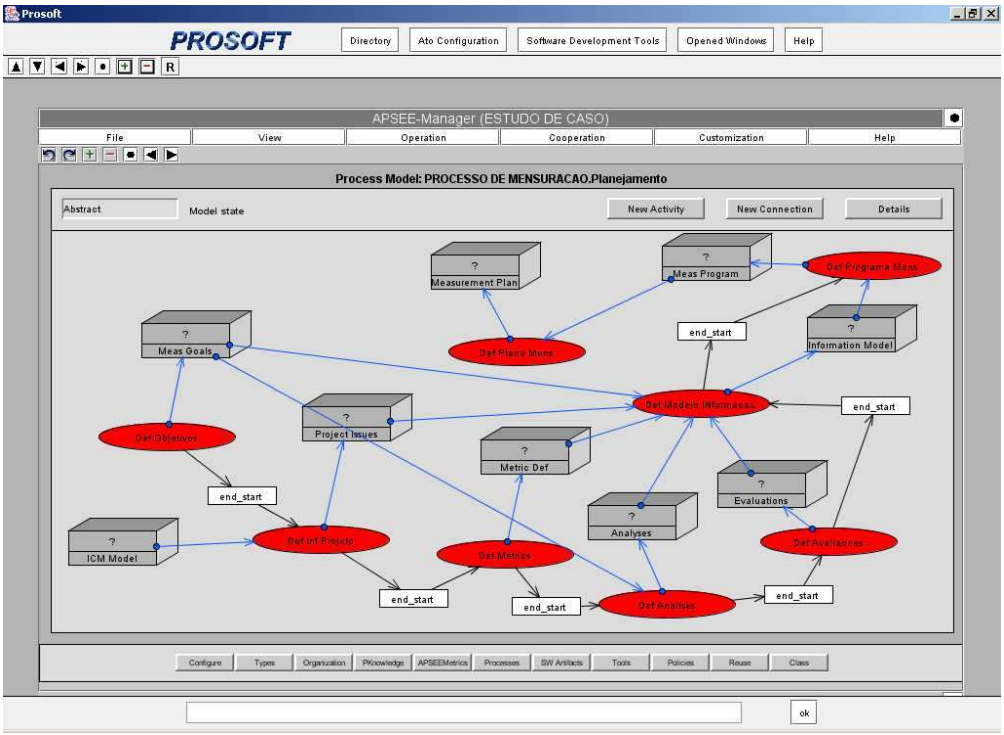


Figura B.1: Modelagem do sub-processo de planejamento

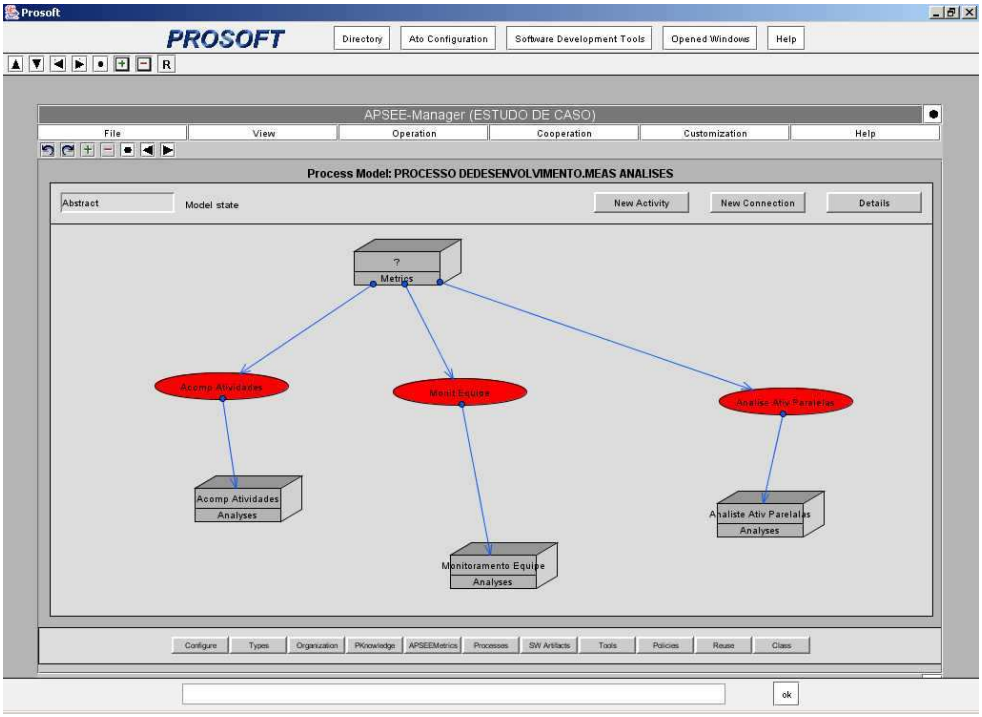


Figura B.2: Modelagem do sub-processo de análises da mensuração

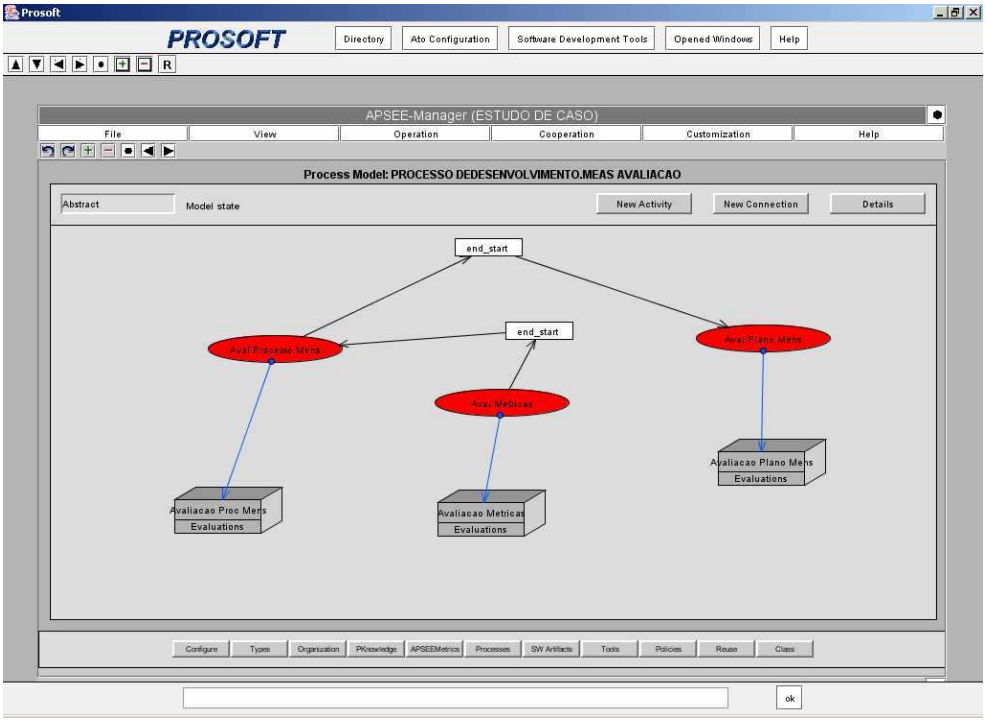


Figura B.3: Modelagem do sub- processo de avaliação da mensuração

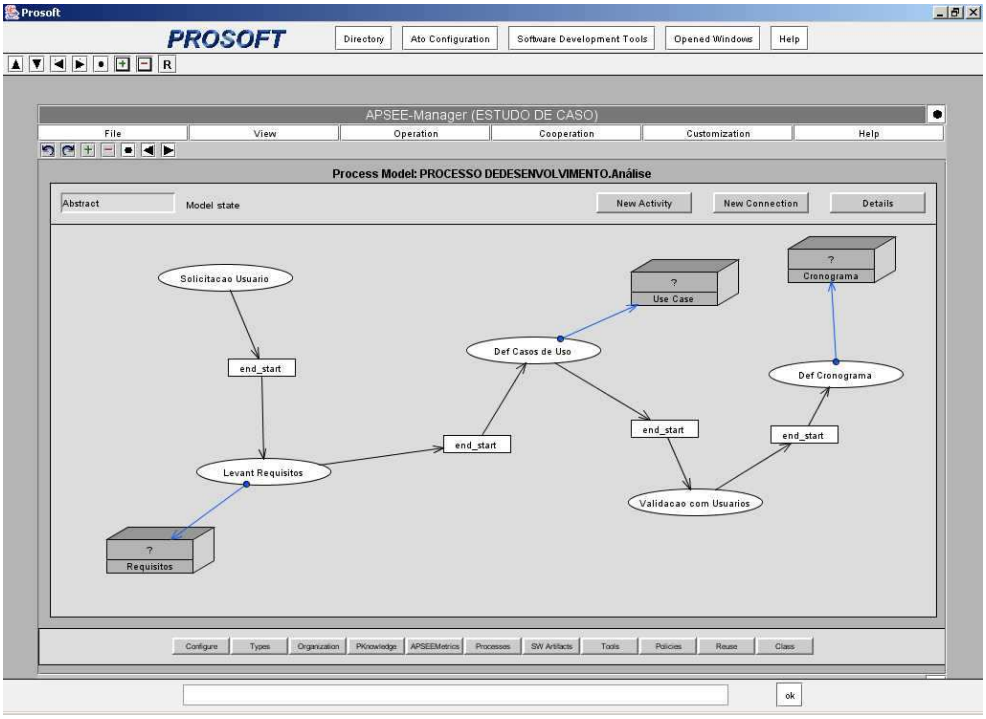


Figura B.4: Modelagem do sub-processo de análise

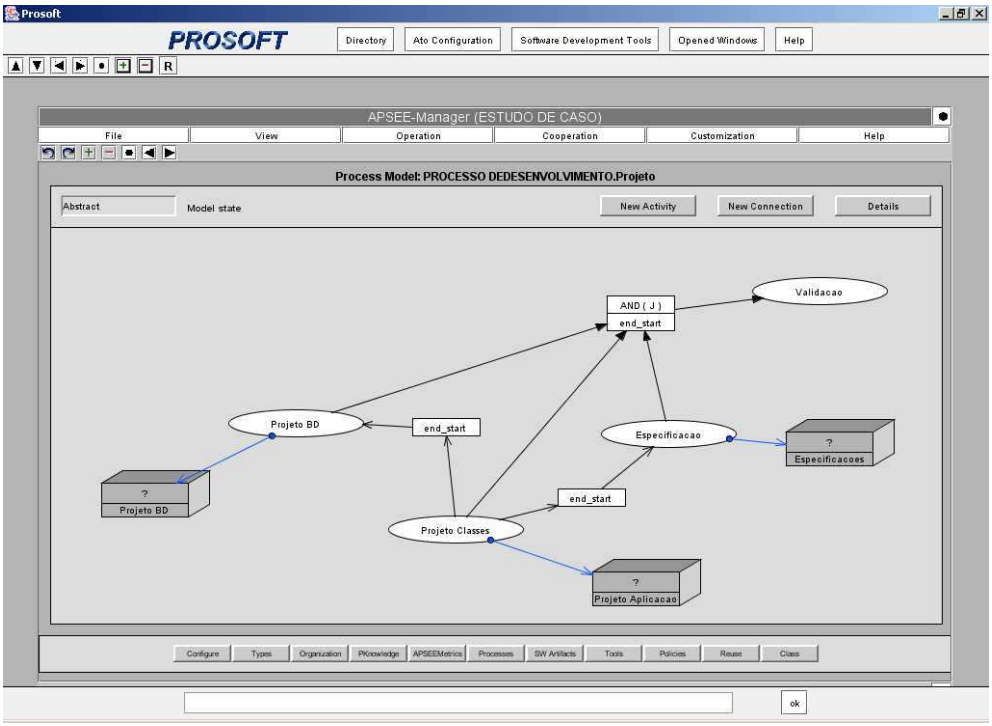


Figura B.5: Modelagem do sub-processo de projeto

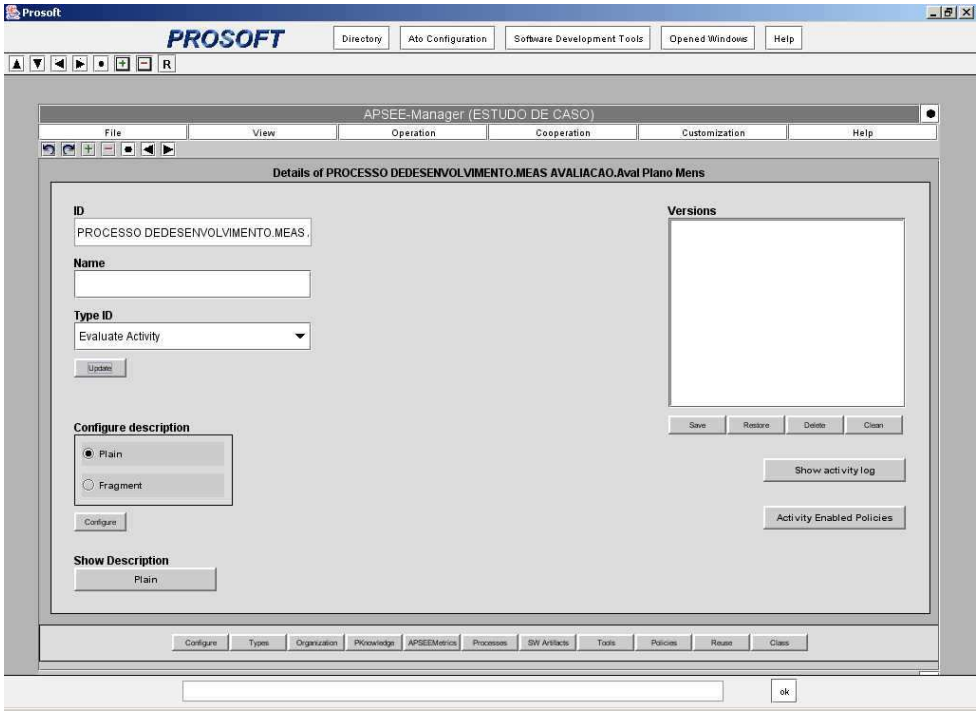


Figura B.6: Definição do tipo de atividade

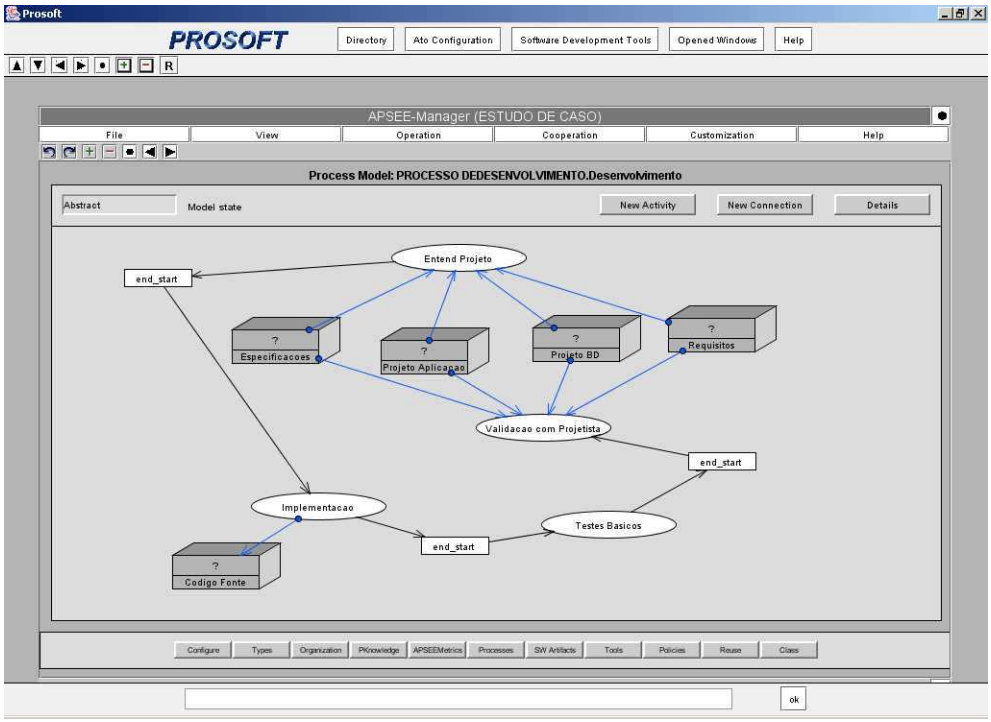


Figura B.7: Modelagem do sub-processo de implementação

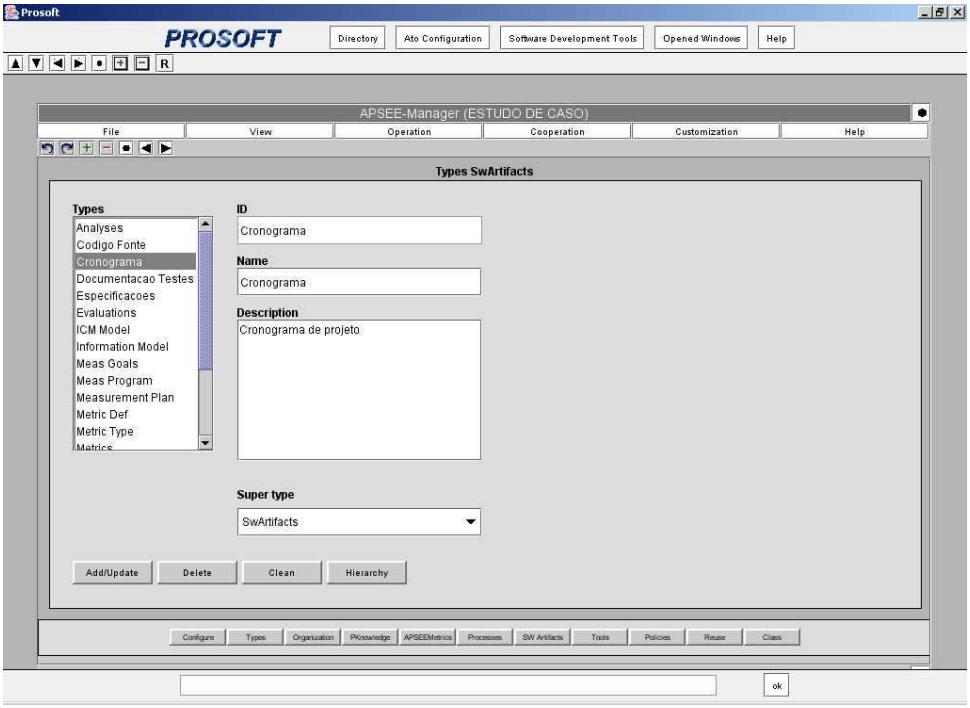


Figura B.8: Definição dos tipos de artefatos de *software*

ANEXO C TIPOS DE DADOS DO APSEE

Neste anexo são apresentadas as principais classes que compõem os tipos de dados de hierarquias de tipos, processos de software e artefatos de software, definidas em (REIS, 2003). Esses tipos de dados foram utilizados na especificação do APSEE-*Metrics*. A descrição das classes apresentadas foge ao escopo deste trabalho, podendo ser encontrada em (REIS, 2003).

- **Tipo Hierarquia de tipos**

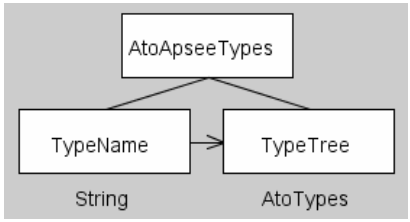


Figura C.1: Classe *APSEETypes*

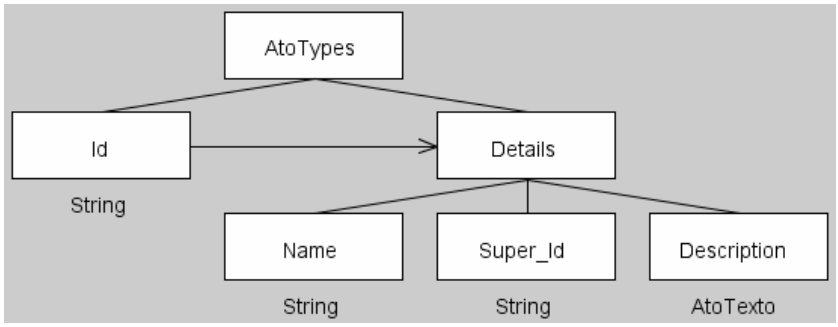


Figura C.2: Classe *Types*

- **Tipo Processos de software**

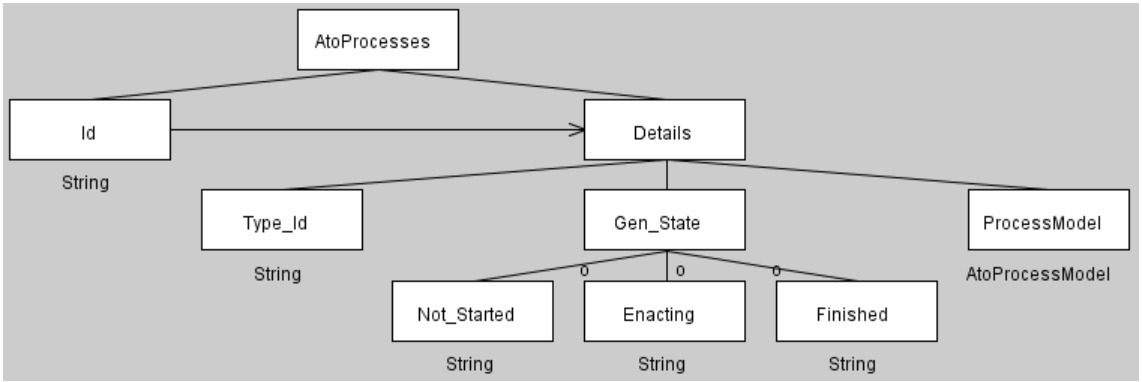


Figura C.3: Classe *Processes*

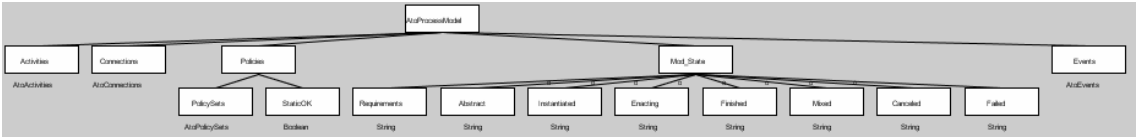


Figura C.4: Classe *ProcessModel*

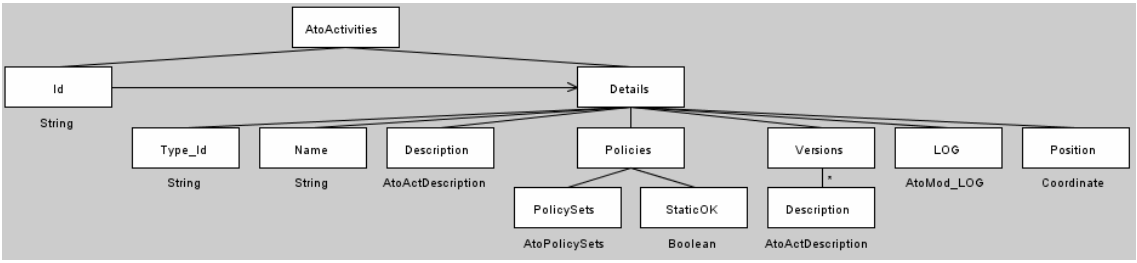


Figura C.5: Classe *Activities*

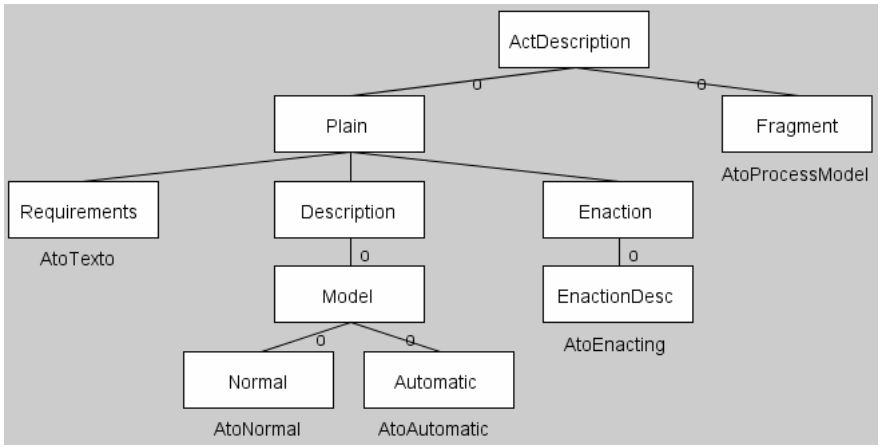


Figura C.6: Classe *ActDescription*

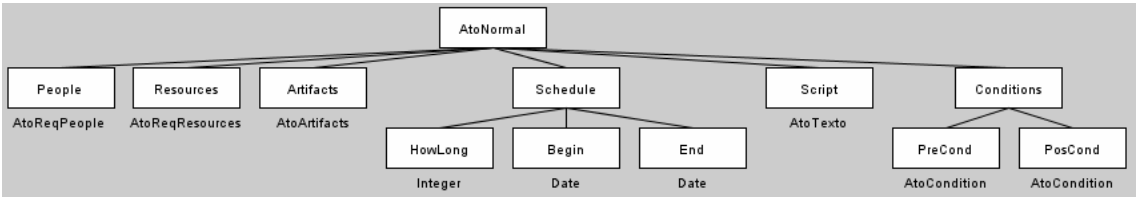


Figura C.7: Classe *Normal*

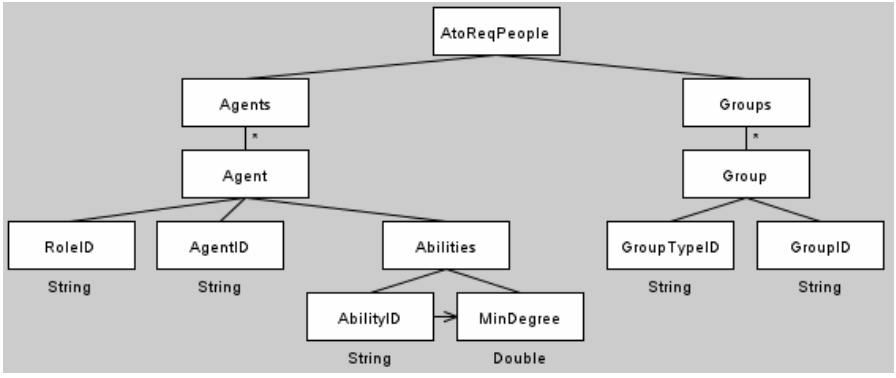


Figura C.8: Classe *ReqPeople*

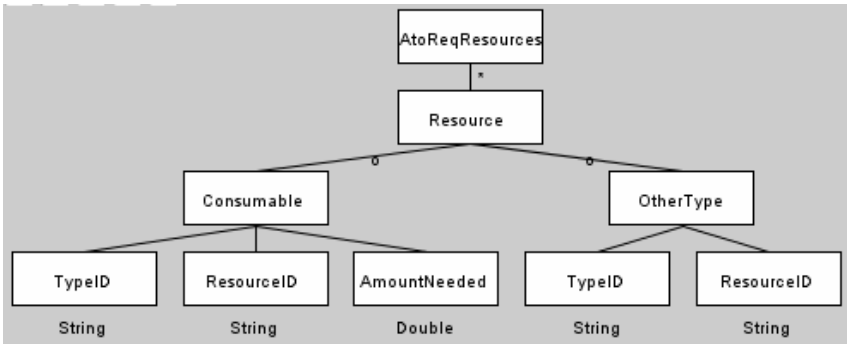


Figura C.9: Classe *ReqResources*

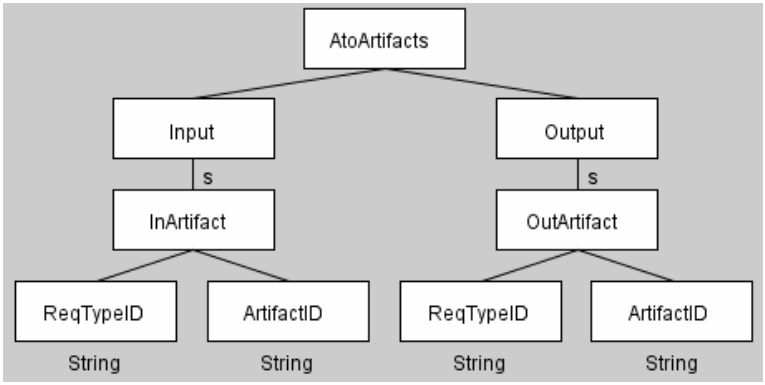


Figura C.10: Classe *Artifacts*

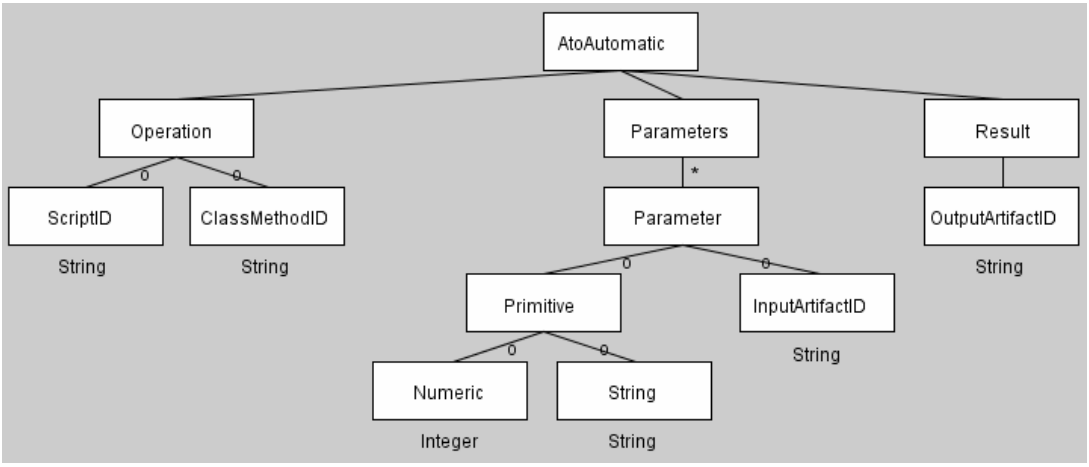


Figura C.11: Classe *Automatic*

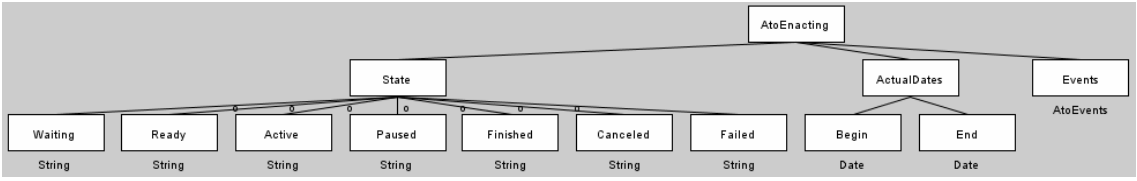


Figura C.12: Classe *Enacting*

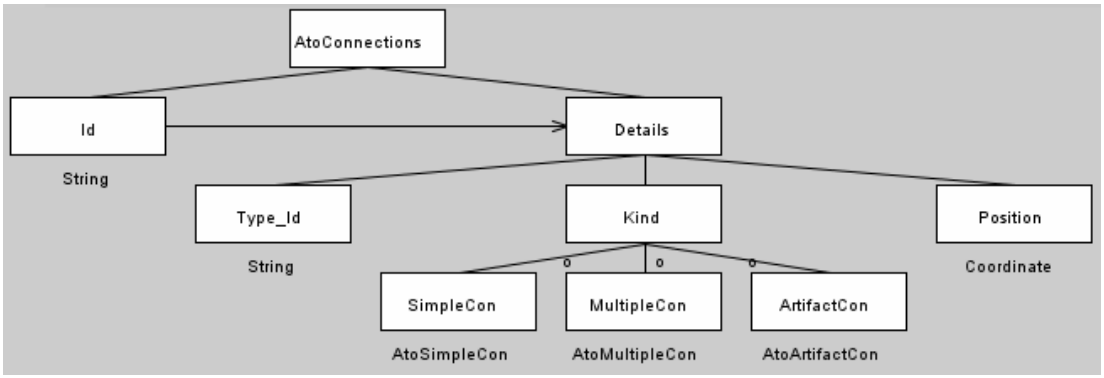


Figura C.13: Classe *Connections*

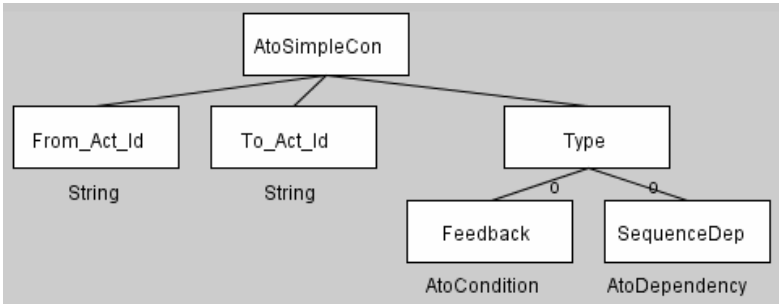


Figura C.14: Classe *SimpleCon*

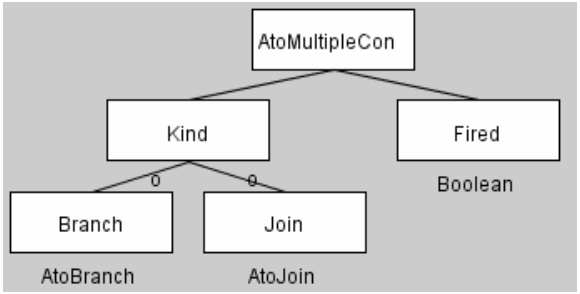


Figura C.15: Classe *MultipleCon*

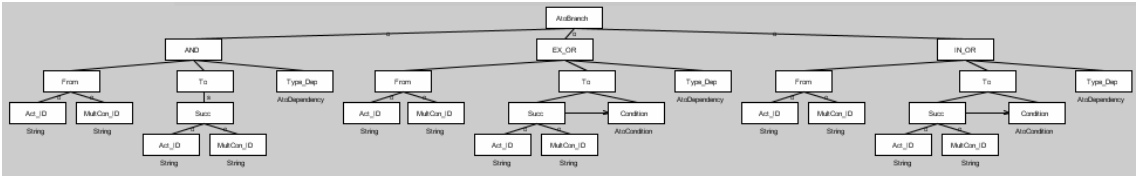


Figura C.16: Classe *Branch*

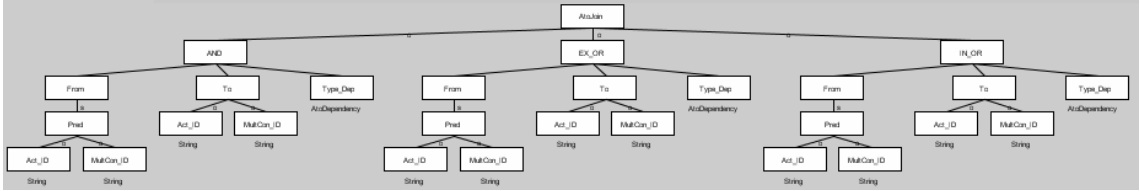


Figura C.17: Classe *Join*

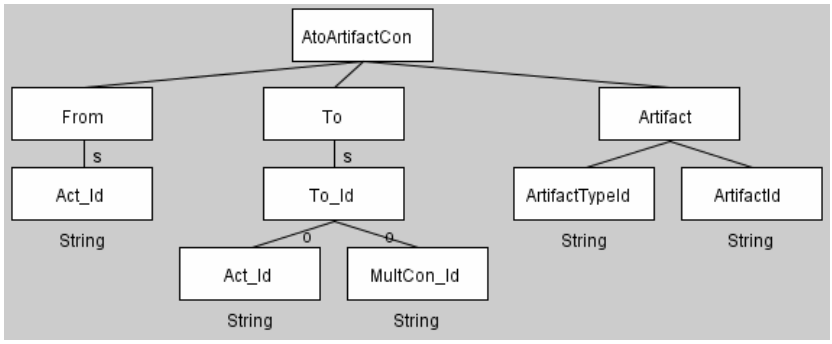


Figura C.18: Classe *ArtifactCon*

- **Tipo Artefatos de software**

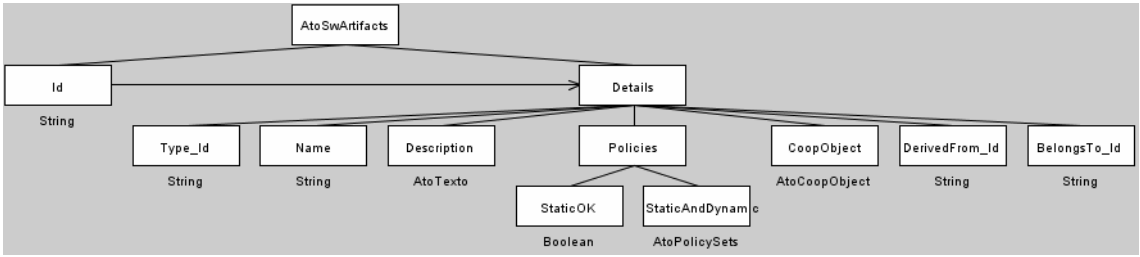


Figura C.19: Classe *SwArtifacts*

ANEXO D TABELA DO MODELO ICM

Neste anexo é apresentada a tabela correspondente ao modelo ICM (USA, 2003).

Issue - Category - Measure Mapping		
Common Issue Area	**Measurement Category**	**Measures**
Schedule and Progress	Milestone Performance	Milestone Dates Critical Path Performance
	Work Unit Progress	Requirements Status Problem Report Status Review Status Change Request Status Component Status Test Status Action Item Status
	Incremental Capability	Increment Content - Components Increment Content - Functions
Resources and Cost	Personnel	Effort Staff Experience Staff Turnover
	Financial Performance	Earned Value Cost
	Environment and Support Resources	Resource Availability Resource Utilization
Product Size and Stability	Physical Size and Stability	Database Size Components Interfaces Lines of Code Physical Dimensions
	Functional Size and Stability	Requirements Functional Change Workload Function Points
Product Quality	Functional Correctness	Defects Technical Performance
	Supportability - Maintainability	Time to Restore Cyclomatic Complexity Maintenance Actions
	Efficiency	Utilization Throughput Timing
	Portability	Standards Compliance
	Usability	Operator Errors
	Dependability - Reliability	Failures Fault Tolerance
Process Performance	Process Compliance	Reference Model Rating Process Audit Findings
	Process Efficiency	Productivity Cycle Time
	Process Effectiveness	Defect Containment Rework
Technology Effectiveness	Technology Suitability Impact Technology Volatility	Requirements Coverage Technology Impact Baseline Changes
Customer Satisfaction	Customer Feedback	Survey Results Performance Rating
	Customer Support	Requests for Support Support Time

ANEXO E OPERAÇÕES DO PROSOFT ALGÉBRICO

Neste anexo são apresentadas algumas das operações do Prosoft Algébrico, utilizadas para a especificação do modelo.

A operação construtora *add* e as outras operações do tipo Conjunto estão presentes na Tabela E.1, que contém suas funcionalidades e alguns exemplos de aplicação.

Tabela E.1: Operações do tipo conjunto

Operação	Funcionalidade	Exemplo
Add	Set x Component → Set	add ({x1,x2},x3) = {x1,x2,x3}
Belongs_to (∈)	Component x Set → Boolean	x1 ∈ {x1,x2} = TRUE
Cardinality (car)	Set → Nat	car ({x1,x2,x3}) = 3
Complement (\ )	Set x Set → Set	{x1,x2} \ {x2} = {x1}
Containtion (⊇)	Set x Set → Boolean	{x1,x2,x3} ⊇ {x1,x3} = TRUE
Delete	Set x Component → Set	delete ({x1,x2},x2) = {x1}
Empty set	→ Set	empty set = { }
Equal (=)	Set x Set → Boolean	{x1} = {x2} = FALSE
Intersection (∩)	Set x Set → Set	{x1,x2} ∩ {x2} = {x2}
Is_in	Set x Component → Boolean	is_in ({x1,x2},x1) = TRUE
Union (∪)	Set x Set → Set	{x1,x2} ∪ {x3} = {x1,x2,x3}

A operação construtora *modify* e as outras operações do tipo Mapeamento estão presentes na tabela E.2, onde contém suas funcionalidades e alguns exemplos de aplicação.

Tabela E.2: Operações do tipo composto Mapeamento

Operação	Funcionalidade	Exemplo
Composition (⊙)	Map x Map → Map	[x1 → y1] ⊙ [x2 → y2] = [x1 → y2]
Domain (dom)	Map → 2^{Set}	dom ([x1 → y1, x2 → y2]) = {x1,x2}
Empty-Mapping	→ Map	empty-mapping = { }
Image_of	Domain, Map → Rng	image_of (x1, [x1 → y1]) = y1
Merge (⌣)	Map x Map → Map	[x1 → y1] ⌣ [x2 → y2] = [x1 → y1,x2 → y2]
Modify	Domain x Rng x Map → Map	modify (x1, y1, [ ]) = [x1 → y1]
Override (+)	Map x Map → Map	[x1→y1,x2→y2]+[x2→y3]=[x1→y1,x2→y3]
Range (rng)	Map → Set	rng ([x1 → y1,x2 → y2]) = {y1, y2}
Restrict_to (|)	Map x 2^{Set} → Map	[x1 → y1,x2 → y2] | {x1} = [x1 → y1]
Restrict_with (\ )	Map x 2^{Set} → Map	[x1 → y1,x2 → y2] \ {x1} = [x2 → y2]

A operação construtora *cons* e as outras operações do tipo Lista estão presentes na tabela E.3, onde contém suas funcionalidades e exemplos de aplicação.

Tabela E.3: Operações do tipo composto Lista

Operação	Funcionalidade	Exemplo
Concatenation(^)	List x List → List	<x1, x2> ^ <x1> = <x1, x2, x1>
Cons	Component x List → List	cons (x1, <>) = <x1>
Elements (elems)	List → 2^{Set}	elems (<x1, x2, x1>) = {x1, x2}
Empty-list	→ List	empty-list = < >
Head (hd)	List → Component	hd (<x1, x2>) = x1
Index (inds)	List → 2^{Nat}	inds (<x1, x2, x1>) = {1, 2, 3}
Last	List → Component	last (<x1, x2, x1>) = x3
length (lng)	List → Nat	lng (<x1, x2, x1>) = 3
Projection (_[ ])	List x Nat → Component	<x1, x2> [2] = x2
Replace (_,+,[_])	List x Nat x Component → List	<x1, x2> + [2, x3] = <x1, x3>
Tail (tl)	List → List	tl (<x1, x2>) = x2

O tipo Registro possui uma operação construtora e outra operação do tipo observadora, *select* (seleciona um dos campos do registro). Essas operações estão presentes na tabela E.4, onde contém suas funcionalidades e exemplos de aplicação. Para facilitar a seleção (e identificação) dos campos dos registros, os tipos dos seus campos são precedidos por uma *tag* (rótulo sublinhado).

Tabela E.4: Operações do tipo composto Registro

Operação e Funcionalidade	Exemplo
(_,_,..,_): $C_1, C_2, ..., C_n$ → Record	(Rua rua, Bairro bairro, Cid cidade, Est estado, Cep cep)
Select-_: Tag Record → S_1 S_2 ... S_n	select-Bairro (_, Bairro Centro, _,_,_) = Centro