AVALIAÇÃO DE PROCESSOS DE *SOFTWARE* BASEADA EM MEDIÇÕES

Augusto Gonçalves Gomes Júnior

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Ana Regina Cavalcanti da Rocha, D.Sc.

Profa. Káthia Marçal de Oliveira, D.Sc.

Profa. Vera Maria Benjamin Werneck, D.Sc.

Dra.Carmen Maidantchik, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JULHO DE 2001

GOMES JÚNIOR, AUGUSTO GONÇALVES

Avaliação de Processos de Software baseada em Medições [Rio de Janeiro] 2001

VIII, 105 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2000)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Avaliação de Processos de Software

2. Métricas de Software

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AVALIAÇÃO DE PROCESSOS DE *SOFTWARE* BASEADA EM MEDIÇÕES

Augusto Gonçalves Gomes Júnior

Julho/2001

Orientadores: Ana Regina Cavalcanti da Rocha
Káthia Marçal de Oliveira

Programa: Engenharia de Sistemas e Computação

Com o intuito de aperfeiçoar o desenvolvimento de software e obter produtos com os níveis desejáveis de qualidade, a última década assistiu a uma mudança de enfoque com relação ao processo de software, visto que este tem se mostrado um dos fatores determinantes para o alcance da qualidade do produto final. A partir disto, intensificou-se a pesquisa sobre o processo de desenvolvimento e várias normas e padrões foram propostos para auxiliar na sua definição e melhoria. No entanto, constatou-se que, para alcançar níveis cada vez mais altos de qualidade, era necessário melhorar cada passo do ciclo de vida de desenvolvimento. Mas, para isso, dados quantitativos capazes de retratar a realidade dos processos precisavam ser obtidos e devidamente analisados. Neste contexto, medições de software têm se mostrado o fator chave para o aumento da qualidade dos processos, pois são a base para a identificação de suas forças e fraquezas. No entanto, definir, coletar e analisar um conjunto de métricas não é uma tarefa trivial e a maioria das propostas de implantação de métricas não obteve o êxito esperado.

Baseado neste panorama, este trabalho apresenta uma abordagem para o uso sistemático de métricas na avaliação dos processos de *software*. Esta proposta abrange a seleção e definição de métricas, a determinação dos procedimentos para a coleta dos dados e a análise dos resultados obtidos com a sugestão dos possíveis problemas existentes no processo.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SOFTWARE PROCESS EVALUATION BASED ON MEASUREMENT

Augusto Gonçalves Gomes Júnior

July/2001

Advisors: Ana Regina Cavalcanti da Rocha
Káthia Marçal de Oliveira

Department: System and Computing Engineering

In order to improve the software development and to produce high quality products within the proposed schedule and budget, the last decade has witnessed a focus change concerning the software process, once it happens to be one of the most important factors to reach the final product quality. Due to this change, research on the software development process was intensified and, several norms and standards were proposed in order to help developers in defining and improving software processes. However, it was observed that to produce better products, it was necessary to improve every step of the development life cycle. But to make this improvement possible, quantitative data, which could show us the reality of the process, must be obtained and properly analyzed. Then, measurement seems to be the key factor to reach a process improvement because it is the basis to identify its straights and weaknesses. However, contrary to what it may seems, the definition, collection and analysis of a group of metrics is a no trivial task and most measurement approaches did not succeed.

Based on this background, this work describes an approach to systematically use measurements to evaluate the software processes. This approach includes an adequate choice of metrics, their definition, procedures to collect data and suggestions of possible process problems based on data analysis.

Aos meus pais pelo apoio irrestrito.

Aos meus irmãos por serem bem mais que irmãos.

Aos meus avós por simplesmente tudo.

A todos os meus parentes e amigos que me ajudam a crescer.

À Carla por estar ao meu lado nesta caminhada.

Ao "Mestre Guerreiro" que mostrou a mim e à minha família a importância da vida.

A Deus pela vida.

AGRADECIMENTOS

À minha orientadora Ana Regina por sua amizade, orientação, apoio, incentivo e por acreditar no meu trabalho.

À Káthia por toda a sua paixão e entusiasmo com que acompanhou o desenvolvimento deste trabalho.

Ao Luis Filipe por nossas conversas, por seus comentários e por sua valiosa ajuda.

Aos professores Ricardo Falbo, Vera Werneck e Blascheck por contribuírem com seu conhecimento e experiência, fundamentais para que esta idéia se tornasse realidade.

À doutora Carmen e, mais uma vez, à professora Vera Werneck por aceitarem participar da banca.

Ao Gleison, Zlot e Sômulo pela ajuda e troca de conhecimentos.

Ao Gustavo S., Raquel, Ana Mirtes, Kelvin, Adriane, Ana Paula, Marluce, Flávio, Magnos, Marcelo, Andrômeda, Gustavo B. e demais amigos que direta ou indiretamente me acompanharam.

Ao pessoal administrativo da COPPE/Sistemas, Ana Paula, Claudia e Solange.

À CAPES pelo apoio financeiro.

Conteúdo

1. Introdução 1

1.1 Motivação 1

1.2 Objetivo da Tese 3

1.3 Organização da Tese 4

2. Métricas e o Processo de Desenvolvimento de *Software* 6

2.1 Introdução 6

2.2 Processo de Desenvolvimento de *Software* 7

2.3 Métricas 10

2.3.1 Escalas de Medições 11

2.3.2 Métricas Objetivas e Subjetivas 13

2.3.3 Seleção e Análise de Métricas 14

2.3.4 Definição de Métricas 16

2.3.5 Métricas da Literatura 18

2.4 Considerações Finais 22

3. A Estação TABA 23

3.1 Introdução 23

3.2 Estrutura 24

3.3 O Processo de Desenvolvimento de *Software* na Estação TABA 26

3.4 Definição de Processos 28

3.4.1 Ferramentas para a Definição de Processos 30

3.5 Ambientes de Desenvolvimento de *Software* 32

3.6 Considerações Finais 34

4. Avaliação de Processos de *Software* 35

4.1 Introdução 35

4.2 Considerações e Comprometimentos 35

4.3 Seleção e Definição das Métricas para Avaliação do Processo 38

4.4 Procedimento para a Construção da Estrutura de Decisão 45

4.5 Estrutura de Decisão para o Primeiro Objetivo 47

4.6 Medição do Processo de *Software* 52

4.7 Experiência de Medição 54

4.8 Considerações Finais 57

5. Avaliação de Processos de *Software* na Estação TABA 59

5.1 Introdução 59

5.2 Avaliação de Processos na Estação TABA 60

5.2.1 Medição do Processo 61

5.2.2 Análise dos Resultados da Medição 69

5.3 Considerações Finais 76

6. Considerações Finais 77

6.1 Conclusões 77

6.2 Perspectivas Futuras 79

Referências Bibliográficas 81

Anexo 1 - Métricas Utilizadas 87

Anexo 2 - Modelagem do Sistema Baseado em Conhecimento 95

Anexo 3 - Alterações no Modelo TABA 102

Capítulo 1
Introdução

Neste capítulo, são apresentadas as motivações que levaram à realização deste trabalho, seu objetivo principal e a forma como a tese está organizada.

1.1 Motivação

Ao longo das últimas décadas, os produtos de *software* sofreram um considerável crescimento de tamanho e complexidade e o seu papel na vida moderna ganhou tamanha importância que hoje já são considerados vitais para diferentes ramos da sociedade.

Contudo, à medida que este crescimento ocorre, o número de problemas enfrentados durante o desenvolvimento também aumenta. Segundo (HUMPHREY, 1989), a produtividade das equipes diminui à medida que crescem o tamanho e a complexidade dos sistemas, e isso ocorre devido à baixa qualidade dos sub-produtos que compõem o produto final. Assim, problemas de orçamento e cronograma podem ser constatados na grande maioria dos projetos e o grande número de defeitos encontrados nos produtos liberados para o uso indica a necessidade da criação de técnicas que aumentem e garantam a qualidade dos sistemas. Além disso, com a competitividade entre as empresas produtoras de *software*, a qualidade passou a não ser mais um diferencial competitivo, mas, sim, um requisito básico para a sobrevivência no mercado (FLORAC e CARLETON, 1999).

Com o intuito de aperfeiçoar o desenvolvimento de *software* e obter produtos com os níveis desejáveis de qualidade, dentro do cronograma e orçamento propostos, a última década assistiu a uma mudança de enfoque com relação à garantia da qualidade. Tem-se, então, uma nova abordagem na qual o foco principal das atenções não está mais nos produtos criados durante o desenvolvimento, mas, sim, no próprio processo produtivo, visto que este tem se mostrado um dos fatores determinantes para o alcance da qualidade do produto final.

A partir desta mudança de foco, intensificou-se a pesquisa sobre o processo de desenvolvimento e várias normas e padrões foram propostos, dentre os quais vale destacar a Norma ISO/IEC 12207 (1995), o CMM (*Capability Maturity Model*)

(PAULK *et al.*, 1995) e o SPICE (*Software Process Improvement and Capability dEtermination*) futura norma ISO/IEC 15504 (EMAM *et al.*, 1998), os quais auxiliam desenvolvedores na definição e melhoria de processos de *software*. Com o avanço dos estudos, chegou-se a conclusão de que, para alcançar níveis cada vez mais altos de qualidade, era necessário melhorar cada passo do ciclo de vida (OMAN e PFLEEGER, 1997).

Para tornar isto possível, dados quantitativos que pudessem descrever a realidade do processo precisavam ser obtidos e devidamente analisados. Muitas métricas foram, então, propostas e aplicadas em casos práticos a fim de alcançar os seguintes objetivos: i) melhorar o entendimento sobre atividades, produtos e recursos e, assim, estabelecer bases para comparação entre medições; ii) avaliar o andamento do projeto comparando com dados planejados; iii) fazer previsões sobre o futuro andamento do projeto baseado em comportamentos passados; iv) promover melhorias, identificando falhas, ineficiências e outras oportunidades para melhorar a qualidade do produto e o desempenho do processo (PARK *et al.*, 1996).

Porém, ao contrário do que possa parecer, definir, coletar e analisar um conjunto de métricas não é uma tarefa trivial. Na realidade, esta é uma tarefa custosa que demanda grande conhecimento para evitar que o seu uso não aumente ainda mais os problemas enfrentados durante o desenvolvimento. De acordo com PARK *et al.* (1996), este problema ocorre devido ao grande número de atributos que podem ser medidos durante um projeto. Uma escolha incorreta das métricas pode levar, além do próprio aumento desnecessário de esforço, a uma visão distorcida do processo, o que dificulta a sua análise e, muitas vezes, termina por orientar decisões equivocadas. Experiências com medições orientadas a objetivos mostraram a importância da definição prévia de metas para facilitar a escolha de métricas e a correta interpretação dos resultados obtidos nas medições.

Todas estas dificuldades fizeram com que a maioria das propostas de implantação de métricas para avaliação e melhoria de processos não tenham tido muito sucesso até os dias de hoje. Poucos foram os casos onde o seu uso não foi considerado ineficiente ou, até mesmo, desaconselhável. FENTON e NEIL (2000) defendem a idéia de que a razão para esta falta de sucesso pode ser atribuída ao fato de que as atividades de medição não tiveram como objetivo o principal requisito, que é prover as informações necessárias para apoiar as decisões gerenciais durante o ciclo de vida do *software*. Além disso, também argumentam que as metodologias tradicionais geralmente são orientadas

por modelos de regressão para estimativa de custo e de defeitos, o que provê pouca informação de caráter decisório para os gerentes. Na opinião destes pesquisadores, o futuro das métricas de *software* está no uso de métricas relativamente simples combinando diferentes aspectos do desenvolvimento de forma a permitir vários tipos de estimativas e avaliações. Assim, torna-se necessário analisar os fatores chave esquecidos pelas abordagens tradicionais como, por exemplo, a relação de causa e efeito e a combinação de evidências.

Além disso, são freqüentes os casos de uso de métricas simplesmente para alcançar níveis mais altos de maturidade. Quando este processo está mais relacionado com questões de *marketing* do que de real melhoria do processo de desenvolvimento, isto poderá não levar a ganho de qualidade e produtividade o que é, e sempre deve ser, seu objetivo principal (MACKEY, 2000).

Novas abordagens precisam ser definidas a fim de facilitar o trabalho de engenheiros de *software* na implantação de programas de medição de forma a tornar menos árduas as tarefas necessárias à definição, coleta e avaliação de métricas em projetos de *software*. E é neste contexto que este trabalho se insere.

1.2 Objetivo da Tese

Analisando o panorama descrito, foi considerado relevante um estudo mais aprofundado sobre o uso de métricas para orientar a avaliação e melhoria de processos. A partir desta pesquisa, foi determinado que o objeto fundamental deste trabalho seria a definição de uma abordagem para o uso sistemático de métricas na avaliação dos processos de *software*.

A abordagem proposta abrange desde a seleção e definição de métricas, o que deve ser baseado nos objetivos definidos para as medições, passando pela determinação dos procedimentos necessários para o correto levantamento dos dados durante a execução do projeto, até a análise dos resultados obtidos com a sugestão dos possíveis problemas que a equipe de desenvolvimento pode ter enfrentado durante o decorrer dos trabalhos. Os problemas identificados servem de base para que os gerentes possam concentrar seus esforços nas principais falhas encontradas na execução do processo e, quando possível, propor melhorias para corrigir seus problemas e otimizar seu desempenho.

Como parte integrante da tese, é proposto um sistema baseado em conhecimento capaz de apoiar o procedimento de avaliação de um processo de *software* utilizado em um projeto específico. Esta ferramenta é capaz de avaliar o processo utilizado se baseando, para isso, nos valores obtidos com o resultado das medições. O conhecimento inserido nesta ferramenta foi obtido através de um levantamento feito na literatura existente e através de entrevistas e questionários feitos com especialistas nas áreas de gerência de projetos, qualidade de *software*, processo e melhoria de processo. Este conhecimento, como utilizado, pode ser facilmente expandido e refinado à medida que a maturidade do processo e das equipes aumentem.

1.3 Organização da Tese

Este trabalho contém, além desta Introdução, mais cinco capítulos resumidamente descritos da seguinte forma:

No segundo capítulo, é apresentado o estudo realizado sobre o processo de desenvolvimento de *software* e sobre o uso de métricas na sua análise e melhoria. Serão apresentados os conceitos inseridos com a mudança de foco das atenções para o processo de *software* e, logo depois, define-se métricas de *software*, seus objetivos, restrições de uso e como deve ser feita a sua seleção, definição e análise.

No terceiro capítulo, é apresentada a Estação TABA, projeto no qual este trabalho está inserido. São discutidos suas motivações, sua estrutura e os principais trabalhos já desenvolvidos, com destaque para a definição de processos e instanciação de ambientes de desenvolvimento de *software*.

No quarto capítulo, é apresentada uma abordagem para a avaliação de processos de *software*, objeto principal deste trabalho. Neste, serão apresentados todos os passos que esta abordagem engloba, desde a seleção e definição das métricas que são utilizadas na avaliação do processo até a análise dos resultados obtidos com as medições.

No quinto capítulo, é apresentada uma proposta para a avaliação semi-automatizada de processos de *software* desenvolvida para apoiar a abordagem teórica descrita no quarto capítulo. Este sistema foi construído a partir da infra-estrutura da Estação TABA apresentada no capítulo 3.

E, finalmente, no sexto e último capítulo, serão apresentados o processo de realização desta tese, suas principais conclusões e as perspectivas futuras para a continuidade deste trabalho.

Capítulo 2

Métricas e o Processo de Desenvolvimento de *Software*

Neste capítulo, são apresentados os estudos realizados sobre processo de desenvolvimento de software e sobre o uso de métricas na sua análise e melhoria.

2.1 Introdução

Para tornar possível uma contínua melhoria na qualidade e produtividade dos processos de *software* tão necessária e almejada nos dias de hoje, é necessário um melhor entendimento dos recursos, produtos e processos utilizados durante a execução dos projetos. Assim como qualquer outra disciplina de engenharia, o desenvolvimento de *software* requer um mecanismo de medição para obter as informações sobre a execução do processo, necessárias para o seu correto entendimento e avaliação (BASILI *et al.*, 1994). O uso de dados quantitativos deve ter como principal objetivo o apoio à tomada de decisões durante o ciclo de vida de desenvolvimento (FENTON e NEIL, 2000). Neste contexto, medições de *software* têm se mostrado o fator chave para o aumento da qualidade dos processos, pois são a base para a identificação de suas forças e fraquezas facilitando a visualização das oportunidades de melhoria.

No entanto, um programa de métricas não deve ter objetivo em si mesmo, mas, sim, ser parte de uma estratégia maior para melhoria do processo de desenvolvimento de *software* (GRADY e CASWELL, 1987). O uso de métricas apenas provê dados referentes às entidades do mundo real envolvidas na execução dos projetos. Serão as ações tomadas a partir do resultado obtido com as medições que farão com que a qualidade dos produtos e o desempenho do processo aumentem. Desta forma, o objetivo principal desta abordagem não deve ser medir o processo, mas, sim, encontrar possibilidades para sua melhoria sendo utilizado, para isso, medições apenas como uma ferramenta de orientação (DASKALANTONAKIS, 1992).

Porém, para que uma abordagem como esta obtenha êxito, detalhes como a escolha das métricas, a sua definição consistente, o correto levantamento dos dados e a elaboração de um mecanismo de análise dos resultados devem ser devidamente estudados. Neste capítulo, serão apresentados os estudos realizados sobre processos de

software e sobre métricas quando, então, será feita uma análise de suas definições e um levantamento do seu uso na melhoria do processo de desenvolvimento.

2.2 Processo de Desenvolvimento de *Software*

O termo "processo de *software*" ainda é bastante discutido na literatura vigente e vários são os trabalhos que procuram formalizar o seu real significado. Somente a título de exemplo: HUMPHREY (1989) define processo como um conjunto de atividades, métodos e práticas utilizadas na produção e desenvolvimento de *software*; por outro lado, FLORAC *et al.* (1997) definem como uma organização lógica de pessoas, materiais, energia, equipamentos e procedimentos empregados na execução de atividades projetadas para produzir um resultado específico; e a ISO/IEC 12207 (1995) define como um conjunto de atividades inter-relacionadas, que transforma entradas em saídas.

Como este trabalho está inserido no projeto Taba, será seguida a definição de processo utilizada atualmente neste projeto que o descreve como uma coleção de atividades relacionadas que têm lugar durante o desenvolvimento de um produto de *software*. Para a execução de cada atividade, é necessário a utilização de recursos para tornar possível a geração de produtos durante o decorrer dos trabalhos. No capítulo 3, estes e outros conceitos relevantes à definição de processo serão mais bem detalhados.

A produção de *software* é um esforço coletivo, criativo e complexo e, por isso, encarar este desenvolvimento como um processo que precisa ser disciplinado tem ajudado significativamente a identificar suas diferentes dimensões e a encontrar os problemas que precisam ser analisados a fim de estabelecer práticas efetivas para a organização. O foco no processo permite que um grupo de indivíduos alinhe o comportamento e as atividades de cada membro no sentido de alcançar o objetivo comum. Assim, acredita-se que a qualidade do produto final está fortemente relacionada à qualidade do processo utilizado para o seu desenvolvimento e manutenção (FUGGETTA, 2000; GRADY e CASWELL, 1987; ZAHRAN, 1998; HUMPHREY, 1997). Quando um produto possui algum problema não se deve corrigir somente o defeito encontrado. É necessário corrigir o processo que permitiu que este fosse inserido, pois, desta forma, não será necessário corrigir problemas como este em trabalhos futuros (MACKEY, 2000).

No entanto, não existe um processo único passível de ser utilizado em qualquer projeto, pois nenhum projeto é idêntico ao outro. Variações nas políticas e procedimentos organizacionais, métodos e estratégias de aquisição, tamanho e complexidade do projeto, requisitos e métodos de desenvolvimento do sistema, entre outros fatores, influenciam na forma como um produto de *software* é adquirido, desenvolvido, operado e mantido (ISO/IEC 12207, 1995).

Além disso, questões relacionadas ao porte da empresa e à cultura organizacional, aos objetivos de projetos específicos, aos recursos disponíveis, às tecnologias de desenvolvimento, ao conhecimento e à experiência da equipe determinam algumas das características que devem nortear a definição do processo (MACHADO, 2000).

Assim, devido à sua importância e diversidade, constatou-se a necessidade de um estudo mais aprofundado sobre processo de *software* e surgiu, então, a necessidade de se entender, definir e modelar o processo para permitir esclarecimento sobre a maneira como sistemas complexos de *software* são desenvolvidos. Com esta modelagem, é possível melhorar o entendimento, gerenciar sua execução, estudar o seu aperfeiçoamento e automatizar parte da sua execução (ARAÚJO, 1998).

Inicialmente, foram propostas ferramentas capazes de automatizar parte do processo como a solução para os problemas. Neste sentido, foram desenvolvidas linguagens de modelagem de processo (PML) para permitir a representação explícita, precisa e compreensível de todos os detalhes e características de um processo de *software*. A partir disso, foram também propostos ambientes de desenvolvimento de *software* centrados em processo (PCSEE- *Process-Centered Software Engineering Environment)* que exploram estas definições explícitas para controlar um conjunto de ferramentas destinadas a apoiar etapas do processo. No entanto, as PML e, conseqüentemente, os PCSEE são muito complexos e extremamente sofisticados, o que dificulta muito o seu entendimento e uso, fazendo com que estes praticamente não tenham saído do ambiente acadêmico (FUGGETTA, 2000).

Além disso, percebeu-se também que o uso de ferramentas não é suficiente e, normalmente, o aspecto menos problemático é a tecnologia, enquanto processos e pessoas ainda são fatores críticos para o sucesso de um projeto (ZAHRAN, 1998). Assim, o uso destas ferramentas constitui um poderoso auxílio aos desenvolvedores, mas por si só não garantem o êxito.

Foram, então, propostos vários modelos e normas com o objetivo de aumentar a maturidade dos processos de *software*. Neste sentido, é importante destacar o trabalho da *International Standard Organization* (ISO) que estabeleceu uma norma padrão para processo de *software* propondo um *framework* com terminologia bem definida e contendo processos, atividades e tarefas que devem ser aplicados durante a aquisição, fornecimento, serviço, desenvolvimento, operação e manutenção de *software*. A norma descreve a arquitetura de um processo de forma geral, mas não especifica em detalhes como implementar ou desempenhar estas atividades, nem descreve formato ou conteúdo da documentação a ser gerada, o que deve ser definido pela organização que pretende utilizá-lo de acordo com suas necessidades e características particulares do projeto (ISO/IEC 12207, 1995).

Além disso, o *Software Engineering Institute* (SEI) propôs o *Capability Maturity Model* (CMM), o qual orienta como deve ser feita uma avaliação do processo utilizado por uma organização, identificando o seu nível de maturidade e permitindo encontrar as áreas chave do processo que precisam ser analisadas e melhoradas para aumentar este nível. Pode-se definir capacidade como o intervalo de resultados esperados que podem ser alcançados com o uso de um processo, e maturidade como a amplitude na qual um processo específico é definido, gerenciado, medido, controlado e executado (PAULK *et al.*, 1995).

E, finalmente, não se pode deixar de mencionar o trabalho desenvolvido no projeto SPICE (*Software Process Improvement and Capability dEtermination*) futura norma ISO/IEC 15504 que objetiva desenvolver um padrão de avaliação destinado tanto para melhoria do processo quanto para determinação da capacidade de uma organização, aplicável a qualquer domínio, necessidades de negócio ou tamanho da organização, sendo independente da estrutura da organização, modelos de ciclos de vida, tecnologias ou métodos de *software* (EMAM *et al.*, 1998).

Com o andamento das pesquisas, observou-se que à medida que as empresas e os projetos crescem, aumenta também a necessidade de uma constante melhoria da qualidade dos produtos de *software* e, conseqüentemente, dos processos utilizados para desenvolvê-los. Constatou-se que as definições de processo não podem ser estáticas, devendo permitir um constante refinamento de forma a atender às novas expectativas de um mercado cada vez mais competitivo e exigente.

Assim, era necessário melhorar cada passo do processo de desenvolvimento e isto somente se torna possível com o uso de dados quantitativos capazes de descrever a realidade vivida durante os projetos. Inúmeros pesquisadores passaram a estudar, definir e utilizar uma grande variedade de métricas com o objetivo de criar a teoria capaz de suprir esta necessidade.

2.3 Métricas

Para que se possa utilizar corretamente as métricas de *software* é preciso entender o seu objetivo e significado. Assim, medição é o processo através do qual números ou símbolos são atribuídos a propriedades das entidades do mundo real com o intuito de descrevê-las através de regras claramente definidas. Ou seja, medição é a forma de obter informações sobre atributos (características ou propriedades) de entidades (objetos ou eventos) do mundo real (FENTON e PFLEEGER, 1997).

A diferença entre medida e medição é que esta é um mapeamento a partir do mundo real para o mundo formal, enquanto medida é um número ou símbolo atribuído à entidade pelo uso deste mapeamento a fim de caracterizar um de seus atributos.

As medições que utilizam somente um atributo de uma entidade são chamadas de medições diretas, enquanto aquelas que utilizam uma composição de atributos são chamadas de indiretas.

Além disso, são quatro as principais razões para o uso de métricas de *software*. Primeiramente, estas são utilizadas para a caracterização e melhor entendimento dos processos, produtos, recursos e ambientes e, assim, estabelecer bases para comparação com trabalhos futuros. Em segundo lugar, são utilizadas para avaliar o andamento dos projetos, comparando os dados estimados com os valores medidos permitindo, assim, agir quando necessário. Além disso, podem ser utilizadas para realizar previsões. A partir do entendimento da interação existente entre entidades, são definidos modelos para relacioná-las de forma que, a partir do resultado de algumas, pode-se prever o resultados das outras. E, finalmente, mede-se também para melhorar a forma de trabalho, obtendo informações que podem ajudar a identificar barreiras, ineficiências, gargalos e outras oportunidades para aumentar a qualidade dos produtos e o desempenho dos processos.

Os principais componentes do desenvolvimento de *software* são processos, produtos e recursos os quais definem domínios de aplicação individuais para a mensuração, permitem diferentes estratégias e podem ser utilizados para classificar métricas de *software* da seguinte forma:

- Métricas de processos: procuram obter informações a respeito das atividades realizadas durante o desenvolvimento de *software*. Normalmente, estas métricas possuem alguma relação com a noção de tempo, devido à seqüência existente entre as atividades. Segundo FENTON e PFLEEGER (1997), as métricas de processo são mais difíceis de serem definidas devido, em grande parte, à falta de entendimento das atividades que compõem o processo.
- Métricas de produto: procuram obter informações a respeito de qualquer artefato que resulte da execução de uma atividade.
- Métricas de recursos: procuram obter informações a respeito de qualquer entidade necessária para a execução de uma atividade.

2.3.1 Escalas de Medições

A premissa básica para a criação destas formas de mapeamento (medições) é para que, através da manipulação de dados numéricos, seja possível entender o comportamento de entidades. Com isso, pode-se gerar conclusões que não poderiam ser visualizadas a partir de observações do mundo real. A partir da análise das transformações matemáticas, deve-se interpretar o seu significado real para, então, agir de forma apropriada.

No entanto, nem todos os mapeamentos são iguais e estas diferenças podem apresentar restrições no tipo de análise que pode ser realizada. As escalas de medição são determinadas pelos tipos de mapeamento utilizados. Desta forma, diferentes mapeamentos definem escalas de medição distintas. Apesar de existirem inúmeros tipos de escalas, serão apresentados apenas cinco, pois estes são os mais utilizados e ilustram o intervalo de possibilidades e questões que devem ser consideradas quando uma medição é realizada. São eles:

- Nominal: este é o primeiro e mais simples tipo de escala. Neste o valor do atributo é representado por um nome ou rótulo e, normalmente, é utilizado para a classificação de entidades. Por exemplo, uma métrica desta escala pode ser utilizada para classificar os tipo de linha de código em: executáveis,

comentários e linhas em branco. Métricas desta escala não possuem relação de ordem entre os diferentes tipos e qualquer representação simbólica ou numérica pode ser utilizada. Porém, nenhuma noção de magnitude poderá ser associada a sua representação.

- Ordinal: a escala ordinal, como o próprio nome diz, acrescenta a noção de ordem à escala nominal. Desta forma, métricas nesta escala permitem a realização de análises que não poderiam ser realizadas com as nominais. Por exemplo, pode-se medir a experiência de um membro da equipe como: sem experiência, com pouca experiência ou experiente. No entanto, ainda não é possível determinar a distância existente entre duas classes. Desta forma, apenas as análises que respeitarem estas características serão consideradas válidas ou significativas. Assim, o cálculo da média aritmética para a experiência de uma equipe não teria nenhum significado, enquanto a mediana poderia ser perfeitamente utilizada neste caso.
- Intervalo: esta escala possui a informação do tamanho dos intervalos que separam as classes e, a partir deste nível, é possível realizar somas e subtrações. Porém, como ainda não possui a noção de razão, não é possível realizar multiplicações ou divisões. Como exemplo, pode-se citar as escalas de temperatura Celsius e Fahrenheit. Quando uma cidade está com dez graus Celsius e outra está com vinte, pode-se dizer que a diferença de temperatura é de dez graus, porém não se pode dizer que uma tem o dobro da temperatura da outra.
- Racional: esta é a escala mais utilizada e, além da ordem e do tamanho dos intervalos entre classes, acrescenta a noção de razão entre as magnitudes. Esta escala já possui o elemento zero que representa total ausência do atributo medido e este é o ponto inicial desta escala que cresce em intervalos iguais. Como exemplo, pode-se determinar a medida de tamanho de algum objeto utilizando esta escala. Nesta, todas as funções aritméticas podem ser utilizadas gerando resultados significativos.
- Absoluta: para esta escala só existe uma maneira através da qual a medição pode ser realizada. Desta forma, a medição para este nível deve ser feita simplesmente contando o número de elementos do conjunto da entidade. Como exemplo, a quantidade de defeitos observados em uma etapa do

desenvolvimento de *software* só pode ser medida de uma maneira, contando os elementos do conjunto, ou seja, o número de defeitos encontrados. Nesta, novamente, todas as funções aritméticas produzem resultados significativos.

No entanto, medições podem progredir nos níveis de escala à medida que as organizações, práticas e ferramentas se tornam mais maduras. Quanto mais maduro é um processo, mais ricas (ou maduras) podem ser as métricas utilizadas (FENTON e PFLEEGER, 1997). Um exemplo disso é a forma como as temperaturas têm sido medidas com o passar dos anos. Antigamente, não se podia determinar exatamente a temperatura de algum objeto ou líquido, sendo possível, apenas, uma ordenação entre temperaturas sem o conhecimento do intervalo entre estas (escala ordinal). No entanto, com os estudos e o desenvolvimento de ferramentas, como, por exemplo, o termômetro, esta métrica mudou de escala passando para os níveis mais abrangentes. Na sua mudança de nível mais recente, foi definida a escala Kelvin que já possui a noção de zero absoluto e constitui uma escala racional para medir a temperatura.

É importante enfatizar que a engenharia de *software* é uma disciplina ainda bastante recente e, assim, não é possível definir métricas de escala abrangente para todas as entidades deste domínio. É necessário começar o esforço de medição com dados mais simples a fim de aumentar a experiência e a maturidade. No entanto, não se pode esquecer que a manipulação e a análise dos dados estão intimamente ligadas à escala utilizada. Somente com o entendimento das características e restrições dos tipos de escala que se torna possível determinar se alguma conclusão obtida a partir de uma análise ou manipulação de dados numéricos tem significado válido no mundo real.

2.3.2 Métricas Objetivas e Subjetivas

Métricas subjetivas são aquelas que dependem do ambiente em que são coletadas e podem variar de pessoa para pessoa refletindo o julgamento de quem realizou a medição. Além disso, se uma mesma pessoa medir um atributo em momentos diferentes poderá encontrar respostas distintas, mesmo que o atributo não tenha se modificado. Ao contrário, métricas objetivas permitem que um mesmo resultado seja obtido independentemente da pessoa que realizou a medição, do momento ou do ambiente em que isto foi feito. Para isso, basta que a condição em que se encontrava o atributo não tenha se modificado.

Por este motivo, ao selecionar uma métrica é interessante optar pelas objetivas sempre que for possível. No entanto, não se deve desprezar o valor de uma métrica subjetiva quando esta for capaz de ajudar na caracterização, avaliação, previsão ou melhoria de algum processo ou produto.

O principal fator que deve ser analisado neste ponto é se o procedimento de medição pode ser repetido gerando o mesmo resultado caso o atributo medido não seja alterado. O uso de métricas objetivas facilita esta repetição, mas isto não impede que as subjetivas também sejam utilizadas. No entanto, para isto, pode ser necessário um estudo mais apurado dos modelos e técnicas para permitir o seu uso de forma consistente como, por exemplo, cálculos estatísticos e análises de tendências. Além disso, ao analisar métricas deste tipo, deve-se ter em mente que a possibilidade de desvios neste caso é maior.

2.3.3 Seleção e Análise de Métricas

O número de métricas que podem ser utilizadas em uma análise é bastante vasto e maior que os recursos de qualquer organização para coletar, apresentar e utilizar estes dados de forma efetiva. Isto faz com que seja necessário selecionar métricas identificando um pequeno subconjunto para ser utilizado em um momento específico (OMAN e PFLEEGER, 1997). No entanto, este conjunto deve ser abrangente o bastante para conter todas as informações necessárias para apoiar a tomada de decisão de forma completa. A arte de um programa de medição está em decidir quais atributos devem ser utilizados para obter uma visão útil das entidades que devem ser analisadas, gerando o mínimo de esforço extra possível (PARK *et al.*, 1996).

De maneira geral, um projeto de *software* pode ser analisado a partir de duas perspectivas distintas: uma *bottom-up* e a outra *top-down*.

A visão *bottom-up* começa a partir dos processos, produtos e recursos do desenvolvimento, procurando estabelecer como estes relacionam entre si e com os objetivos definidos para o projeto. O foco desta abordagem está nos elementos básicos que compõem o núcleo do desenvolvimento. A partir da definição destes elementos, é especificado um conjunto de medições que devem ser realizadas em qualquer projeto de forma que seja possível responder a, praticamente, qualquer tipo de pergunta. Assim, isto o torna independente dos objetivos organizacionais definidos para o projeto.

No entanto, PARK *et al.* (1996) argumentam que são tantas as entidades do mundo real que podem ser analisadas em um programa de medição que este pode gerar um esforço maior que os benefícios que poderá trazer. Com isso, foi definida uma segunda forma de abordagem: a *top-down*. Esta visão começa a partir dos objetivos definidos para o projeto e procura definir como estes podem ser divididos em conjuntos de produtos, recursos e processos capazes de suportar a sua análise. Ou seja, as métricas são selecionadas de forma a atender objetivos claramente definidos (OMAN e PFLEEGER, 1997). O modelo *top-down* mais conhecido é o *Goal/Question/Metric* (GQM) proposto por BASILI *et al.*(1994) que será apresentado em detalhes a seguir.

A Abordagem *Goal/Question/Metric*

A abordagem GQM é baseada na suposição de que para uma organização medir de maneira eficaz, esta precisa especificar primeiro os objetivos da organização e dos projetos e, então, relacioná-los com os dados que os definem de forma operacional. Para isso, é preciso definir um *framework* capaz de interpretar estes dados segundo os objetivos definidos.

Assim, é importante tornar claro, pelo menos de maneira geral, quais são as necessidades de informação de forma que estas possam ser quantificadas e analisadas a fim de responder se os objetivos foram ou não alcançados.

O resultado da aplicação desta abordagem é a definição de um sistema de medição visando um conjunto particular de questões e baseado em um conjunto de regras para a interpretação dos dados obtidos nas medições. O modelo de medição resultante possui três níveis:

- Nível Conceitual (*Goal*): um objetivo é definido para um objeto, por uma variedade de razões, segundo vários modelos de qualidade, de vários pontos de vista e relativo a um ambiente específico. Todo objetivo, definido nesta abordagem, deve ser composto das seguintes informações: propósito da medição, questão a ser analisada, objeto a ser medido e o ponto de vista que deve ser levado em consideração no momento da medição.
- Nível Operacional (*Question*): é utilizado um conjunto de questões para caracterizar como a avaliação ou o alcance de um objetivo específico será determinado, respeitando algum modelo de caracterização. Estas questões

procuram caracterizar o objeto de medição, respeitando questões de qualidade e procurando definir sua qualidade a partir do ponto de vista selecionado.

- Nível Quantitativo (*Metric*): um conjunto de dados é associado a todas as questões de modo a respondê-las de forma quantitativa. Os dados podem ser objetivos ou subjetivos.

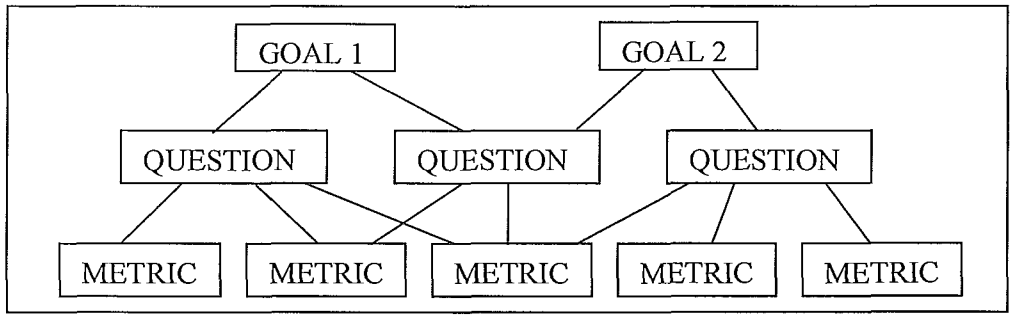


Figura 2.1: Estrutura Hierárquica do Modelo GQM.

Como pode ser visto na figura 2.1, o modelo GQM é uma estrutura hierárquica que parte de um objetivo e o refina em várias questões que, normalmente, divide a questão em seus principais componentes. A partir disto, cada questão é novamente refinada em uma ou várias métricas necessárias para que a questão seja avaliada. É importante observar que uma mesma métrica pode ser necessária para analisar mais de uma questão e uma mesma questão também pode ser necessária para analisar mais de um objetivo. Sempre que isto ocorrer, é importante que o ponto de vista de cada objetivo seja considerado no momento da medição, ou seja, uma mesma métrica poderá ter valores diferentes quando analisada segundo pontos de vista distintos.

Em suma, a abordagem GQM é um mecanismo para selecionar e analisar medições de *software* que pode ser utilizado de forma isolada ou dentro de um contexto de uma abordagem mais geral para a melhoria da qualidade de *software*. Ele combina conceitos de várias abordagens e os generaliza de forma que possa ser utilizado para analisar processos, produtos e, até mesmo, recursos. Isto o torna mais fácil de ser adaptado como pode ser constatado pelo seu uso em diversas organizações (BASILI *et al.*, 1994; DASKALANTONAKIS, 1992; GRADY e CASWELL, 1987).

2.3.4 Definição de Métricas

Uma vez identificadas as métricas que deverão ser utilizadas, estas ainda precisam ser claramente definidas de modo a facilitar a sua repetição e comunicação. Este é um

passo importante para um programa de medição, pois quando os usuários dos dados não sabem como estes foram coletados e quais as suas definições, visões distorcidas da realidade podem ser geradas o que poderá levar a decisões equivocadas.

Pode-se dizer que não existe nenhuma métrica absolutamente objetiva, pois sempre haverá algum grau de subjetividade na caracterização das entidades e seus atributos (FENTON e PFLEEGER, 1997). Além disso, a comunicação sem ambigüidades dos resultados da medição é inerentemente difícil (PARK *et al.*, 1996).

Desta forma, neste passo, deve-se criar especificações para cada métrica descrevendo suas definições operacionais a fim de diminuir ao máximo seu grau de subjetividade. Neste sentido, vários pesquisadores desenvolveram diferentes formas de definição de métricas e vale destacar o trabalho desenvolvido por CARLETON *et al.* (1992) que definem *checklists* a serem preenchidos de forma a indicar explicitamente todos os detalhes que estão sendo incluídos no levantamento de cada métrica e, também, aqueles que estão sendo excluídos. Na figura 2.2, pode ser visto um exemplo de *cheklist* utilizado para descrever a métrica utilizada para obter o tamanho de um *software* em número de linhas de código.

Definition Checklist for Source Statement Counts

Definition name: *Physical Source Lines of Code (basic definition)* Date: *8/7/92*
Originator: *SEI*

Measurement unit: Physical source lines ✓ / Logical source statements ☐

Statement type	Definition ✓	Data array ☐		Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.					
1 Executable			Order of precedence -> 1	✓	
2 Nonexecutable					
3 Declarations			2	✓	
4 Compiler directives			3	✓	
5 Comments					
6 On their own lines			4		✓
7 On lines with source code			5		✓
8 Banners and nonblank spacers			6		✓
9 Blank (empty) comments			7		✓
10 Blank lines			8		✓
11					
12					

How produced	Definition ✓	Data array ☐	Includes	Excludes
1 Programmed			✓	
2 Generated with source code generators			✓	
3 Converted with automated translators			✓	
4 Copied or reused without change			✓	
5 Modified			✓	
6 Removed				✓
7				
8				

Figura 2.2: *Checklist* definido para a métrica de tamanho.

Além destas questões, ainda é importante considerar o conhecimento da equipe e a maturidade do processo a ser medido. Uma característica que pode ser observada entre os níveis de maturidade é a habilidade de desenvolvedores e gerentes em visualizar e entender o que realmente acontece durante o processo de desenvolvimento. À medida que a maturidade aumenta, os processos se tornam mais claros, bem definidos e entendidos. A definição de cada métrica depende deste nível de maturidade, pois deve estar condizente com a capacidade de visualização da equipe do projeto. Desta forma, um programa de medição deve começar pelos problemas mais críticos ou objetivos principais de cada projeto (uso do GQM), utilizando métricas simples de forma a analisar somente o que é visível ou significativo para o nível em que se encontra o processo da organização. E, como parte da gerência, o programa de métricas deve evoluir à medida que a maturidade do processo e da equipe aumentem.

2.3.5 Métricas da Literatura

São inúmeras as métricas que podem ser empregadas para avaliar processos, produtos e recursos utilizados durante o desenvolvimento de *software*. No entanto, CARLETON *et al.* (1992) definem um conjunto básico de métricas simples que qualquer programa de medição, de alguma forma, deve levar em consideração. Este conjunto procura analisar características importantes para o entendimento, planejamento, acompanhamento e melhoria de processos e produtos.

- **Tamanho:** o tamanho do *software* deve ser medido em número de linhas de código, excluindo linhas em branco e linhas contendo apenas comentários. Além desta, existem outras formas de medir o tamanho do *software* como, por exemplo, pontos por função. No entanto, recomenda-se utilizar linhas de código devido à facilidade de medição, relativa independência em relação à linguagem de programação (sendo necessário somente diferenciar as formas de comentário) e devido à maior parte dos dados históricos utilizados para a construção de modelos de estimativa de custo ser baseada no número de linhas de código. Sabe-se que esta não é a melhor métrica a ser utilizada para medir o tamanho de um *software*, mas para utilizar outras, como, por exemplo, pontos por função, ainda é necessário um maior conhecimento por parte da equipe o que dificulta o seu uso direto.

- **Esforço:** o esforço empregado em uma atividade ou em um projeto deve ser medido em homens-hora. Um homem-hora é o tempo de uma hora gasto por um membro da equipe (IEEE P1045/D5.0, 1992). CARLETON *et al.* (1992) defendem a utilização de homem-hora ao invés de outras medidas como homem-mês ou homem-dia devido à dificuldade em estabelecer consenso do número de horas trabalhadas por dia ou mês. Além disto, a menor granularidade dos dados pode auxiliar o detalhamento em atividades específicas. É interessante que os dados de esforço estimados para o projeto ou atividade sejam comparados com os dados reais para permitir um melhor acompanhamento do projeto e uma análise das estimativas de projeto. Além disso, o esforço empregado em atividades de retrabalho também deve ser levantado, pois constituem um importante fator de análise para a melhoria do processo (PARK *et al.*, 1996).
- **Cronograma:** deve-se acompanhar o cronograma de um projeto armazenando as datas estimadas e reais associadas aos marcos e pontos de controle definidos para o projeto. Desta forma, pode-se obter o tempo estimado e real em número de dias para todo o projeto e para cada uma de suas principais atividades. O cronograma é um componente chave para a gerência dos projetos tanto para o acompanhamento de sua execução como para a análise do momento da distribuição do produto para seu público alvo. Além disso, deve ser utilizada, também, para a análise das estimativas de cronograma.
- **Defeitos:** o número de defeitos encontrados no *software* deve ser medido para que seja possível analisar a qualidade dos produtos que estão sendo desenvolvidos. Esta é uma informação fundamental para que a gerência encontre os problemas existentes no processo. A partir da definição das origens dos problemas, pode-se melhorar a qualidade dos produtos e o desempenho do processo. Desta forma, ROBERTS (1996) diferencia os defeitos encontrados na própria atividade do processo em que foram gerados e aqueles que passaram pela avaliação final, sendo somente descobertos em atividades posteriores. Como pode ser visto na figura 2.3, os defeitos que se encontram na diagonal da matriz são erros encontrados antes que o produto da atividade fosse aceito, enquanto os defeitos acima da reta indicam que modificações foram necessárias após a sua aceitação.

		ATIVIDADES		
		ANÁLISE	PROJETO	CODIFICAÇÃO
DOCUMENTOS	REQUISITOS	10	15	1
	PROJETO	-	531	87
	CÓDIGO	-	-	1235

Figura 2.3: Defeitos encontrados durante o desenvolvimento.

Além destas métricas principais, outros trabalhos propõem a combinação de alguns destes dados com o objetivo de gerar métricas compostas que, juntas, provêem mais informação do que quando usadas em separado. Dentre estas vale citar:

- **Densidade de Defeitos:** esta métrica é definida como a razão entre o número de defeitos encontrados durante o desenvolvimento de um produto de *software* e o seu tamanho. Utilizando esta métrica é possível estabelecer bases para comparação entre diferentes projetos. Assim, o acompanhamento da qualidade pode ser feito baseado em projetos anteriores, podendo ser utilizado para encontrar os módulos do sistema com a maior concentração de defeitos e, assim, priorizar a sua análise (FENTON e PFLEEGER, 1997).
- **Deterioração do *Software*:** esta é uma métrica utilizada por empresas japonesas para analisar a qualidade dos produtos de *software* liberados para os usuários. Esta é definida como a razão entre o esforço necessário para corrigir defeitos encontrados após a liberação do *software* para uso e todo o esforço empregado durante o desenvolvimento. Assim, esta também permite que bases para comparação sejam definidas para analisar a qualidade dos produtos gerados em projetos diferentes. Porém, neste caso, é enfatizada a percepção de qualidade observada pelos usuários, pois esta métrica é utilizada somente após a liberação do sistema. Quanto maiores e mais numerosos forem os defeitos encontrados pelo usuário, mais tempo será necessário para a correção e redistribuição do sistema. Assim, quanto maior for a deterioração do *software*, menor é a qualidade observada pelo usuário (TAJIMA e MATSUBARA, 1981).

Além destas métricas para processos e produtos, existem também as métricas que procuram levantar informações a respeito dos recursos utilizados no projeto. Como o componente humano é o principal fator em um esforço de desenvolvimento, algumas

métricas foram propostas para melhorar a sua caracterização, dentre as quais vale comentar:

- **Experiência:** acredita-se que a experiência da equipe é um importante dado que pode ser utilizado para explicar a qualidade do produto final e a produtividade da equipe durante o desenvolvimento. No entanto, é difícil medi-la de maneira significativa. Qualquer que seja a forma de medição definida, ela deverá permitir a análise da experiência de cada membro e de toda a equipe para questões como ferramentas, métodos, linguagens, etc. Para isso, pode ser definida uma métrica ordinal simples para cada um destes itens, derivada a partir de uma classificação objetiva e ortogonal, como, por exemplo: 0 – nenhuma experiência; 1 – familiaridade, mas nenhuma experiência prática; 2 – experiência em um projeto; 3 – experiência em vários projetos; e 4 – altamente experiente. Como se trata de uma escala ordinal, pode-se definir uma métrica válida para a experiência de toda a equipe como sendo a mediana da experiência de cada um de seus membros (FENTON e PFLEEGER, 1997).
- **Rotatividade:** Por último, um outro fator que também pode ter grande influência na produtividade de uma equipe é a sua rotatividade. Sabe-se que quando uma pessoa sai de um projeto, parte do seu conhecimento se perde ou é passado de alguma maneira para uma outra pessoa. De qualquer forma, isto pode causar a diminuição do desempenho geral da equipe. Além disso, acrescentar mais mão-de-obra, de maneira imprópria, a um projeto atrasado, torna-o mais atrasado ainda (BROOKS, 1975). Isto ocorre, pois esta pessoa precisa conhecer o problema a ser resolvido e o seu domínio. Assim, até que consiga realmente produzir, muito esforço terá sido empregado em atividades não relacionadas ao desenvolvimento propriamente dito. Além disso, algum dos antigos membros do projeto deverá despender parte do seu tempo lhe passando informações. Assim, a produtividade da equipe é sensivelmente influenciada por sua rotatividade a qual pode ser medida relacionando o número de pessoas que entraram ou saíram durante o projeto com o número total de pessoas que passaram pelo projeto (FENTON e PFLEEGER, 1997).

2.4 Considerações Finais

O desenvolvimento de *software*, como disciplina, já existe a mais de quatro décadas, porém, até este momento, não foi possível transformá-la efetivamente em uma disciplina de engenharia. O recente foco no processo de desenvolvimento se mostrou um grande passo nesta direção, mas somente quando este for perfeitamente compreendido que será possível produzir *software* de qualidade controlada dentro do cronograma e orçamentos previstos (ZAHRAN, 1998).

Neste sentido, as métricas de *software* desempenham um importante papel permitindo que, a partir da análise de dados numéricos, conclusões válidas possam ser geradas a respeito do comportamento das entidades do mundo real. Desta forma, podem ser utilizadas para entender melhor as características dos processos, produtos e recursos, prever comportamentos futuros, comparar resultados entre outros. Somente com o uso de dados quantitativos que gerentes poderão tomar decisões vitais para o sucesso de um projeto de forma consciente e correta.

No entanto, como uma disciplina de engenharia relativamente recente, muito trabalho ainda precisa ser feito para que seja possível o uso de métricas de maneira fácil, eficiente e eficaz. FENTON e NEIL (2000) defendem a idéia de que os programas de medição não utilizam todos os dados necessários para apoiar o procedimento de tomada de decisão. Além disso, estes modelos não levam em consideração as relações de causa e efeito existente entre as métricas necessárias para a sua correta interpretação.

Assim, novas metodologias, abordagens e ferramentas devem ser desenvolvidas para apoiar o uso de medições de forma a dar mais um passo no sentido de transformar o desenvolvimento de *software* em uma verdadeira engenharia.

Capítulo 3
A Estação TABA

Neste capítulo, é apresentada a Estação TABA, projeto no qual este trabalho está inserido. São discutidos suas motivações, sua estrutura e os principais trabalhos já desenvolvidos, com destaque para a definição de processos e instanciação de ambientes de desenvolvimento de software.

3.1 Introdução

A Estação TABA (ROCHA *et al.*, 1990) é um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de *software* (ADSs) adequados às particularidades de processos de desenvolvimento e de projetos específicos. ROCHA *et al.* (1990) definem meta-ambiente como um ambiente que abriga um conjunto de programas que interage com usuários para definir interfaces, selecionar ferramentas e estabelecer os tipos de objetos que irão compor o ambiente de desenvolvimento específico.

O projeto TABA foi criado no início da década de 90, a partir da constatação de que domínios de aplicação diferentes possuem características distintas, e que estas devem incidir nos ambientes de desenvolvimento através dos quais os engenheiros de *software* desenvolvem aplicações. Desta forma, a Estação TABA tem por objetivo auxiliar na definição, implementação e execução de ADS adequados a contextos específicos.

Com o intuito de atender a este objetivo, quatro funções foram definidas para a Estação TABA:

- Auxiliar o engenheiro de *software* na especificação e instanciação do ambiente mais adequado ao desenvolvimento de um produto específico a partir do processo de *software* e/ou de uma definição do domínio de aplicação;
- Auxiliar o engenheiro de *software* na implementação das ferramentas necessárias ao ambiente definido;
- Permitir aos desenvolvedores do produto de *software* a utilização da estação através do ambiente instanciado;

- Permitir a execução do *software* na estação configurada para o seu desenvolvimento.

Estas funções determinam quatro ambientes na Estação TABA

- **Ambiente Especificador e Instanciador de ADSs:** é o próprio meta-ambiente TABA. Sua principal função é especificar o ADS mais adequado para o desenvolvimento de um produto de *software*, em um determinado domínio de aplicação, e instanciar este ADS;
- **Ambiente para Construção de Ferramentas:** este ambiente auxilia o engenheiro de *software* na construção de novas ferramentas para a Estação TABA e sua incorporação ao meta-ambiente;
- **Ambiente de Desenvolvimento:** é o ADS que foi especificado e instanciado através do meta-ambiente;
- **Ambiente de Execução:** é o local onde o produto de *software* poderá ser executado.

Considerando que as particularidades de domínios de aplicação também devem ser consideradas na instanciação de um ADS para que este ofereça ferramentas mais específicas e possibilite a reutilização do conhecimento do domínio, OLIVEIRA (1999) estendeu a definição da Estação TABA de forma que esta possa instanciar ADS orientados a domínios específicos, chamados de Ambientes de Desenvolvimento de *Software* Orientados a Domínio (ADSOD) através da inclusão do conhecimento de domínio para as diversas atividades do desenvolvimento de *software*.

3.2 Estrutura

A sua estrutura foi definida como um conjunto de componentes integrados que possuem controle sobre a sua existência, suas informações, estados e funcionalidades básicas associadas. Esta definição baseou-se na idéia de que um ADS está inserido no contexto de um produto de *software* e é o responsável, com seu conjunto de funcionalidades e informações associadas, por traduzir, de acordo com a necessidade do usuário, uma representação do mundo real para o computacional (TRAVASSOS, 1994).

Para que sejam atingidos os objetivos da Estação TABA, devem estar presentes os seguintes componentes: sistema computacional, interface com o usuário, cooperação, controle de processos, suporte inteligente, reutilização, conhecimento, repositório

comum do ADS e controle de versões. A utilização, em conjunto, destes componentes permite a integração das ferramentas ao ambiente e provê recursos para a incorporação de ferramentas externas.

TRAVASSOS (1994) definiu a filosofia de integração da Estação TABA e dos seus ambientes instanciados, onde considera quatro tipos de integração:

- **Apresentação e Interação:** é responsável por proporcionar ao usuário a sensação de integração no ambiente. É através dela que se torna possível a homogeneização das formas de apresentação das informações e das técnicas de interação do usuário;
- **Gerenciamento e Controle:** é responsável por prover, às ferramentas e ao ambiente, serviços e funcionalidades básicas que permitam o funcionamento de forma organizada, ao longo do processo de desenvolvimento;
- **Dados:** estabelece a forma como as ferramentas da Estação TABA realizarão o tratamento das informações. Provê os serviços básicos de armazenamento e gerenciamento de estruturas de informação obtidas a partir das ferramentas que compõem o ambiente;
- **Conhecimento:** torna possíveis os serviços básicos de armazenamento, gerenciamento e utilização do conhecimento descrito e adquirido ao longo do processo de desenvolvimento. O conhecimento compartilhado pelas ferramentas descreve o significado das informações construídas ao longo do processo de desenvolvimento e permite a descrição da sintaxe apropriada para a apresentação destas informações por parte das ferramentas.

A integração do conhecimento tem sido constante objeto de estudo no contexto da Estação TABA, sofrendo bastantes modificações desde sua concepção até o momento atual. Nesse sentido, FALBO (1998) propôs a utilização de Servidores do Conhecimento no qual o conhecimento é modelado para reuso e disponibilizado em um conjunto de componentes que podem ser usados por várias ferramentas a ser desenvolvidas. Esses componentes são construídos a partir de ontologias que descrevem o conhecimento do domínio de interesse e de modelos de tarefas que descrevem o conhecimento genérico de tarefas aplicáveis a vários domínios. O objetivo dos Servidores de Conhecimento é prover esta infra-estrutura comum, para o desenvolvimento de um conjunto de sistemas baseados em conhecimento em um universo de discurso (FALBO, 1998).

No que se refere à tecnologia de representação utilizada para construção desses componentes, a Estação TABA provê uma classe de representação de conhecimento que permite a definição de diferentes tecnologias de representação: rede neural e sistemas lógicos, englobando *frames*, sistema de programação em lógica e rede semântica. Atualmente, tem-se implementada a classe de sistema de programação em lógica baseada no uso de uma máquina Prolog externa incorporada à Estação TABA através de uma biblioteca de classes disponível para a linguagem de programação utilizada[1].

3.3 O Processo de Desenvolvimento de *Software* na Estação TABA

A construção de um ADS a ser instanciado pela Estação TABA é realizada a partir da definição do processo de desenvolvimento que será utilizado na execução de um projeto específico para o qual este ADS está sendo definido. No contexto do projeto TABA, um processo de desenvolvimento se caracteriza pela descrição de uma seqüência de atividades que utilizam recursos para gerar produtos de software. Desta forma, os principais conceitos relacionados à modelagem de processos de desenvolvimento foram definidos da seguinte forma (ARAÚJO, 1998):

- **Atividades:** são as tarefas ou trabalhos a serem realizados. Uma atividade requer recursos, pode consumir ou produzir artefatos e adotar um procedimento para sua realização. Pode, ainda, ser decomposta em outras atividades o que define três tipo: (i) uma atividade que não compõe nenhuma outra é chamada de Macro-atividade; (ii) aquela que não pode ser mais decomposta é chamada de Atividade Elementar; e as demais que podem ser decompostas e ao mesmo tempo compõem outras atividades são chamadas de Atividades Intermediárias. Além disso, em qualquer nível, podem depender da finalização de outras atividades, denominadas pré-atividades;
- **Artefatos:** são produtos de *software* produzidos ou consumidos por atividades durante a sua realização. São exemplos de artefatos: manuais de qualidade, relatórios de reuniões de revisão, diagramas de fluxo de dados, código fonte, etc. Um artefato pode ser decomposto em outros artefatos.
- **Procedimentos:** são condutas bem estabelecidas e ordenadas para a realização de atividades. Alguns procedimentos podem ser parcialmente automatizados

[1] A versão atual da Estação TABA está implementada na linguagem C++ e o ambiente de desenvolvimento que está sendo utilizado é o Microsoft Visual C++. OLIVEIRA (1999) apresenta maiores detalhes sobre a implementação da versão atual e da anterior.

por ferramentas de *software*. São exemplos de procedimentos: métodos de construção de sistemas, tal como método de Booch para o desenvolvimento orientado a objetos, técnicas de avaliação de qualidade como o método Rocha, métodos de estimativa de projeto como Ponto Função ou COCOMO, etc.

- **Recursos:** são as pessoas, as ferramentas de *software*, os equipamentos ou quaisquer outros recursos necessários à execução de uma atividade.
- **Processos:** são coleções de atividades relacionadas que têm lugar durante o desenvolvimento de um produto de software.

Baseado neste modelo, vários trabalhos foram propostos, no contexto do projeto TABA, com o intuito de apoiar os procedimentos de definição e execução de processos dentre os quais destacam-se a máquina de processo proposta por ARAÚJO (1998) e implementada na primeira versão do TABA e a integração do conhecimento permitindo assistência inteligente na definição de processo, proposta por FALBO (1998) na primeira versão deste meta-ambiente e reimplementada em sua versão atual[2].

A máquina de processo definida por ARAÚJO (1998) estabeleceu uma estrutura para definição do processo de forma a permitir a sua flexibilização, apoiando adaptações a serem realizadas durante o decorrer do desenvolvimento. A infra-estrutura criada para apoiar o processo de desenvolvimento nos ambientes instanciados permitia que o acompanhamento do processo fosse realizado durante todo o esforço do projeto. Além disso, quando se tornava necessário modificar o processo de forma a atender a novas necessidades, esta estrutura permitia que as alterações não afetassem a vida passada da execução do processo evitando que problemas e inconsistências pudessem aparecer. Desta forma, mudanças eram permitidas somente a partir do ponto corrente de execução do projeto.

Como o processo é a base para a definição e construção de qualquer ambiente, foi considerado fundamental que o conhecimento sobre processo estivesse disponível para o meta-ambiente, residindo no repositório de conhecimento, de modo a prover suporte às tarefas de definição de processo e instanciação de um ambiente específico. Além disso, é interessante que este conhecimento possa evoluir de forma que a descrição de processo criada possa ser enriquecida pela experiência adquirida ao longo do tempo, ou, até mesmo, que novas descrições possam ser criadas, representando diferentes versões.

[2] Na versão atual da Estação TABA foi reimplementado apenas o núcleo central do meta-ambiente e a infra estrutura de conhecimento conforme descrito em OLIVEIRA (1999b).

Com este objetivo, FALBO (1998) construiu um Servidor de Conhecimento de Processo no qual uma ontologia sobre processo de desenvolvimento de *software* foi elaborada alterando a estrutura da descrição do conhecimento sobre processo para incluir todos os detalhes relevantes identificados na ontologia. Esta descrição de conhecimento é utilizada em um assistente inteligente capaz de auxiliar os engenheiros de *software* na descrição de um processo específico para o ambiente a ser instanciado.

Assim, a ontologia de processo serve como fonte de conhecimento para a descrição de um processo específico que é instanciado pela máquina de processo segundo padrões determinados e, que por sua vez, é acompanhado e adaptado durante a execução por ferramentas específicas para alterações do processo de desenvolvimento feitas de forma a não causar inconsistências no modelo existente.

3.4 Definição de Processos

Como apresentado anteriormente, o processo de desenvolvimento é considerado um dos fatores determinantes para o alcance da qualidade dos produtos de *software* e, desta forma, se tornou importante a criação de uma abordagem mais ampla para a definição de processos no contexto da Estação TABA. Esta nova abordagem deveria considerar os novos modelos e padrões internacionais definidos recentemente e as necessidades e características específicas da organização para a qual o processo deveria ser definido. Além disso, todo o conhecimento e infra-estrutura existente na Estação TABA também deveriam ser levados em consideração.

Com este objetivo, MACHADO (2000), partindo de trabalhos anteriormente realizados por OLIVEIRA (1999) e MAIDANTCHIK (1999), definiu uma abordagem que permite a definição de processos de *software* de acordo com os requisitos estipulados pela Norma ISO/IEC 12207 (1995) e pelos modelos de maturidade (CMM, ISO/IEC 15504) (PAULK *et al.*, 1995; EMAM *et al.*, 1998). Além disso, esta abordagem permite, ainda, que características específicas da organização e do próprio projeto sejam incorporadas à definição final do processo a ser utilizado em um desenvolvimento específico. O modelo proposto divide em três etapas o procedimento de definição como pode ser visto no esquema apresentado na figura 3.1.

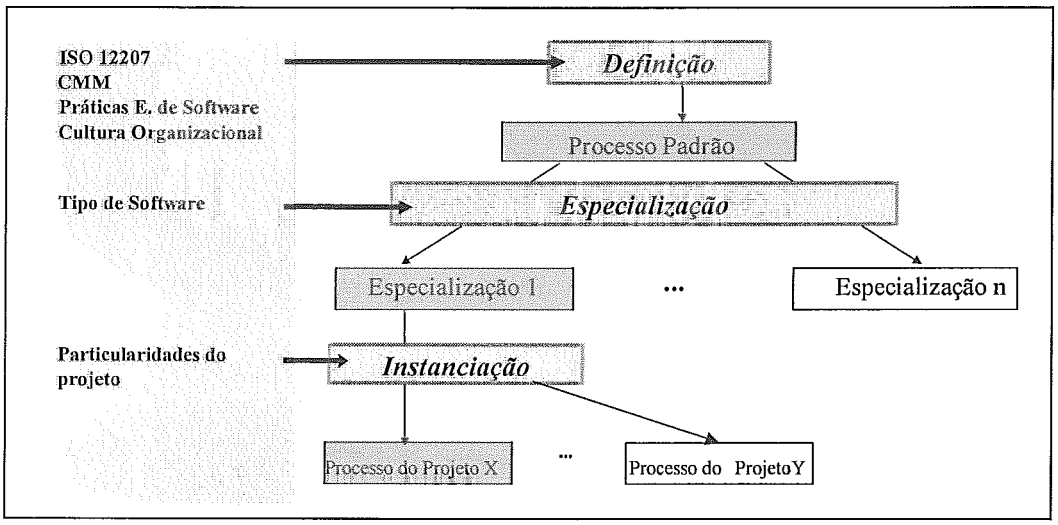


Figura 3.1: Modelo para a Definição de Processos de *Software*.

Em uma organização, diversos projetos podem coexistir possuindo características específicas. Porém, existe um conjunto de elementos fundamentais que se deseja que sejam incorporados em qualquer processo definido. Assim, EMAM *et al.* (1998) definem o conjunto destes elementos fundamentais como o Processo Padrão, ou seja, o processo básico que guia o estabelecimento de um processo comum na organização. Desta forma, um Processo Padrão define uma estrutura única a ser seguida por todas as equipes envolvidas em um projeto de *software*, independentemente das características do *software* a ser desenvolvido.

Na definição do processo, deseja-se que este seja o mais adequado possível às equipes que irão utilizá-lo e, desta forma, na abordagem proposta por MACHADO (2000) a primeira etapa de definição do processo é a definição do Processo Padrão, devendo ser realizada, obrigatoriamente, como base na Norma ISO/IEC 12207 e nas características de desenvolvimento da organização. Considerando que os principais objetivos das organizações produtoras de *software* são definir, utilizar e estabelecer melhorias contínuas nos seus processos, torna-se fundamental que estas direcionem seus esforços não só para a aplicação de métodos e práticas de engenharia de *software*, como, também, passem a considerar características relacionadas ao ambiente de trabalho, ao conhecimento e experiência das equipes envolvidas e à própria cultura e experiência da organização em desenvolver *software*. A definição poderá, ainda, considerar um dos modelos de maturidade associado como o nível de capacitação da organização em

engenharia de *software*, e o tipo de ADS que será instanciado (orientado ou não a domínio).

Definido o Processo Padrão, a segunda etapa consiste em adaptá-lo de forma a atender a diferentes objetivos sendo levados em consideração os seguintes fatores: o tipo de *software* que se deseja desenvolver (por exemplo, sistemas especialistas ou sistemas de informação), o paradigma de desenvolvimento a ser adotado (por exemplo, orientação a objetos), o modelo e o nível de maturidade a ser utilizado e o tipo de ADS que se deseja instanciar (orientado ou não a domínio). Assim, ao se especializar um Processo Padrão, são adicionadas atividades particulares de acordo com o contexto para o qual se está realizando a especialização. Ao final desta etapa, obtém-se um processo aderente ao tipo de *software* e ao paradigma de desenvolvimento a ser adotado. A especialização, considerando um modelo de maturidade e um tipo de ADS, também poderá ser realizada, desde que estas características não tenham sido incluídas na primeira etapa deste procedimento.

Finalmente, a última etapa, a instanciação do processo, consiste em incorporar ao processo especializado características do projeto para o qual se está definindo o processo, devendo ser considerado: as características do projeto e da equipe, o modelo de ciclo de vida, as características do produto e os métodos, ferramentas e recursos a serem utilizados. Somente neste ponto, o processo tem sua definição finalizada, podendo, assim, ser utilizado em um projeto específico.

3.4.1 Ferramentas para a Definição de Processos

Atualmente, o meta-ambiente TABA possui três ferramentas capazes de apoiar a tarefa de definição do processo. A primeira, e mais simples, é a ferramenta Edit-Pro (ZLOT e SANTOS, 1999) a qual permite que esta definição seja realizada a partir do conhecimento existente na Estação TABA. Através do Edit-Pro, o usuário define o processo selecionando as atividades do conhecimento que o compõem e organizando a seqüência de execução que deve ser seguida. Depois disto, define se as atividades serão seqüenciais ou iterativas e, neste caso, o número de repetições que deve ser considerado. Finalmente, são definidos os recursos necessários para a execução de cada atividade e os produtos gerados em cada uma delas. No entanto, esta ferramenta não provê nenhum tipo de orientação para os engenheiros de *software* ficando sob sua responsabilidade a tarefa de concepção do processo.

Em outro trabalho, FALBO (1998) define um assistente inteligente capaz de auxiliar os engenheiros de *software* na descrição de um processo específico para o ambiente a ser instanciado. Esta ferramenta, o Assist-Pro, contém o conhecimento sobre vários modelos de ciclo de vida existentes e, a partir de informações obtidas com o engenheiro de *software*, procura identificar as principais características do projeto a ser desenvolvido para, então, sugerir qual ou quais modelos de ciclo de vida melhor se adequam ao projeto. Além disso, dependendo das características do problema a ser resolvido, da equipe de desenvolvimento e da tecnologia e paradigma selecionados, o sistema automaticamente adiciona ao processo atividades consideradas necessárias segundo estas características, como, por exemplo, atividades de treinamento, gerência ou teste. A partir disto, o engenheiro de *software* pode alterar o processo de desenvolvimento de forma que este atenda a necessidades específicas do projeto a ser apoiado.

Finalmente, por último, MACHADO (2000) desenvolveu o Def-Pro, uma ferramenta mais completa na qual a definição de um processo de *software* leva em consideração todos os fatores definidos em sua abordagem (apresentada anteriormente na seção 3.4), desde a definição do Processo Padrão passando por sua especialização e seguindo até o nível do processo instanciado para um projeto particular. Esta ferramenta foi concebida para integrar-se à Estação TABA e, assim, possibilitar o uso do conhecimento disponível no meta-ambiente. Ela propõe as seguintes formas de utilização: definir o processo padrão da organização, especializar e instanciar o processo padrão para diversos projetos, definir um processo de *software* para projetos específicos, armazenar um processo padrão já existente na organização, especializá-lo e instanciá-lo para diferentes projetos, redefinir o processo padrão à medida que ocorram melhorias na capacitação da organização e, finalmente, reutilizar processos de *software*. O engenheiro de *software* pode decidir entre definir o processo padrão utilizando um modelo de maturidade ou apenas a Norma ISO/IEC 12207. Além disso, o processo poderá ou não ser orientado a domínio o que implica na necessidade da inclusão de atividades de investigação do domínio no processo definido conforme proposto por OLIVEIRA (1999).

Como resultado do uso destas ferramentas é gerado um processo de desenvolvimento capaz de ser utilizado em um esforço de desenvolvimento real. Todos os processos definidos desta forma são cadastrados pela Estação TABA a fim de que estes possam ser utilizados em vários projetos distintos. Cabe ao engenheiro de *software*

definir um novo processo ou selecionar algum cadastrado que seja adequado às características do projeto para que se possa, então, gerar os ambientes instanciados.

3.5 Ambientes de Desenvolvimento de *Software*

Como apresentado anteriormente, o objetivo da Estação TABA é gerar, através de instanciação, ambientes de desenvolvimento de *software* adequados às particularidades de processos e de projetos específicos, considerando ou não um determinado domínio. Com este objetivo, foram incorporadas, ao meta-ambiente, as funcionalidades necessárias para permitir a geração de ADSs a partir da definição do processo de desenvolvimento a ser apoiado.

O primeiro passo a ser executado para a instanciação de um ambiente é a definição de suas características: seu nome, uma breve descrição e o processo que deverá apoiar. Para isso, o engenheiro de *software* poderá selecionar qualquer um dos processos já existentes na Estação TABA. Caso este ainda não tenha sido definido, pode-se utilizar uma das ferramentas apresentadas no item anterior para realizar esta tarefa e, então, relacioná-lo com o ADS que está sendo gerado. Além destes dados, quando se tratar de um ADSOD, deve-se identificar qual o conhecimento do domínio que deve ser incluído.

Neste ponto, as informações sobre o ambiente são inseridas na base de dados da Estação TABA e pode-se passar para a etapa de instanciação propriamente dita, quando o meta-ambiente compila o código fonte do ADS e gera o seu arquivo executável. Para ter acesso a base de dados, é efetuada uma cópia da base da Estação TABA para o ambiente instanciado de forma que este se torne totalmente independente, podendo ser executado tanto a partir da Estação TABA quanto externamente como qualquer outro programa.

Como pode ser visto na figura 3.2, a tela principal de um ADS instanciado é bastante simples e contém as atividades do processo de desenvolvimento e as ferramentas que podem ser utilizadas para a realização de cada uma destas atividades. A interface padrão foi desenvolvida de forma a proporcionar ao engenheiro de *software* uma visão geral de todas as atividades associadas ao processo, assim como a relação de hierarquia e ordem de execução entre elas. Desta forma, toda a interface do ADS se

baseia na definição do processo que está sendo apoiado e qualquer ferramenta inserida neste ambiente deverá estar condizente com esta abordagem.

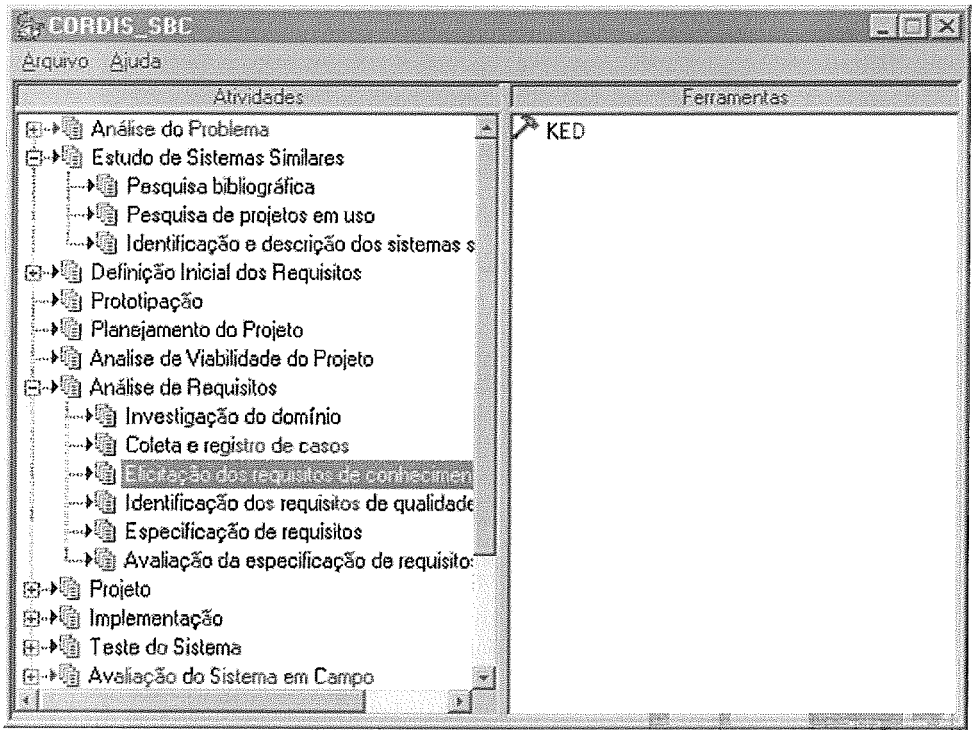


Figura 3.2: Um Ambiente Instanciado pela Estação TABA

O ambiente instanciado ainda permite a visualização de todas as características de cada uma das atividades do processo. Podem ser exibidos os recursos necessários para a sua execução o que inclui recursos humanos e de *hardware*, suas pré e sub-atividades, os artefatos que deverão ser produzidos e os procedimentos que deverão ser utilizados para a execução da atividade. Assim, todos os artefatos de uma macro-atividade deverão ser gerados por algum recurso humano a partir da execução de alguma de suas atividades elementares definidas no processo. Para isso, pode-se fazer uso de ferramentas internas ou externas, procedimentos e métodos a fim de produzir o artefato desejado.

Com todas estas informações disponíveis no ambiente instanciado, é possível realizar pequenas modificações nos ADS para armazenar os dados sobre a execução do processo como, por exemplo, o tempo e o esforço empregados em cada macro-atividade, quais os artefatos produzidos e consumidos entre outras informações. Esta infra-estrutura é a base para a implantação de um sistema de acompanhamento do

processo permitindo que métricas sejam calculadas a partir da composição destes dados básicos coletados durante a execução dos projetos.

3.6 Considerações Finais

No atual estágio em que se encontram os trabalhos, o projeto TABA possui as capacidades de definir processo de desenvolvimento e de instanciar ambientes para apoiar os trabalhos em projetos específicos. Apesar de não se encontrar disponível nenhum tipo de acompanhamento do projeto no contexto dos ambientes instanciados (como, por exemplo, a abordagem proposta por ARAÚJO), estes possuem a infra-estrutura básica necessária para a implantação de sistemas com esta finalidade.

Além disso, a Estação TABA não possui uma outra característica, cada vez mais necessária nos dias de hoje, que é o apoio à melhoria contínua do processo de desenvolvimento utilizado pelas organizações. Embora os ambientes sejam definidos e instanciados considerando um processo, atualmente não há nenhum tipo de apoio para avaliar se este é realmente adequado para a execução do projeto. Definir o Processo Padrão e os processos instanciados para uma empresa não é mais suficiente. Estes devem ser monitorados e avaliados para que problemas e melhorias possam ser detectados de forma que o desempenho e a maturidade dos processos possam ser otimizados.

Nos próximos capítulos deste trabalho, será apresentada uma abordagem para a avaliação de processos de *software*. Com este objetivo, no capítulo 4, será apresentada a abordagem teórica para a avaliação do processo baseada no uso de métricas coletadas durante o decorrer do projeto e no capítulo seguinte será apresentada a implementação de uma ferramenta para apoiar esta abordagem no contexto do projeto TABA.

Capítulo 4

Avaliação de Processos de *Software*

Neste capítulo, é apresentada uma abordagem para a avaliação de processos de software. Esta considera um conjunto de passos desde a seleção e definição das métricas, que são utilizadas na avaliação do processo, até a análise dos resultados obtidos com as medições.

4.1 Introdução

A avaliação de processos de *software* é uma atividade fundamental para o alcance de níveis cada vez mais altos de qualidade para os produtos e de desempenho para os processos e deve estar baseada no uso de métricas. Entretanto, a grande maioria das empresas não possui profissionais qualificados para definir e implantar uma abordagem como esta. Desta forma, para que um programa assim seja efetivo e eficiente é necessário prover os desenvolvedores com o adequado ferramental tecnológico e metodológico (LAVAZZA, 2000).

A partir dos estudos da literatura vigente, foi constatada a importância da formalização de cada etapa que compõe uma abordagem de melhoria de processo de forma a aumentar as chances de se obter êxito. Uma nova proposta deve possuir procedimentos para a definição sistemática do conjunto de métricas que serão utilizadas, deve definir como os dados referentes a cada métrica devem ser obtidos e como deverá ser feita a análise dos resultados. Neste capítulo, será apresentado cada um destes passos da abordagem proposta neste trabalho.

4.2 Considerações e Comprometimentos

Apesar de existir a possibilidade da implantação de métricas independentemente do uso de um processo (DASKALANTONAKIS, 1992), a abordagem proposta neste trabalho não tem por objetivo somente medir, mas, sim, levantar dados sobre as características do processo a fim de poder avaliá-lo. Desta forma, parte-se do pré-suposto que o processo de *software* foi definido conforme proposto por MACHADO

(2000) e adaptado de forma que os procedimentos de medição fossem integrados ao próprio processo produtivo como será apresentado mais à frente neste capítulo. Além disso, deve-se entender que o processo está sendo devidamente seguido pela equipe de desenvolvimento de maneira que os dados obtidos nas medições possam realmente retratar a realidade vivida no decorrer do projeto.

Além destes pré-requisitos, alguns aspectos a respeito do uso de métricas foram considerados relevantes para qualquer nova abordagem e, desta forma, deverão ser utilizados neste trabalho. São eles:

a) Seguindo a abordagem GQM, a seleção de métricas deve ser baseada nos objetivos definidos para o processo de avaliação. Estes objetivos, por sua vez, devem estar relacionados às metas organizacionais que levaram a implantação do programa de melhoria, pois isto facilita a análise dos resultados além de ser um fator de grande influência para a obtenção do apoio da alta gerência;
b) Para obter o comprometimento da equipe de desenvolvimento é importante deixar bem claro que os resultados obtidos serão utilizados na avaliação do processo e não do desempenho do pessoal envolvido no projeto. A definição clara dos objetivos da abordagem e o seu amplo entendimento por parte da equipe podem ser a diferença entre o sucesso e o fracasso de um programa de medição (BAUMERT, 1992);
c) Para que o uso de uma métrica seja considerado viável é importante observar se o custo necessário para sua utilização não supera os benefícios que esta informação pode trazer para a abordagem de melhoria. Sempre que isto ocorrer, a métrica deverá ser excluída, pois sua finalidade básica foi violada; Porém, esta não é uma análise simples de ser feita e, assim, é importante que o conjunto utilizado seja de fácil aquisição na tentativa de viabilizar o seu uso;
d) Outro ponto importante que deve ser observado para a seleção de uma métrica é o nível de maturidade da empresa. Quanto menos maduro é o processo de uma organização, menos ricas (ou maduras) podem ser as métricas utilizadas (FENTON e NEIL, 2000). Como no contexto brasileiro[3], no qual este trabalho

[3] Segundo o relatório de Qualidade e Produtividade no Setor de *Software* brasileiro apenas 3,6% das empresas conhece e utiliza sistematicamente a ISO/IEC 12207, 1,8% conhece e utiliza sistematicamente o CMM e apenas 0,2% das empresas conhece e utiliza sistematicamente o SPICE. Além disso, os métodos de Engenharia de *Software* que efetivamente são utilizados pelas empresas indicam uma baixa maturidade na maioria dos processos existentes, onde o foco principal está na correção de defeitos e não a sua prevenção (PBQP, 1999).

está inserido, o nível de maturidade da grande maioria das empresas é baixo, optou-se por utilizar métricas de fácil utilização de forma a permitir que a abordagem atinja rapidamente um público maior;

e) Para facilitar a coleta dos dados, evitar um aumento significativo de esforço por parte da equipe de desenvolvimento e aumentar a consistência das informações, deve-se incorporar o procedimento de coleta de métricas ao próprio processo produtivo. Desta forma, a equipe de desenvolvimento pode concentrar seus esforços na realização das atividades de produção enquanto as atividades de análise da qualidade podem ser feitas por uma equipe especializada (BASILI *et al.*,1994);

f) Deve-se dar preferência para o uso de métricas objetivas, isto é, métricas que dependam somente do objeto que está sendo medido, pois isto faz com que o ponto de vista da pessoa que coleta os dados não interfira no resultado da medição;

g) Sempre que possível, o conjunto de métricas deve ser aplicado às várias entidades que compõem um projeto: processo, produtos e recursos. Desta forma, dados referentes aos vários aspectos do projeto serão levados em conta na avaliação do processo;

h) O contexto no qual os dados estão inseridos é fundamental para a avaliação do processo. Quando o contexto não está explicito, este pode não ser visualizado por quem estiver realizando a tarefa de análise das métricas, o que pode levar a conclusões equivocadas. Nenhuma contagem ou medição tem significado separada de seu contexto (PARK *et al.*,1996);

i) E, finalmente, o que pode ser considerado a base de toda a abordagem proposta, deve-se, de alguma forma, levar em consideração o relacionamento existente entre todos os dados coletados. Acredita-se que conclusões podem ser inferidas sobre as deficiências do processo a partir do contexto em que o projeto está inserido e a partir dos relacionamentos existentes entre as métricas coletadas.

Uma abordagem que consiga atender a todos estes requisitos definidos como práticas recomendadas para a implantação de um programa de medição tem maiores chances de obter êxito.

4.3 Seleção e Definição das Métricas para Avaliação do Processo

O primeiro passo para a implantação de um programa para a avaliação de processos de *software* baseado em medição é a seleção do conjunto de métricas que deverá ser utilizado na análise do processo. Não é objeto deste trabalho a definição de novas métricas para a avaliação de processos. Baseado nesta decisão, foi decidido que todas as métricas utilizadas nesta abordagem seriam extraídas ou adaptadas da literatura.

O foco das atenções deste trabalho está na seleção de métricas já existentes que, de alguma forma, possam prover informações a respeito do processo utilizado, dos produtos gerados e dos recursos consumidos e, de acordo com as métricas selecionadas, estabelecer uma estrutura de análise de seus resultados de forma condizente com sua definição. Assim, é importante ressaltar que não se pretende que este seja o melhor conjunto de métricas para a avaliação de um processo de *software*, mesmo porque, como discutido, não é este o objetivo. Este é apenas um conjunto válido com o qual será possível apresentar as principais idéias da abordagem proposta.

Seguindo a abordagem GQM (BASILI *et al.*,1994), foram inicialmente estabelecidos os objetivos a serem alcançados no programa de avaliação proposto neste trabalho. Para estabelecer estes objetivos, especialistas nas áreas de gerência de projetos, qualidade de *software*, processo e melhoria de processo foram consultados a respeito dos principais problemas comumente enfrentados pelas empresas desenvolvedoras de *software* no contexto brasileiro. Nas entrevistas realizadas, vários foram os problemas citados, porém, três se destacaram como sendo os mais relevantes para iniciar uma abordagem de melhoria de processo baseada em medições, analisado de acordo com o ponto de vista dos gerentes de projetos. São eles:

i) A falta de precisão das estimativas de projeto;

ii) A baixa qualidade observada nos produtos liberados para uso;

iii) O alto custo envolvido no desenvolvimento de *software*.

Segundo os especialistas, a solução ou o melhor entendimento dos fatores que interferem nestes três itens aumentaria consideravelmente a qualidade e o desempenho das empresas que desenvolvem *software*. Desta forma, atendendo estes pontos estaríamos também atendendo ao comprometimento a) estabelecido anteriormente que descreve a importância de basear a seleção de métricas nas metas da organização.

A partir da identificação destes problemas, foi definido que a abordagem proposta neste trabalho teria como principais objetivos:

i) Melhorar a precisão das estimativas de projeto;

ii) Aumentar a qualidade dos produtos liberados para uso;

iii) Diminuir o custo final dos projetos.

Uma vez definidos os objetivos do programa de medição, estes devem ser utilizados para orientar a definição de questões a serem respondidas a fim de analisar se as metas estipuladas foram devidamente alcançadas. A tarefa agora é identificar o que é necessário saber para conhecer melhor as atividades relacionadas com o alcance dos objetivos propostos (PARK *et al.*, 1996).

Estas questões devem permitir uma análise quantitativa dos atributos pertinentes a cada objetivo da avaliação de forma que se tornem um elo entre as metas subjetivas e os valores quantitativos levantados durante o desenvolvimento. São atributos pertinentes aqueles que, se quantificados, ajudam a responder as perguntas ou estabelecer o contexto para interpretar as repostas (PARK *et al.*, 1996).

Foram, então, selecionadas algumas questões e suas respectivas métricas, como será apresentado a seguir.

Objetivo 1: Melhorar a Precisão das Estimativas de Projeto

Para entender melhor o primeiro objetivo definido pelos especialistas, foram definidas duas perguntas para melhor caracterizar o objeto em questão: "qual a precisão da estimativa de cronograma?" e "qual a precisão da estimativa de esforço?". Com estas perguntas, é possível estudar os dois principais aspectos de uma estimativa de desenvolvimento e, desta forma, pode-se avaliar se o projeto atende à primeira meta estipulada.

A partir da definição destas questões, foram definidas métricas que comparam os dados estimados para o projeto com os dados reais medidos durante o decorrer dos trabalhos. Foram selecionadas duas métricas para cada uma das questões de forma a considerar tanto os valores globais quanto os valores detalhados por macro-atividade, como pode ser visto na Figura 4.1.

Para facilitar a coleta, o entendimento e a análise dos dados, cada métrica selecionada para um objetivo deverá ser claramente definida de forma a atender os comprometimentos firmados no início deste capítulo. No anexo 1, podem ser vistas todas as métricas utilizadas nesta abordagem com suas respectivas definições e formas de cálculo.

Objetivo 1 :

- Propósito: Melhorar
- Questão: precisão
- Objeto: estimativas de projeto
- Ponto de Vista: analisado pelo ponto de vista dos gerentes de projeto

Questão 1.1 :

Qual a precisão das estimativas de cronograma do projeto?

Métrica 1.1 a) Precisão da Estimativa de Cronograma = $\frac{\text{Tempo real de todo o projeto}}{\text{Tempo estimado para o projeto}}$

Métrica 1.1 b) Pr. Est. Cron. por Macro-Atividade = $\frac{\text{Tempo real de cada macro-atividade}}{\text{Tempo estimado para a macro-atividade}}$

Questão 1.2 :

Qual a precisão das estimativas de esforço do projeto?

Métrica 1.2 a) Precisão da Estimativa de Esforço = $\frac{\text{Esforço real de todo o projeto}}{\text{Esforço estimado para o projeto}}$

Métrica 1.2 b) Pr. Est. Esforço por Macro-Atividade = $\frac{\text{Esforço real por macro-atividade}}{\text{Esforço estimado para a macro-ativ}}$

Figura 4.1: Definição do primeiro objetivo da abordagem de avaliação do processo segundo a abordagem GQM.

Seguindo a orientação de (CARLETON *et al.*, 1992), foi definido que para medir "tempo" em um projeto seria utilizado o número de dias decorridos entre a data de início dos trabalhos (de uma macro-atividade ou de todo o projeto) e a data na qual todas as atividades desta etapa são consideradas como encerradas. Ainda segundo o mesmo trabalho, foi definido que a métrica de "esforço" seria medida em homens-hora sendo que um homem-hora é o tempo de uma hora empregado por um membro da equipe do projeto (IEEE P1045/D5.0, 1992). Assim, o esforço deve ser o somatório do número total de horas empregadas por todos os membros da equipe na execução de uma atividade (macro-atividade ou todo o projeto).

Definidos desta forma, medidas de tempo e esforço podem ser facilmente obtidos, sendo necessário, para isso, o armazenamento das datas, das atividades executadas por cada membro da equipe e do intervalo de tempo empregado na a sua realização.

Objetivo 2: Aumentar a Qualidade dos Produtos de *Software*

Uma análise similar foi feita para o segundo objetivo e novamente foram definidas duas questões a serem respondidas: "qual a qualidade dos produtos antes da

sua liberação para uso?" e "qual a qualidade dos produtos após a sua liberação para uso?". Entretanto, medir a qualidade de um produto de *software* não é uma tarefa tão fácil e direta como no caso anterior devido ao grande número de componentes que podem interferir direta ou indiretamente na qualidade final do produto. A própria definição de qualidade é bastante vaga o que dificulta a definição das métricas para o seu estudo. Não se pode esquecer, ainda, dos comprometimentos feitos de forma a não utilizar métricas de difícil coleta que possam atrapalhar a sua direta aplicação em empresas de baixa maturidade. Na Figura 2, podem ser vistas as questões definidas para a análise deste segundo objetivo com suas respectivas métricas.

Objetivo 2 :

- Propósito: Aumentar
- Questão: qualidade
- Objeto: produtos liberados para uso
- Ponto de Vista: analisado pelo ponto de vista dos gerentes de projeto

Questão 2.1 :

Qual a qualidade dos produtos antes da sua liberação para uso?

Métrica 2.1 a) Densidade de Defeitos = $\frac{\text{Número de erros e modificações}}{\text{Tamanho do sistema}}$

Questão 2.2 :

Qual a qualidade dos produtos após sua liberação para uso?

Métrica 2.2 a) Deterioração do *Software* = $\frac{\text{Esforço após a liberação para uso}}{\text{Esforço antes da liberação}}$

Figura 4.2: Definição do segundo objetivo da abordagem de avaliação do processo segundo a abordagem GQM.

Visando melhor estudar a qualidade dos produtos antes da sua liberação para uso, a densidade de defeitos no sistema será utilizada como o fator de análise para a primeira questão levantada. Defeitos introduzidos em um produto de *software* podem gerar uma grande variedade de efeitos. Desta forma, é necessário esforço e capacitação para detectar, remover, reparar e re-testar os produtos (FLORAC *et al.*, 1997). Porém, acredita-se que, à medida que o número de defeitos encontrados aumenta, cresce também a possibilidade de novos problemas aparecerem devido a correções mal feitas e ao possível aumento da complexidade da implementação do *software*. Desta forma, chegou-se à conclusão de que o número de erros e modificações encontrados durante o desenvolvimento, padronizados pelo tamanho do produto, é uma medida que pode

prover indícios a respeito da qualidade do *software* antes da sua liberação para o usuário. Além disso, esta é uma das poucas métricas objetivas para analisar a qualidade dos produtos de *software* o que condiz com o comprometimento f), que sugere dar preferências a este tipo de métrica e evitar, quando possível, o uso das subjetivas.

Foi estabelecido que "erro" é qualquer problema novo encontrado em um documento no momento da sua aprovação. Para que seja possível coletar estes dados, toda as reuniões de revisão deverão gerar um relatório contendo uma listagem dos erros encontrados.

Porém, quando um novo problema é encontrado em um artefato que já tenha sido aprovado anteriormente, é necessário modificá-lo para que este atenda corretamente aos requisitos estipulados. Sempre que isto for necessário, deve-se documentar as modificações realizadas a fim de tornar possível sua posterior contagem de forma semelhante àquela feita com os erros.

Assim, a diferença básica entre erro e modificação é que contamos um erro quando estamos aprovando um documento e este contém problemas, mas contamos uma modificação quando é necessário corrigir um problema encontrado em um documento já aprovado em atividades anteriores.

Para finalizar a definição da métrica de densidade de defeitos, o tamanho do produto deve ser medido em número de linhas de código, excluindo linhas em branco e linhas contendo apenas comentários, como sugerido por (CARLETON *et al.*, 1992). Para garantir a consistência entre diferentes medições e evitar que divergências apareçam devido a formas distintas de contagem, deve-se construir uma ferramenta simples capaz de contar o número de linhas de código produzidas. É importante observar que para padronizar o número de defeitos encontrados durante os projetos, outras métricas de tamanho podem ser utilizadas, como é o caso, por exemplo, de pontos por função. Para isto, basta que a maturidade da empresa permita sua fácil implantação. Como dito no início deste capítulo, a definição absoluta da métrica não se torna relevante desde que aplicada de forma consistente e analisada de forma coerente à sua definição (GRADY e CASWELL, 1987).

Sabe-se que o número de defeitos encontrados em um *software* antes da sua liberação para uso não indica necessariamente uma baixa qualidade do produto. Um número elevado de defeitos pode indicar exatamente o contrário, isto é, pode indicar que a técnica utilizada para a detecção de defeitos é muito eficiente e, com isso, o *software* possui uma alta qualidade e confiabilidade. Assim, para avaliar corretamente a

qualidade dos produtos gerados, é necessário compreender o comportamento do produto após a sua liberação para uso. Somente com uma análise conjunta destas informações que uma correta inferência pode ser realizada a respeito da qualidade dos produtos gerados.

Além dos problemas internos que estão sendo utilizados para responder a primeira questão, defeitos não encontrados durante a produção do *software* e que são detectados somente pelo usuário final estão sendo levantados para analisar sua satisfação com o produto final. Determinar o que realmente representa qualidade do ponto de vista dos consumidores (usuários) é uma tarefa bastante complicada. Porém, é bastante claro que o número de problemas associados ao *software* varia inversamente à percepção de qualidade (FLORAC *et al.*, 1997). Seguindo a abordagem proposta por (TAJIMA e MATSUBARA, 1981), a segunda questão levantada objetiva analisar a qualidade dos produtos liberados para uso. Isto está sendo feito através da comparação do esforço necessário para a realização de correções de problemas encontrados após a liberação com o esforço total empregado antes da liberação. Com o uso desta métrica, vários fatores são implicitamente considerados na contagem do esforço empregado com a correção de defeitos após a liberação para uso. Somente a título de exemplo, podem ser mencionadas a quantidade de modificações realizadas e a complexidade de cada uma delas.

Objetivo 3: Diminuir o Custo Final dos Projetos

Assim como no objetivo anterior, analisar genericamente os fatores que compõem o custo final de um projeto de *software* não é uma tarefa simples e, desta forma, foi decidido que somente o componente humano seria analisado visto que, na grande maioria dos casos, este é o fator determinante para o custo final dos projetos. Assim, este é o ponto a ser analisado neste terceiro objetivo.

Como pode ser visto na Figura 4.3, foram estabelecidas duas questões de forma a definir melhor o objeto de estudo. A primeira pergunta procura analisar o componente final do esforço empregado pela equipe, tanto para o projeto como um todo como para cada uma das macro-atividades que o compõem. Desta forma, pode-se visualizar qual a ordem de grandeza do esforço do projeto e analisar qual macro-atividade ou quais macro-atividades possuem o maior impacto no valor global. Estas são informações importantes que nos permitem direcionar os trabalhos de redução de custo e comparar

projetos de mesma ordem de grandeza a fim de entender melhor os fatores que influenciam o valor final.

Objetivo 3 :

- Propósito: Diminuir
- Questão: custo final
- Objeto: dos projetos
- Ponto de Vista: analisado pelo ponto de vista dos gerentes de projeto

Questão 3.1 :

Qual é o esforço do projeto?

Métrica 3.1 a) Esforço total do projeto.

Métrica 3.1 b) Esforço por macro-atividades do projeto.

Questão 3.2 :

Qual o percentual de retrabalho em relação ao esforço do projeto?

Métrica 3.2 a) Percentual de Retrabalho Total = $\frac{\text{Esforço em retrabalho} * 100}{\text{Esforço total do projeto}}$

Métrica 3.2 b) P. Retrab. Macro-Ativ. = $\frac{\text{Esforço em retrabalho na macro-ativ} * 100}{\text{Esforço total na macro-atividade}}$

Figura 4.3: Definição do último objetivo da abordagem de avaliação do processo segundo a abordagem GQM.

Porém, sozinha, esta métrica não possui informações suficientes para alcançar o propósito aqui desejado que é o de diminuir o custo final dos projetos ou, neste caso, diminuir o esforço total empregado. De acordo com os especialistas, um fator chave que contribui para o aumento desnecessário do valor final é o esforço empregado com análise, correção, testes e implantação ocorridos devido a problemas detectados durante e após o desenvolvimento. O retrabalho gerado, quando um problema encontrado precisa ser corrigido, deve ser devidamente analisado, pois esta é uma poderosa fonte de redução de custo dos projetos. Assim, o esforço em retrabalho empregado durante o projeto está sendo comparado com o valor do esforço total, tanto para o valor global como para os valores detalhados por macro-atividade. Com este tipo de informação, pode-se direcionar os trabalhos de redução de custo às atividades que possuem um nível de retrabalho considerado acima do normal.

Retrabalho foi definido como qualquer atividade de correção, modificação ou melhoria realizada sobre um artefato de *software* que já deveria ter sido finalizado. Assim, não é considerado retrabalho, por exemplo, os problemas encontrados por um programador durante o desenvolvimento de um módulo ou função, pois este ainda não

foi validado e considerado finalizado. Somente após a realização de uma atividade de avaliação do documento, feita, de preferência, por outra pessoa que não o próprio desenvolvedor, que qualquer atividade realizada para sua correção, modificação ou melhoria será considerada retrabalho.

4.4 Procedimento para a Construção da Estrutura de Decisão

Pode-se observar, através da análise dos objetivos definidos, que nem todas as informações necessárias para a melhoria do processo podem ser obtidas aplicando somente estas medidas. Elas apenas ajudam a determinar se as metas estabelecidas foram realmente alcançadas, porém, quando isto não acontece, pouca informação é fornecida a respeito da origem dos desvios.

Segundo FENTON e NEIL (2000), este problema ocorre devido aos modelos atuais não conseguirem fornecer todas as informações necessárias para a correta análise e tomada de decisão. O uso de métricas tem sido dominado por modelos estatísticos, enquanto o que é realmente necessário é um modelo causal, pois este provê uma estrutura lógica para explicar eventos que, então, podem ser quantificados.

Seguindo esta abordagem, foi definida uma estrutura capaz de encontrar, a partir dos resultados obtidos nas medições, quais as possíveis causas para os desvios ocorridos quando os objetivos estipulados não forem alcançados. Sempre que um valor não aceitável para o resultado de uma métrica for obtido, alguma análise pode ser feita a respeito das características do processo utilizado.

Para definir o que é um valor não aceitável para o resultado de uma métrica, algumas técnicas foram analisadas e o uso de valores de tolerância definidos pelo usuário do sistema foi selecionado devido à sua simplicidade e ao seu poder. Esta é uma alternativa interessante, pois valores mais próximos da realidade da empresa ou organização podem ser utilizados e parâmetros cada vez menores permitem um refinamento contínuo do procedimento de avaliação do processo. Sempre que possível, é sugerido que uma primeira execução do processo em um projeto seja medido para avaliar qual o nível em que se encontram estes valores de forma a facilitar sua definição. Tolerâncias obtidas na literatura também podem ser de grande ajuda na definição destes valores.

Para a definição da estrutura, foram analisados os objetivos, as perguntas e as métricas definidas através do uso do GQM e observou-se que as perguntas podem ser

vistas como sub-objetivos do objetivo principal e as métricas constituem a forma como estes sub-objetivos devem ser avaliados. Ou seja, através das métricas, pode-se definir se uma questão é ou não verdadeira (isto é, se um sub-objetivo foi ou não alcançado) e, a partir do resultado destas questões, pode-se determinar se o objetivo principal foi ou não atingido. Estendendo esta análise aos sub-objetivos, cada um pode ser visto como um objetivo que pode ser explicado por novos sub-objetivos repetindo o mesmo procedimento inicial realizado seguindo o GQM. Desta forma, pode-se detalhar objetivos em sub-objetivos e estes em novos sub-objetivos até que não seja mais possível ou interessante realizar esta decomposição.

Baseado nesta análise, buscou-se, novamente, a ajuda de especialistas para tornar possível a construção da estrutura de avaliação do processo e foi estabelecido o seguinte procedimento:

I- Com um objetivo selecionado, deve-se questionar os especialistas a respeito de quais os possíveis problemas diretos que a equipe de desenvolvimento pode ter enfrentado para não ser possível alcançar este objetivo. Neste passo, o objetivo é dividido nos sub-objetivos diretos que o compõem. Quando não for possível ou interessante detalhá-lo melhor, marque este como um nó terminal da estrutura e passe para o passo seguinte;

II- A partir disto, os especialistas devem informar o que pode ser considerado um problema para este objetivo levando em consideração o uso ou a definição de uma métrica para sua análise e/ou o uso de sub-objetivos. Ou seja, deve ser definido como decidir se o objetivo selecionado foi ou não alcançado. Deve-se definir uma única métrica que possa descrever o objetivo selecionado de forma mais precisa e, assim, permitir a sua análise. Isto nem sempre será possível, pois alguns objetivos são, na verdade, um conjunto de fatores e, somente quando detalhados, poderão ser analisados por métricas específicas;

III- Enquanto existir sub-objetivo não analisado, reiniciar o processo tomando como objetivo um destes sub-objetivos.

Este procedimento procura definir métricas capazes de analisar cada um dos sub-objetivos que compõem o objetivo principal de forma que os objetivos sejam encadeados de maneira hierárquica na qual os níveis inferiores definem causas para problemas encontrados nos níveis superiores. Uma decisão difícil, neste procedimento,

está em definir quais são causas diretas e quais são causas indiretas para o correto encadeamento dos nós na estrutura.

Seguindo este procedimento, será possível decidir quais as possíveis causas para o desvio quando uma meta não for alcançada. É importante observar que a construção desta estrutura pode ser tão mais detalhada quanto mais maduros forem as métricas e o processo utilizados.

Esta estrutura pode ser vista como um *framework* capaz de integrar diversas métricas com características distintas (como, por exemplo, tipos diferentes de escala). Além disso, informações a respeito do contexto do projeto ou, até mesmo, das características do ambiente de desenvolvimento podem ser introduzidas e relacionadas como qualquer outra métrica. Esta é uma característica muito interessante, pois este tipo de informação é de suma importância para a correta interpretação dos resultados das medições. Todos estes dados, vistos em conjunto, formam o contexto geral do projeto e permitem um melhor apoio à tomada de decisão.

4.5 Estrutura de Decisão para o Primeiro Objetivo

A fim de exemplificar a construção desta estrutura causal, foi escolhido o primeiro objetivo proposto neste trabalho (melhorar a precisão das estimativas) para mostrar a utilização desta idéia. A partir de então, foi seguido o procedimento anteriormente definido como será descrito nesta seção.

O objetivo principal, que deve ocupar o topo da estrutura, é a "Precisão das Estimativas" e, de acordo com o passo I, este deve ser dividido em dois sub-objetivos diretos que são: "Precisão Total do Cronograma" e "Precisão Total de Esforço". Como não existe uma única métrica que possa descrever seu comportamento, deve ser considerado que o objetivo principal não foi alcançado quando, pelo menos, um dos sub-objetivos não for atendido (passo II). Abaixo pode ser vista a estrutura definida até o momento.

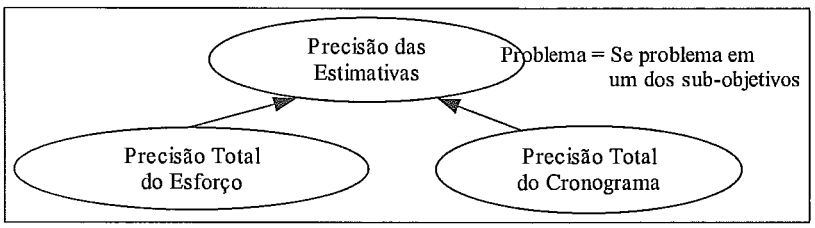


Figura 4.4: Estrutura de decisão para o objetivo 1 após uma execução do procedimento de geração.

Seguindo o passo III do procedimento definido, foi selecionado para análise o objetivo "Precisão Total do Cronograma" o qual pode ser detalhado na "Precisão do Cronograma por Macro-atividade" (passo I). Como pôde ser visto na Figura 4.1, a métrica definida para este objetivo é "o tempo total sobre o tempo estimado para todo o projeto". Neste ponto, cabe ao gerente do projeto determinar qual deve ser o valor máximo tolerado para esta métrica, ou seja, determinar qual o valor máximo de atraso aceito como um valor normal. Para isto, deve-se levar em consideração fatores como, por exemplo, as características da equipe e o nível de maturidade do processo.

Uma análise semelhante a esta foi feita para o outro sub-objetivo chegando a um grafo da seguinte forma:

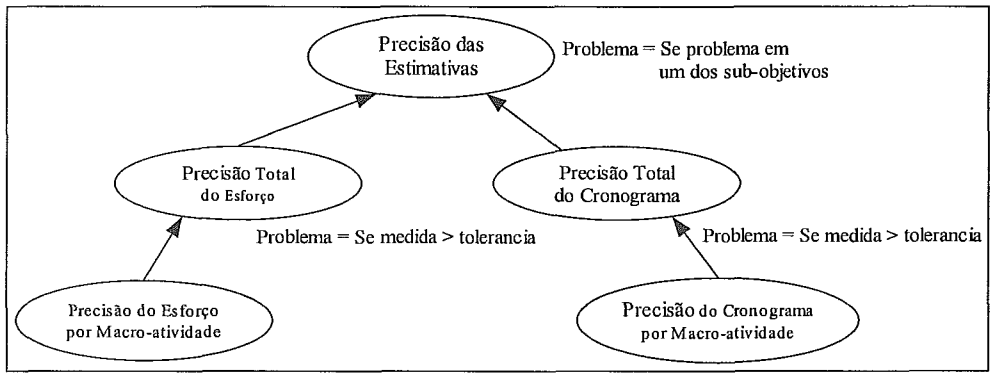


Figura 4.5: Estrutura de decisão para o objetivo 1 parcialmente desenvolvida.

Como pode ser observado, até este momento foi feita apenas uma reorganização do objetivo, das questões e das métricas selecionadas anteriormente seguindo a abordagem GQM.

Para a "Precisão do Cronograma por Macro-atividade" duas podem ter sido as causas de um atraso no cronograma segundo os especialistas: aumento de esforço ou emprego incorreto do esforço. O aumento de esforço já está sendo representado pela "Precisão do Esforço por Macro-atividade" devendo somente ser ligado como uma possível causa para o problema da precisão no cronograma. Por outro lado, o "Emprego do Esforço na Macro-atividade" deverá ser incluído à estrutura como uma segunda causa possível.

Aqui, novamente, será utilizada a mesma tolerância já definida para a "Precisão Total do Cronograma" para determinar se houve um problema na "Precisão do Cronograma por Macro-atividade".

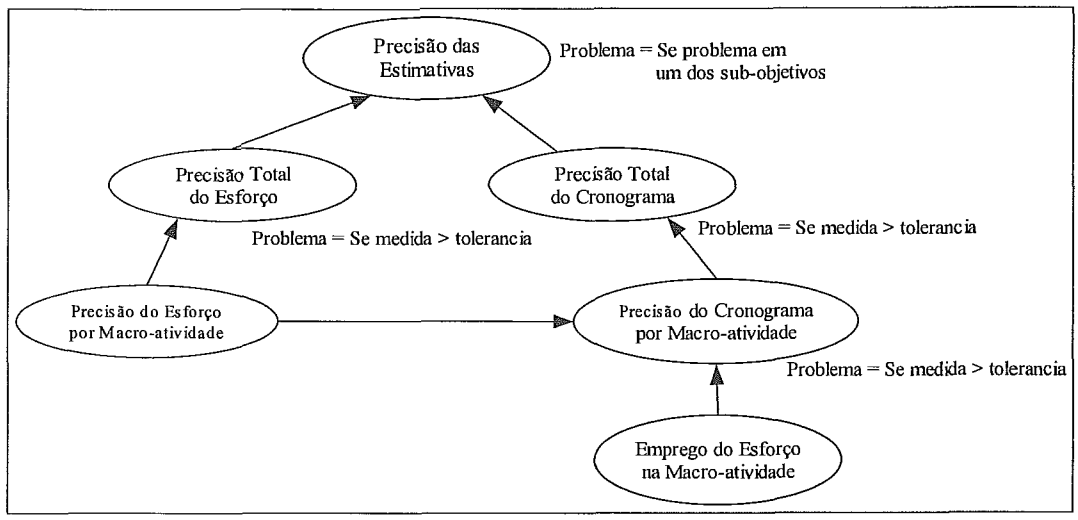


Figura 4.6: Estrutura de decisão para o objetivo 1 parcialmente desenvolvida.

Ao repetir esta análise para o "Emprego do Esforço na Macro-atividade" observamos que não é possível, para a grande maioria das empresas de *software*, detalhar melhor as possíveis causas para este problema. Desta forma, este não será detalhado em sub-objetivos sendo um nó terminal da estrutura. Assim, pode-se considerar que ocorreu um problema no "Emprego do Esforço na Macro-atividade" quando não houver um problema de "Precisão do Esforço na Macro-atividade", ou seja, se ocorreu um problema no tempo, mas o esforço foi dentro do previsto, então, conclui-se que o problema foi o incorreto emprego do esforço durante os trabalhos. Somente para facilitar a visualização, todos os nós terminais da estrutura serão representados por retângulos ao invés de elipses como pode ser visto na figura 4.7.

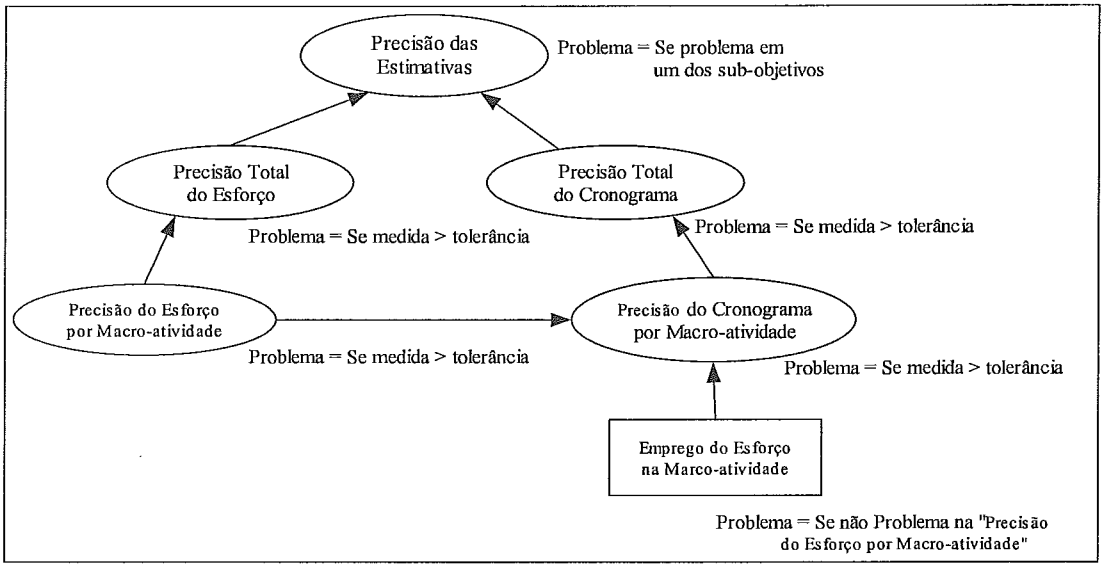
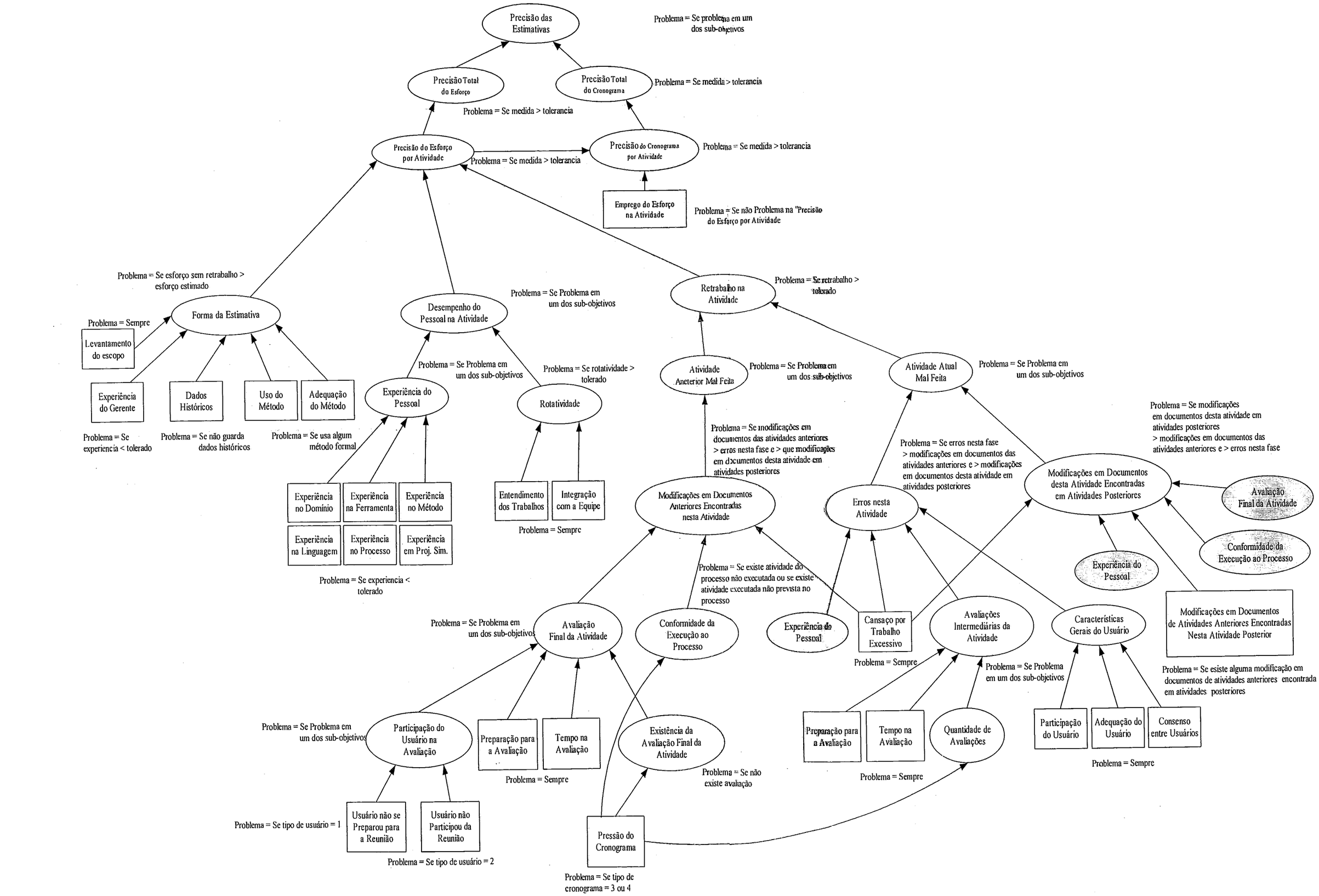


Figura 4.7 – Estrutura de decisão parcial para o objetivo 1 com nó terminal.

Este procedimento foi repetido para a "Precisão do Esforço por Macro-atividade" e seus sub-objetivos até que não existissem mais nós passíveis de serem expandidos, ou que não fosse interessante sua expansão no nível em que se encontram os trabalhos. A estrutura final pode ser vista na figura 4.8. A partir do sub-objetivo "Precisão do Esforço por Macro-atividade" várias outras métricas foram incluídas como, por exemplo, o número de erros e modificações ocorridos em cada Macro-atividade, a experiência da equipe de desenvolvimento, a rotatividade do pessoal e informações a respeito do contexto do projeto entre outras.

Com esta estrutura se tornou possível analisar melhor os possíveis problemas ocorridos quando o primeiro objetivo definido para esta abordagem não for alcançado.


Precisão das Estimativas
Problema = Se problema em um dos sub-objetivos
Precisão Total do Esforço
Precisão Total do Cronograma
Problema = Se medida > tolerancia
Problema = Se medida > tolerancia
Precisão do Esforço por Atividade
Precisão do Cronograma por Atividade
Problema = Se medida > tolerancia
Emprego do Esforço na Atividade
Problema = Se não Problema na "Precisão do Esforço por Atividade
Problema = Se esforço sem retrabalho > esforço estimado
Forma da Estimativa
Problema = Se Problema em um dos sub-objetivos
Desempenho do Pessoal na Atividade
Problema = Se retrabalho > tolerado
Retrabalho na Atividade
Problema = Sempre
Levantamento do escopo
Experiência do Gerente
Dados Históricos
Uso do Método
Adequação do Método
Problema = Se Problema em um dos sub-objetivos
Experiência do Pessoal
Problema = Se rotatividade > tolerado
Rotatividade
Atividade Anterior Mal Feita
Problema = Se Problema em um dos sub-objetivos
Atividade Atual Mal Feita
Problema = Se Problema em um dos sub-objetivos
Problema = Se experiencia < tolerado
Problema = Se não guarda dados históricos
Problema = Se usa algum método formal
Experiência no Domínio
Experiência na Ferramenta
Experiência no Método
Experiência na Linguagem
Experiência no Processo
Experiência em Proj. Sim.
Problema = Se experiencia < tolerado
Entendimento dos Trabalhos
Integração com a Equipe
Problema = Sempre
Problema = Se modificações em documentos das atividades anteriores > erros nesta fase e > que modificações em documentos desta atividade em atividades posteriores
Problema = Se erros nesta fase > modificações em documentos das atividades anteriores e > modificações em documentos desta atividade em atividades posteriores
Problema = Se modificações em documentos desta atividade em atividades posteriores > modificações em documentos das atividades anteriores e > erros nesta fase
Modificações em Documentos Anteriores Encontradas nesta Atividade
Erros nesta Atividade
Modificações em Documentos desta Atividade Encontradas em Atividades Posteriores
Avaliação Final da Atividade
Conformidade da Execução ao Processo
Experiência do Pessoal
Problema = Se existe atividade do processo não executada ou se existe atividade executada não prevista no processo
Problema = Se Problema em um dos sub-objetivos
Avaliação Final da Atividade
Conformidade da Execução ao Processo
Experiência do Pessoal
Cansaço por Trabalho Excessivo
Problema = Sempre
Avaliações Intermediárias da Atividade
Características Gerais do Usuário
Modificações em Documentos de Atividades Anteriores Encontradas Nesta Atividade Posterior
Problema = Se esiste alguma modificação em documentos de atividades anteriores encontrada em atividades posteriores
Problema = Se Problema em um dos sub-objetivos
Problema = Se Problema em um dos sub-objetivos
Participação do Usuário na Avaliação
Preparação para a Avaliação
Tempo na Avaliação
Existência da Avaliação Final da Atividade
Problema = Sempre
Problema = Se não existe avaliação
Preparação para a Avaliação
Tempo na Avaliação
Quantidade de Avaliações
Problema = Sempre
Participação do Usuário
Adequação do Usuário
Consenso entre Usuários
Problema = Sempre
Problema = Se tipo de usuário = 1
Usuário não se Preparou para a Reunião
Usuário não Participou da Reunião
Problema = Se tipo de usuário = 2
Pressão do Cronograma
Problema = Se tipo de cronograma = 3 ou 4

4.6 Medição do Processo de *Software*

Para tornar possível a medição e a melhoria de um processo de *software*, este deve ser definido de forma clara e precisa a fim de evitar problemas na sua interpretação e deve ser devidamente executado pela equipe de desenvolvimento para que os valores coletados sejam válidos para a sua avaliação.

A tarefa de definição do processo de *software* segue a abordagem proposta por (MACHADO, 2000) apresentada no item 3.3.2, e considera todo o procedimento de definição do Processo Padrão, sua especialização para o tipo de *software* e cultura organizacional e terminando na sua instanciação para um projeto específico.

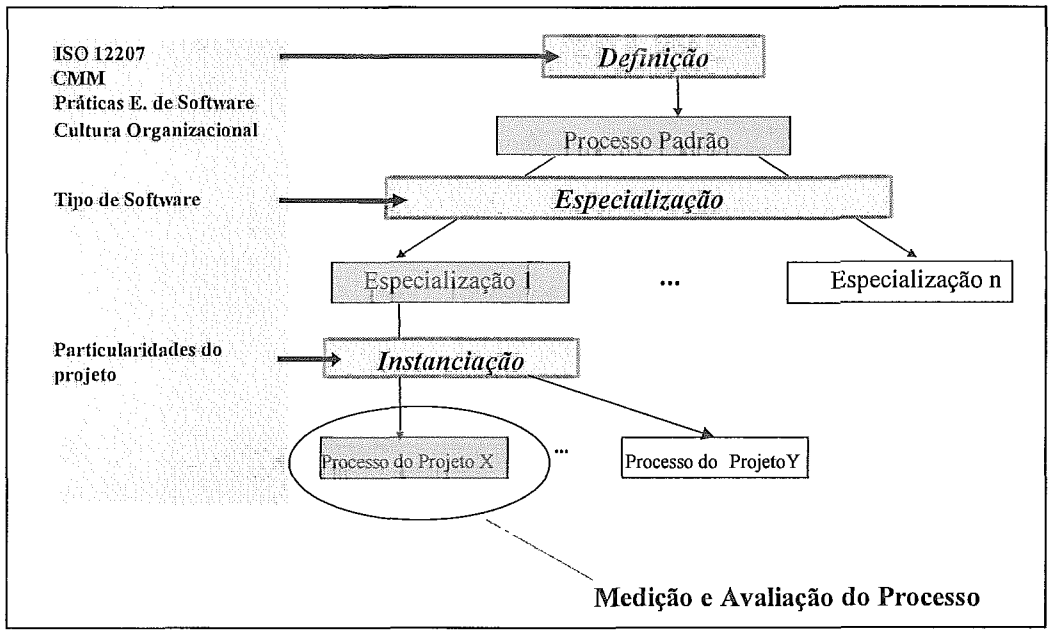


Figura 4.9: Modelo para Definição de Processos de *Software* com adaptação para a sua medição e avaliação

Porém, a partir da definição do processo, alguns dos documentos já previstos de serem elaborados durante o desenvolvimento do projeto foram adaptados e outros novos foram inseridos de forma que todos os dados necessários para a análise do processo fossem facilmente obtidos. Desta forma, a atividade de medição passou a ser parte integrante do próprio processo de desenvolvimento o que minimiza o tempo gasto na coleta de dados e permite separar as atividades de desenvolvimento das atividades de análise e melhoria do processo (comprometimento e). Além disso, este procedimento tenta evitar que dados inconsistentes ou incompletos sejam utilizados no procedimento de avaliação do processo, o que pode levar a erros de decisão.

Para concluir o exemplo da avaliação do processo de acordo com o primeiro objetivo proposto nesta abordagem ainda é necessário adaptar o processo de forma a incorporar os procedimentos de coleta de dados para o cálculo das métricas utilizadas. Como pôde ser visto na estrutura final para o primeiro objetivo apresentada na Figura 4.8, várias métricas foram utilizadas para a análise do processo e os seguintes documentos foram inseridos ou adaptados:

Estimativas de Projeto: o gerente do projeto deve estimar os valores de tempo e esforço para cada uma das macro-atividades existentes no processo utilizado. Neste momento, deve informar também se a estimativa foi feita de forma *ad hoc*, baseada em experiências passadas ou utilizando algum tipo de método formal de estimativa.

Planilha de Atividades: todos os membros da equipe devem informar em uma planilha diária quais foram as atividades do processo executadas, o tempo gasto em cada uma delas e se trata-se ou não de retrabalho.

Finalização de uma Atividade: para declarar que uma atividade do processo terminou é necessário um documento contendo a data da sua aprovação e a assinatura do responsável.

Com estas informações, pode ser criada uma forma automática capaz de levantar o tempo e o esforço empregados em cada macro-atividade do processo e o esforço empregado em retrabalho por macro-atividade.

Relatório de Erros: todas as atividades de avaliação de um artefato devem produzir um relatório dos erros encontrados durante a avaliação. Para facilitar os trabalhos iniciais, foi exigida somente uma breve descrição do erro a fim de garantir que um mesmo problema não fosse contado duas vezes. Com o aumento da maturidade, informações como o nível de gravidade do erro e o seu tipo podem enriquecer as análises.

Relatório de Modificações: sempre que for necessário modificar algum documento já aprovado, um relatório contendo uma breve descrição das modificações deve ser gerado. Assim como para o caso dos erros, as modificações inicialmente se restringirão a uma breve descrição para evitar duplicidade.

Aprovação de um Artefato: para declarar que um artefato foi totalmente concluído, é necessário um documento contendo a data da sua aprovação e a assinatura do cliente.

Desta forma, é possível saber o número de erros e modificações encontrados em cada artefato e as macro-atividade onde foram encontrados.

Questionário sobre a Experiência da Equipe: todos os membros da equipe precisam responder a um questionário simples contendo perguntas sobre sua experiência em projetos de porte similar, no domínio do projeto, no processo, nas ferramentas, na linguagem de programação e no método utilizados. Para diminuir a subjetividade desta métrica, foi estabelecida a seguinte classificação: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros.

Relatório de Rotatividade: documentação de todas as mudanças na equipe ocorridas durante o decorrer do projeto.

Questionário sobre o Contexto do Projeto: informações sobre o contexto do projeto respondidas pelo gerente antes de iniciar a avaliação do processo.

Com o correto preenchimento de todos este documentos durante os trabalhos, os dados para o cálculo das métricas para a avaliação do processo segundo o primeiro objetivo poderão ser, facilmente, obtidos.

4.7 Experiência de Medição

Com o intuito de experimentar o uso desta abordagem em um ambiente real, foi realizado o acompanhamento de um projeto desenvolvido por uma empresa situada no Rio de Janeiro que atua como fábrica de *software*. Como a equipe de Engenharia de Software da COPPE já havia definido o processo de software da empresa, foi seguida a abordagem deste trabalho, inserindo e adaptando documentos que a equipe deveria preencher para que fosse possível a coleta de todas as métricas.

No entanto, no decorrer do projeto, foi constatada a dificuldade de preenchimento dos documentos por parte da equipe, não por se tratar de uma tarefa complicada, mas, sim, por esta não ter o hábito de realizar este tipo de atividade. Além disso, foi observado que o conhecimento em Engenharia de *Software* por parte da equipe do projeto interfere significativamente na qualidade dos dados coletados. Desta forma, foi necessário analisar e, em certos casos, corrigir os dados coletados pela equipe a fim de que estes fossem coerentes com as definições das métricas utilizadas para a análise do processo.

Um dos documentos, entretanto, apresentou reais dificuldades para o seu preenchimento manual: a planilha de atividades. Este documento é de suma importância para o acompanhamento dos trabalhos, mas foi constatado que o seu preenchimento

manual é difícil e cansativo. A fim de evitar possíveis problemas, foi utilizada uma planilha eletrônica para apoiar o levantamento destes dados de forma que esta automaticamente totalizasse o número de horas trabalhadas por cada membro da equipe, detalhadas por macro-atividade.

Outro ponto que se mostrou relevante foi a necessidade de incorporar efetivamente o procedimento de coleta de métricas ao próprio processo produtivo. Para isso, não basta que o processo definido incorpore esta atividade, é preciso conscientizar a equipe do projeto da necessidade da coleta sistemática dos dados à medida que as atividades são executadas. Para isso, é necessário o treinamento da equipe e a apresentação dos ganhos obtidos com o uso destes valores numéricos. Além disso, é fundamental o apoio da alta gerência para garantir que esta coleta seja feita da forma adequada.

Todos os problemas e constatações levantados nesta experiência prática de uso da abordagem proposta neste trabalho vieram confirmar tudo aquilo que foi exposto anteriormente, comprovando que é necessário atender a todos os comprometimentos apresentados no início deste capítulo para que o uso da abordagem de medição obtenha êxito. No entanto, pôde-se observar que o componente humano é o principal fator que deve ser cuidadosamente analisado para tornar possível alcançar o sucesso para o uso de uma abordagem com esta.

Depois de finalizada a coleta dos dados, foram calculados os valores para as métricas necessárias para a avaliação do processo segundo o primeiro objetivo e foram estabelecidos os parâmetros de tolerância que deveriam ser utilizados na análise dos resultados das medições. Como, por questões de tempo, não era possível medir inicialmente um projeto para facilitar o estabelecimento destes parâmetros (como sugerido na abordagem) foram apresentados valores obtidos na literatura com a finalidade de orientar o responsável pelo projeto nesta definição. Porém, mesmo com esta ajuda, foi observado que ainda existe uma dificuldade muito grande para o estabelecimento de valores com os quais os gerentes e responsáveis não estão acostumados a trabalhar. Concluiu-se, então, que somente com o emprego desta abordagem e com uma maior familiaridade no uso de métricas que estes parâmetros estarão mais próximos da realidade da empresa. No entanto, para permitir que uma análise inicial destes dados fosse realizada, o responsável pelo projeto na empresa estabeleceu valores limites segundo o que chamou de "valores de bom senso".

A partir da coleta de todas estas informações, foi iniciada a análise do processo seguindo a estrutura de decisão definida anteriormente. Os problemas sugeridos nesta análise coincidiram com a observação sobre o processo e sobre o desenvolvimento do projeto.

O maior problema observado neste projeto foi a falta de precisão das estimativas de tempo e esforço do projeto ocasionadas por uma definição *ad hoc* destes valores. A falta de conhecimento da real capacidade da equipe do projeto é uma falha das definições feitas de forma aleatória e isto acabou contribuindo para aumentar o desvio entre os valores estimados e os valores reais. Assim, o primeiro ponto a ser analisado para os futuros projetos é a forma como as estimativas de projetos são realizadas e esta foi a primeira modificação que a empresa adotou para os próximos trabalhos.

De acordo com a estrutura de decisão, além do problema na forma como as estimativas são feitas, outra causa para o problema de precisão das estimativas pode ser o desempenho da equipe ou um alto índice de retrabalho. Como o percentual de retrabalho medido para a maioria das macro-atividades do projeto não ultrapassou o valor tolerado de 30% (exceto para uma macro-atividade cujo esforço total não é representativo para o valor total do projeto), foi considerado que a outra causa para o problema foi o desempenho da equipe do projeto. Este, segundo a estrutura de decisão, ocorre quando a experiência da equipe é baixa e/ou a rotatividade de pessoal é alta. Neste projeto, foi definido que experiência baixa seria aquela que obtivesse medidas abaixo do valor mínimo 2 (prática em até três projetos) de acordo com a escala das métricas de experiência utilizadas e foi estabelecido também que uma alta rotatividade seria quando o valor medido ultrapassasse o limite de 15%. Foi decidido pelo uso de uma tolerância baixa para a rotatividade por se tratar de um projeto curto no qual modificações na equipe podem trazer um impacto maior no desempenho.

Baseado nestes valores, foi constatado tanto uma baixa experiência da equipe em todas as métricas de experiência exceto uma (experiência na linguagem de programação) e, além disso, foi constatada uma rotatividade de 16,67% que também está acima do valor tolerado para esta métrica.

Assim, foi concluído que os principais problemas deste projeto foram: o estabelecimento das estimativas do projeto sem o uso de algum método formal ou bases históricas, uma alta rotatividade de pessoal acarretando em uma diminuição no desempenho total da equipe e, finalmente, a falta de experiência da equipe do projeto.

No entanto, o mais interessante desta experiência foram as observações do próprio responsável pelo projeto. Para ele, o principal valor de se estar seguindo e medindo um processo durante os trabalhos é que, desta forma, se torna possível determinar com maior objetividade em que ponto do processo estão os principais problemas e qual é o seu real impacto na execução do projeto. Esta foi uma conclusão muito importante e acredita-se que, sozinha, já justificaria o uso de processos de *software* e de uma abordagem para a análise da sua execução.

Além disso, foi constatado que o próprio processo de disciplinar as pessoas para coletar os dados referentes ao uso do processo e ao seu próprio desempenho, aumentou o conhecimento da equipe de desenvolvimento a respeito de suas capacidades e limitações fazendo com que o processo de coleta de métricas se torne, por si só, a primeira forma de melhoria do processo de *software* que a empresa experimentou.

4.8 Considerações Finais

É possível começar a entender o mundo real utilizando métricas pouco sofisticadas, mas à medida que a maturidade e o entendimento aumentam, formas mais detalhadas de medir as características das entidades do mundo real se tornam possíveis e, até mesmo, necessárias.

Um programa de melhoria de processos baseado no uso de métricas é parte da gerência da qualidade de *software* e deve evoluir à medida que técnicas de gerência e de desenvolvimento se modificam. Com o aprimoramento destas técnicas, melhorias no processo podem ser constatadas e ganhos de produtividade e qualidade são alcançados como uma conseqüência natural deste processo (GRADY e CASWELL, 1987).

A forma como as métricas estão sendo utilizadas atualmente provê pouca informação útil para o alcance destes objetivos. Vistas de forma isolada, cada métrica descreve uma característica particular que pode ser analisada e comparada com uma meta ou norma. Examinadas individualmente no decorrer do tempo, podem ser úteis para encontrar tendências e prever comportamentos futuros. Porém, vistas em conjunto, as métricas podem prover uma representação mais precisa do mundo real na qual o contexto geral do projeto pode ser visualizado. Esta pode ser uma poderosa ferramenta de análise ainda pouco explorada pelos modelos existentes. Um pequeno, mas significativo, conjunto de métricas analisado como um todo poderá trazer benefícios que, da forma tradicional, dificilmente poderiam ser alcançados.

No próximo capítulo, será apresentado o sistema construído para apoiar todo o processo definido neste capítulo desde a coleta dos dados até a sugestão dos possíveis problemas no processo utilizado durante um projeto.

Capítulo 5

Avaliação de Processos de *Software* na Estação TABA

Neste capítulo, é apresentada uma proposta para a avaliação semi-automatizada de processos de software desenvolvida para apoiar a abordagem teórica descrita no capítulo anterior. Este sistema foi construído a partir da infra-estrutura da Estação TABA, projeto no qual este trabalho está inserido.

5.1 Introdução

No capítulo anterior, foi apresentada a abordagem proposta para a avaliação de processos de *software* baseada no uso de métricas e foram definidos os passos que devem ser seguidos para a sua elaboração: seleção e definição do conjunto de métricas, incorporação da coleta de dados ao próprio processo produtivo e a organização destes dados em uma estrutura de decisão capaz de orientar o procedimento de avaliação do processo.

Porém, segundo os estudos realizados, constatou-se que as definições de métricas, por mais detalhadas que sejam, permitem que ambigüidades possam aparecer tanto no levantamento dos dados durante o decorrer do projeto como na interpretação dos resultados no final dos trabalhos (CARLETON *et al.*, 1992). Uma definição imprecisa pode levar a uma variação muito grande nos resultados causada pelo uso de diferentes procedimentos de medição. Somente a título de exemplo, o valor do número de linhas de código pode variar em até cinco vezes de uma contagem para outra devido à modificação da técnica utilizada (JONES, 1986). Desta forma, pior do que não ter dados sobre o processo para realizar a sua análise, é utilizar dados incompletos ou inconsistentes com a realidade para apoiar a tomada de decisão.

Foi, então, decidido que, para orientar a equipe de desenvolvimento nos procedimentos de coleta de métricas e análise dos resultados, seria construído um sistema capaz de implementar a abordagem teórica anteriormente apresentada e, desta forma, aumentar as possibilidades de sucesso para o seu uso. Qualquer que seja o processo de desenvolvimento utilizado, o sistema proposto deve ser capaz de apoiar o correto preenchimento dos documentos definidos para armazenar os dados referentes às

métricas e, a partir disso, utilizar a estrutura de decisão definida anteriormente para construir a base de conhecimento necessária para a avaliação do processo utilizado.

Neste capítulo, são apresentados os passos seguidos para o desenvolvimento de um sistema para a coleta de dados durante a execução de um projeto e um sistema baseado em conhecimento construído para atender a análise de processos de acordo com o primeiro objetivo definido para este trabalho (melhorar a precisão das estimativas). Este sistema está baseado na abordagem teórica apresentada no capítulo anterior e na infra-estrutura já existente na Estação TABA, projeto no qual este trabalho está inserido.

5.2 Avaliação de Processos na Estação TABA

Como apresentado anteriormente, a Estação TABA tem como objetivo instanciar Ambientes de Desenvolvimento de *Software* destinados a apoiar os trabalhos realizados durante o decorrer de um projeto específico. Para isso, deve-se, inicialmente, definir o processo de desenvolvimento a ser utilizado no projeto através do uso de algumas das ferramentas atualmente disponíveis no meta-ambiente e, a partir disto, instanciar o ambiente para apoiar a execução do processo. No entanto, é importante ressaltar que, atualmente, não existe nenhum tipo de acompanhamento de processo disponível nos ambientes instanciados pela Estação TABA.

Contudo, as funcionalidades de definição de processo e instanciação de ambiente formam os requisitos básicos para a construção de uma ferramenta capaz de apoiar o procedimento de avaliação de processos de *software* utilizado em projetos específicos. A partir destas funcionalidades, é possível alterar os ambientes instanciados de forma a permitir um acompanhamento simplificado da execução do processo com a coleta dos dados necessários para o cálculo das métricas utilizadas no procedimento de avaliação. Com estes dados coletados, pôde-se desenvolver um sistema baseado em conhecimento capaz de analisar estes resultados procurando encontrar os possíveis problemas existentes no processo.

Assim, tornou-se necessário modificar a estrutura dos ambientes instanciados de forma que fossem implantadas as seguintes funcionalidades: (i) acompanhamento do processo com a coleta semi-automática de métricas e a (ii) análise dos resultados das medições apoiada por um sistema baseado no conhecimento obtido de especialistas e descrito pela estrutura apresentada na figura 4.8 do capítulo anterior.

No entanto, a primeira etapa, o acompanhamento do processo com a coleta de métricas, pode ser tão melhor detalhada quanto melhores forem o processo e as métricas utilizadas. O ideal seria que todas as atividades realizadas durante o decorrer dos trabalhos fossem apoiadas pelo ADS permitindo que diversas informações fossem diretamente armazenadas pelo sistema. Este procedimento se torna desejável, pois permite que dados reais do projeto sejam armazenados sem aumentar o esforço da equipe e permitindo que inúmeras métricas sejam definidas a partir de combinações destes dados básicos. Desta forma, caso seja interessante o uso de alguma nova métrica, não se torna necessário modificar toda a base de dados, pois as mudanças estarão restritas ao procedimento de cálculo do valor da métrica que está sendo inserida no momento. Além disso, pode-se determinar o seu valor mesmo para projetos que foram realizados antes da sua inclusão, desde que os dados necessários para o seu cálculo tenham sido armazenados.

Como não seria possível desenvolver um sistema deste porte durante este trabalho, foi definido que somente os procedimentos de coleta dos dados necessários para a análise do processo segundo o primeiro objetivo seriam implementados. Porém, a estrutura foi definida de forma que, futuramente, outras funcionalidades possam ser inseridas nos ambientes instanciados de modo que novas métricas possam ser coletadas.

A seguir são apresentadas as funcionalidades implementadas para apoiar as duas etapas do procedimento de avaliação: primeiro a coleta semi-automatizada de métricas e, em seguida, a análise dos resultados das medições apoiada por um sistema baseado em conhecimento.

5.2.1 Medição do Processo

A primeira etapa para a avaliação de processos de *software* é a coleta dos dados para o cálculo das métricas. Atualmente, a Máquina de Processo proposta por ARAÚJO (1998) ainda não está implementada e, mesmo se estivesse, sozinha não seria capaz de coletar todos os dados necessários para a avaliação. Isso ocorre, pois esta apenas prevê a inclusão de mecanismos para coleta de métricas, mas não os provê realmente.

Tomando como base o acompanhamento de processo proposto por ARAÚJO (1998), foi desenvolvido neste trabalho um acompanhamento simplificado da execução das atividades durante o decorrer dos trabalhos. Para que fosse possível a implementação de todo o sistema de avaliação do processo no tempo previsto para a realização deste trabalho, foi necessário desconsiderar alguns dos estados nos quais cada

atividade pode passar. Assim, não está sendo permitido, por exemplo, que atividades do processo sejam interrompidas ou suspensas.

Além deste acompanhamento, foram inseridos novos procedimentos para tornar possível a coleta de todos os dados necessários à avaliação, os quais foram incorporados ao próprio processo produtivo, através do preenchimento de documentos durante o decorrer dos trabalhos. Estas alterações nos documentos foram apresentadas na abordagem teórica do item 4.6 do capítulo anterior e se resumem nos seguintes documentos: estimativas de projeto, planilha de atividades, aprovação de artefato, finalização de uma atividade, relatório de erros e modificações, questionário sobre a experiência da equipe, relatório de rotatividade e questionário sobre o contexto do projeto.

A inclusão destes procedimentos implicou em modificações no modelo da Estação TABA (anexo 3) para atender aos novos requisitos e na implementação de pequenas funcionalidades incorporadas ao ambiente instanciado de forma a apoiar o preenchimento de cada um dos documentos. Assim, ao final dos trabalhos, todos os dados necessários à avaliação do processo se encontrarão disponíveis no ambiente instanciado. Desta forma, fica a cargo de cada membro da equipe utilizar o ADS para preencher os documentos e prover estes dados ao sistema.

A primeira modificação a ser realizada no ambiente instanciado foi a definição dos membros que compõem a equipe do projeto. Antes estes ambientes possuíam somente informações de tipo e quantidade de recursos humanos utilizados (por exemplo, dois programadores e um analista). No entanto, agora se tornou necessário determinar cada membro especificamente e o tipo de recurso humano passou a indicar o papel que esta pessoa desempenha no projeto. Esta é uma informação importante para o sistema, pois é através desta que o controle de acesso a ferramentas e documentos está sendo realizado.

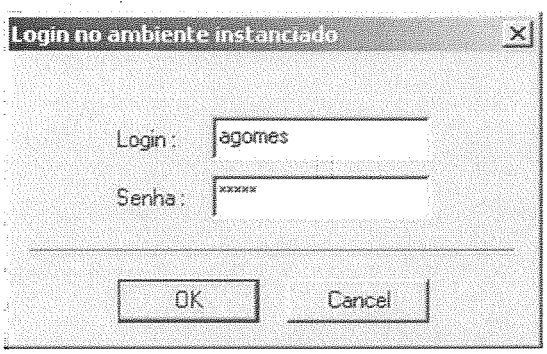


Figura 5.1: Tela de acesso ao Ambiente Instanciado pela Estação TABA

Foi incorporada ao ambiente instanciado uma tela de controle de acesso de forma a determinar exatamente qual é o usuário que o está utilizando no momento, como pode ser visto na figura 5.1. A partir de suas características, as opções de acesso às ferramentas e documentos são personalizadas.

A princípio todos os ambientes, quando instanciados, possuem um usuário *default*, o Super Usuário, responsável por cadastrar os membros da equipe que terão acesso ao sistema. Na figura 5.2, pode ser vista a tela de inclusão de novos usuários e, ao fundo, a tela de cadastro de usuários incorporada ao ADS. Para incluir um novo usuário, deve-se informar seu nome, login, senha e o seu tipo de recurso humano.

Pode-se observar na tela de cadastro de usuários uma coluna para registrar a saída de membros da equipe (a última coluna denominada "Reg. Saída"). Sempre que uma pessoa deixar o projeto antes do seu término, deve-se registrar sua saída para que esta informação possa ser utilizada no cálculo da métrica "Rotatividade de Pessoal". Este é um exemplo de ferramenta que somente será exibida para os membros da equipe cadastrados no sistema com o recurso humano do tipo "gerente de projeto".

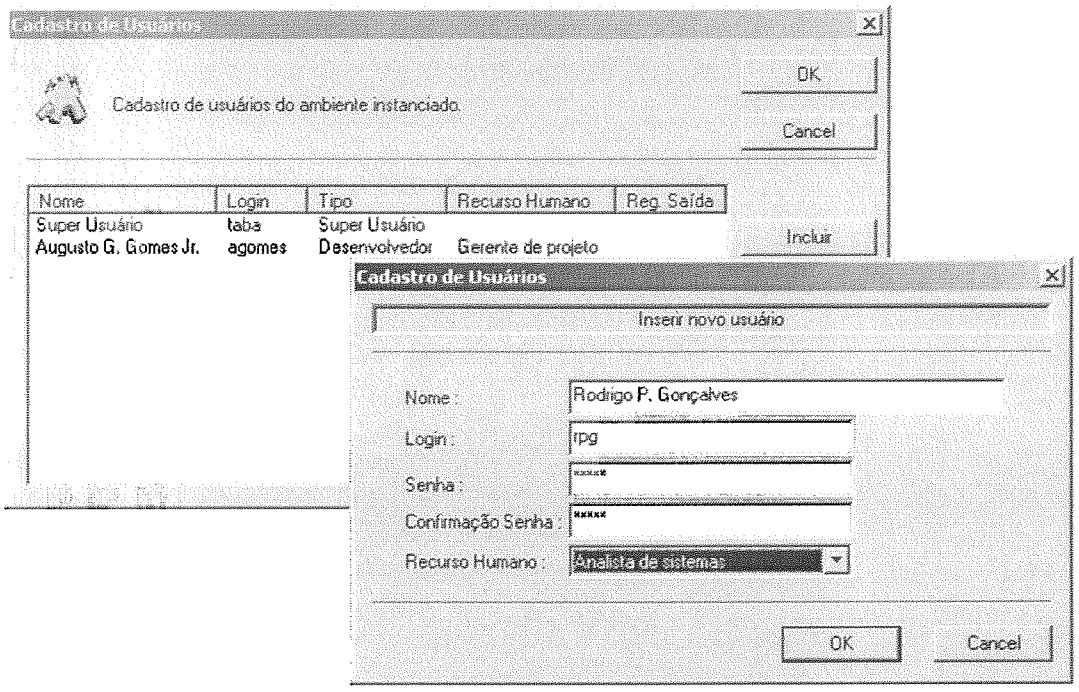


Figura 5.2: Tela de cadastramento de usuários do ADS.

Antes do início dos trabalhos, todos os membros da equipe devem preencher um questionário a fim de medir a experiência do grupo em cada um dos seis itens apresentados na figura 5.3 (projetos similares, domínio, método, ferramentas,

linguagens de programação e processo). Para cada um dos itens, está sendo utilizada a seguinte classificação: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros.

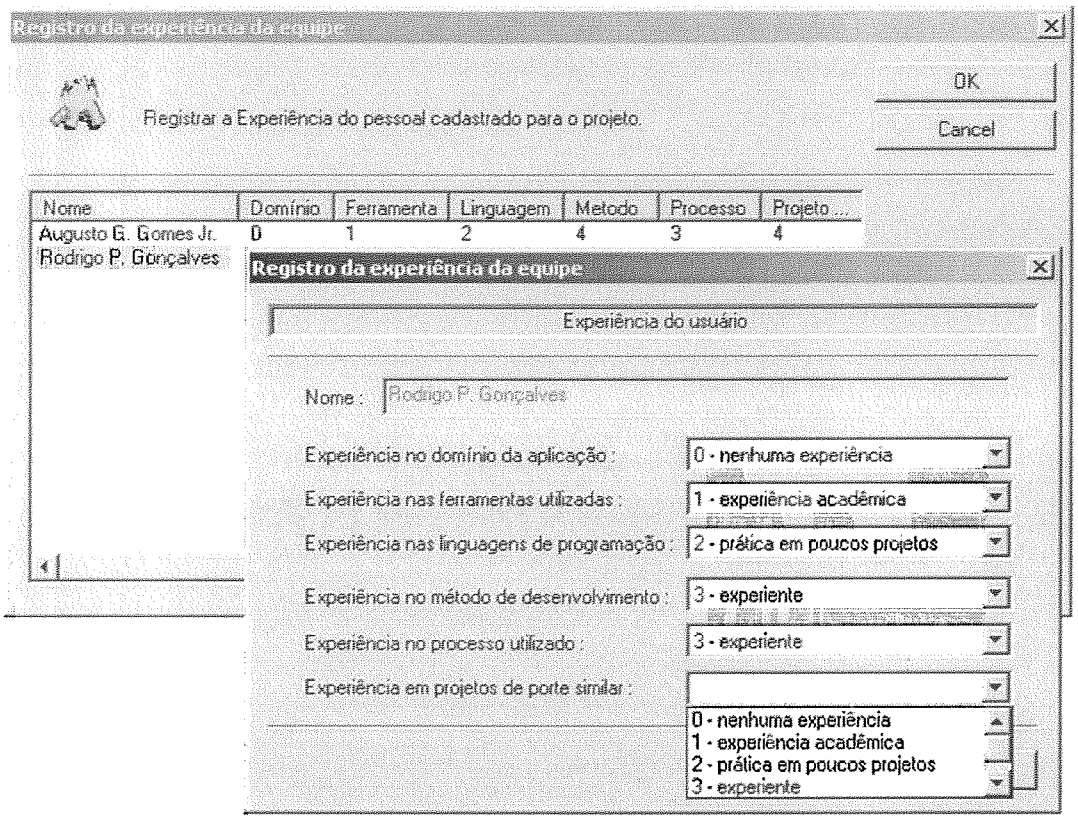


Figura 5.3: Questionário sobre a experiência da equipe do projeto.

Depois de permitir o cadastramento de todos os membros, a próxima tela a ser implementada foi a de Estimativas de Projeto. Nela, deverão ser estimados os valores de tempo e esforço para cada uma das macro-atividades que compõem o processo de desenvolvimento antes do início dos trabalhos, como pode ser visto na figura a seguir. É importante observar que esta funcionalidade apenas permite que sejam inseridos os valores previstos para o projeto. Futuramente, esta deverá ser substituída ou aperfeiçoada de forma a orientar o estabelecimento destes valores a partir do uso de métodos de estimativa existentes como, por exemplo, Pontos por Função ou COCOMO, ou a partir do uso de uma base histórica de projetos similares. No entanto, atualmente, é permitido informar ao sistema como a previsão foi realizada através do botão "Forma da Estimativa".

Seguindo as orientações da literatura vigente, o tempo foi definido como o número de dias decorridos a partir de uma data inicial e o esforço como o número de homens/hora empregados para a realização de cada macro-atividade.

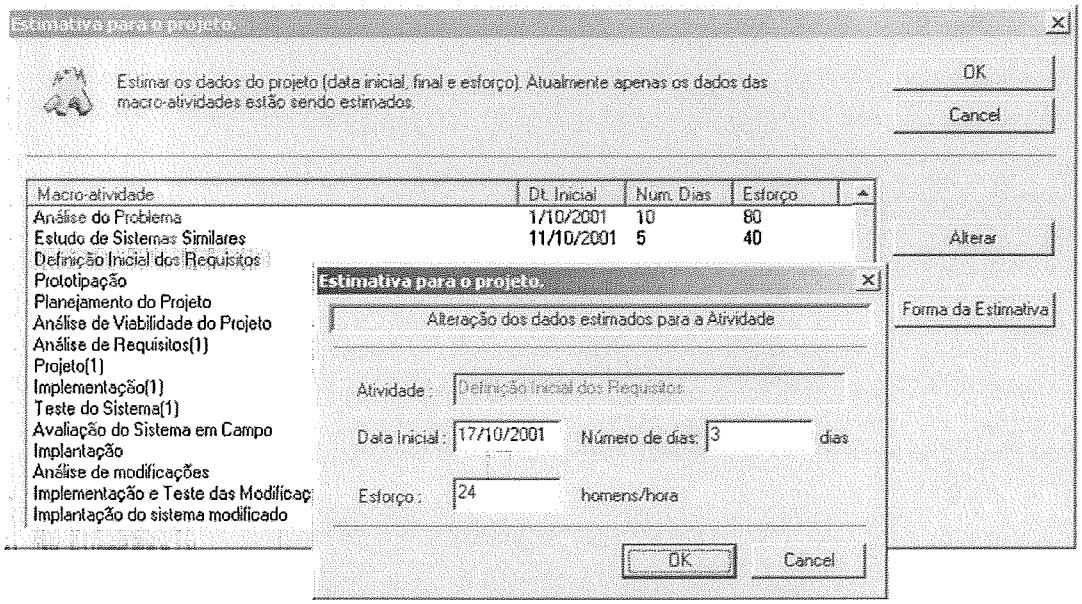


Figura 5.4: Tela de cadastro das estimativas de tempo e esforço para cada macro-atividade do projeto.

Definidos os usuários e as estimativas do projeto, pode-se iniciar a execução real do processo e, a partir deste momento, cabe ao ADS prover o suporte a esta execução. Neste sentido, a próxima ferramenta implementada foi a Planilha de Atividades na qual cada usuário deve registrar suas atividades, diariamente. Nesta ferramenta, devem ser informadas a atividade elementar do processo que foi executada, a data de sua execução, a hora de início e de término, uma breve descrição dos trabalhos realizados e se trata-se ou não de um retrabalho. É importante observar que somente as atividades elementares podem ser realmente executadas, ou seja, a execução de uma macro-atividade ou atividade intermediária deve ser realizada através da execução das atividades elementares que a compõem.

Quando algum membro da equipe executa pela primeira vez uma atividade, esta passa do estado "Planejamento" para o estado "Ativada"[4]. Além disso, a data da execução da atividade é armazenada como sua data de início e o esforço total é

[4] Esta nomenclatura segue a proposta de ARAÚJO (1998)

incrementado com o valor do tempo empregado. Sempre que uma atividade elementar é executada, as informações de data de início e de esforço são "repassadas" para a atividade de nível hierárquico imediatamente superior, sendo que isto é feito de forma recursiva até que a macro-atividade seja atualizada. Desta forma, toda atividade não elementar (macro ou intermediária) totaliza o esforço empregado nas atividades elementares que a compõem. Além disso, sua data de início é a menor dentre as datas de início de suas elementares.

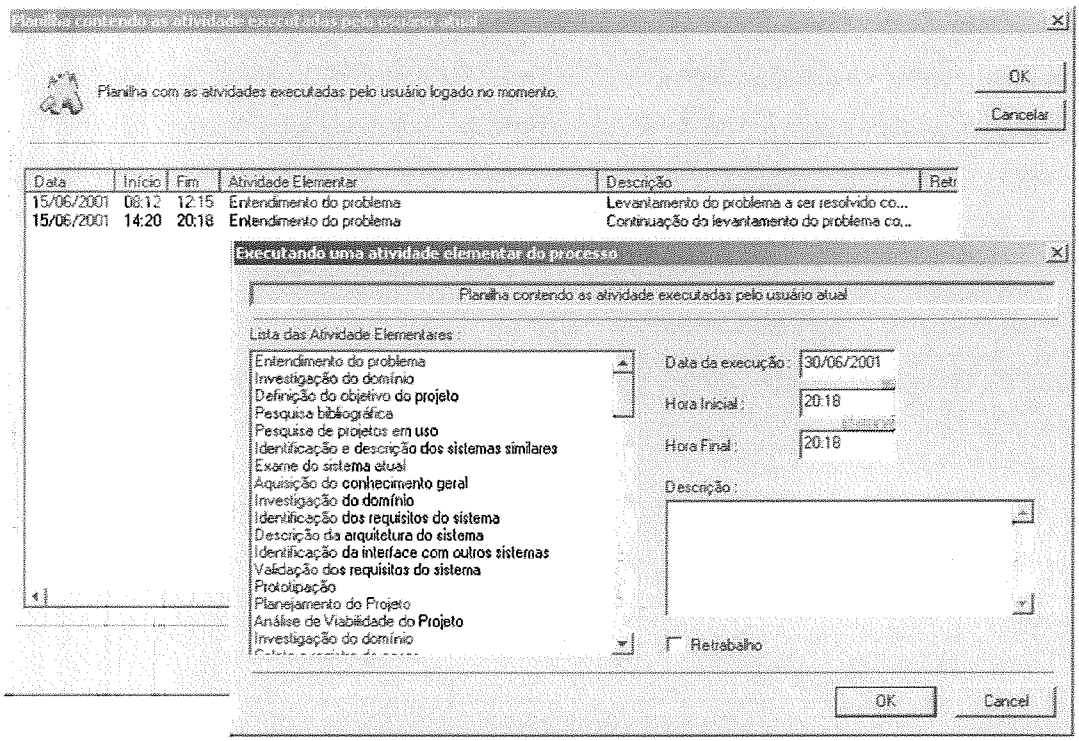


Figura 5.5: Planilha de Atividades parcialmente preenchida.

Retrabalho foi definido como qualquer atividade de correção, modificação ou melhoria realizada sobre um artefato que já deveria ter sido finalizado. Sempre que isso ocorrer, deve-se selecionar a opção "Retrabalho" e, assim, permitir ao sistema armazenar o esforço empregado em retrabalho em cada atividade do processo. Este dado é necessário para calcular o percentual de retrabalho por macro-atividade e é uma importante fonte de redução de custos para os projetos. No entanto, é importante deixar bem claro para a equipe de desenvolvimento qual é a definição de retrabalho para que a coleta deste dado seja feita corretamente. Além disso, é importante estar sempre lembrando que estes dados não serão utilizados para a avaliação de pessoas, mas, sim,

do processo utilizado, pois, caso isto não fique bastante claro, a tendência é que esta função nunca seja utilizada.

Quando um artefato for considerado finalizado, é necessário informar ao sistema a data na qual sua aprovação foi realizada. Para isto, foi desenvolvida uma funcionalidade que permite ao(s) gerente(s) de projeto registrar esta data, como pode ser visto na figura 5.6.

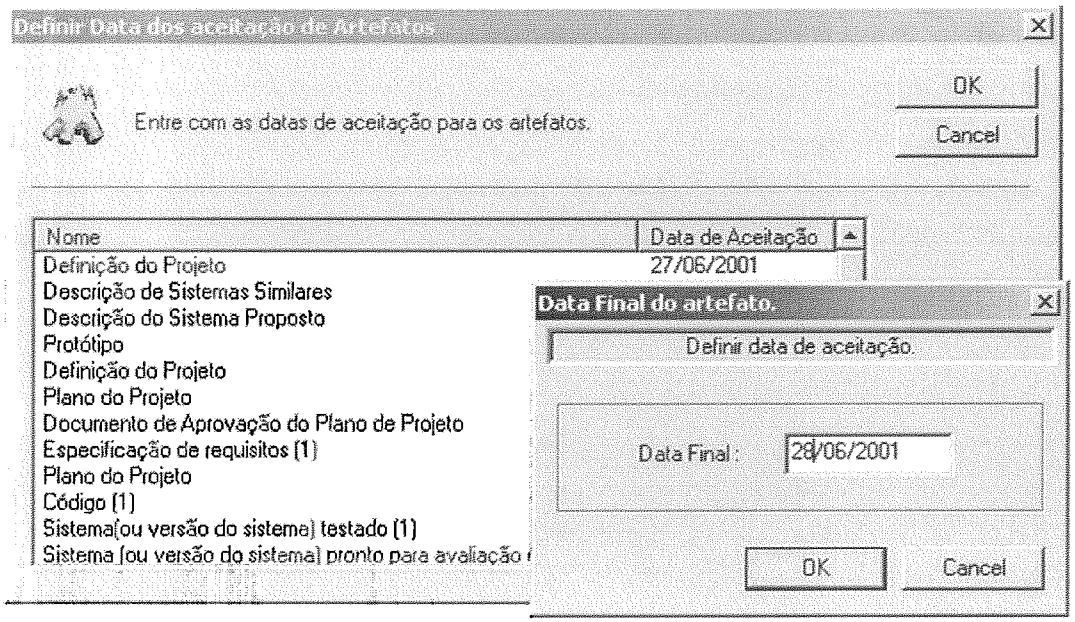


Figura 5.6: Registro da data de aprovação de um artefato.

De forma semelhante, para informar ao sistema que uma atividade foi considerada finalizada deve-se informar a sua data na tela apresentada na figura 5.7. Esta opção será permitida, somente, para as atividades elementares do processo de forma que a data final de uma macro-atividade ou atividade intermediária seja a maior data final entre as atividades elementares que a compõem.

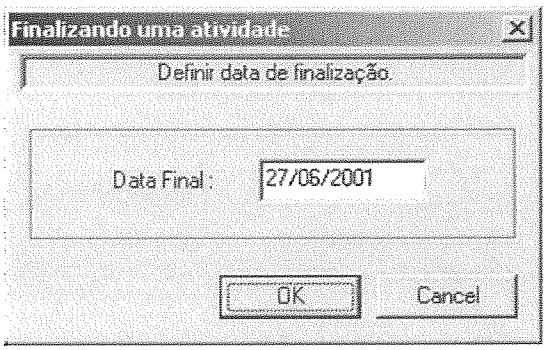


Figura 5.7: Registro da data de finalização de uma atividade elementar.

Antes que uma macro-atividade seja aprovada, deve existir uma atividade para a avaliação do artefato produzido. Para todas estas atividades de avaliação, será incluída no painel de ferramentas da tela principal do ambiente instanciado uma ferramenta específica para a avaliação de artefatos. Como pode ser visto na figura 5.8, esta necessita que sejam informados a data da avaliação e o artefato avaliado. Para cada avaliação são apresentados os erros já cadastrados e é permitida a inclusão de novos.

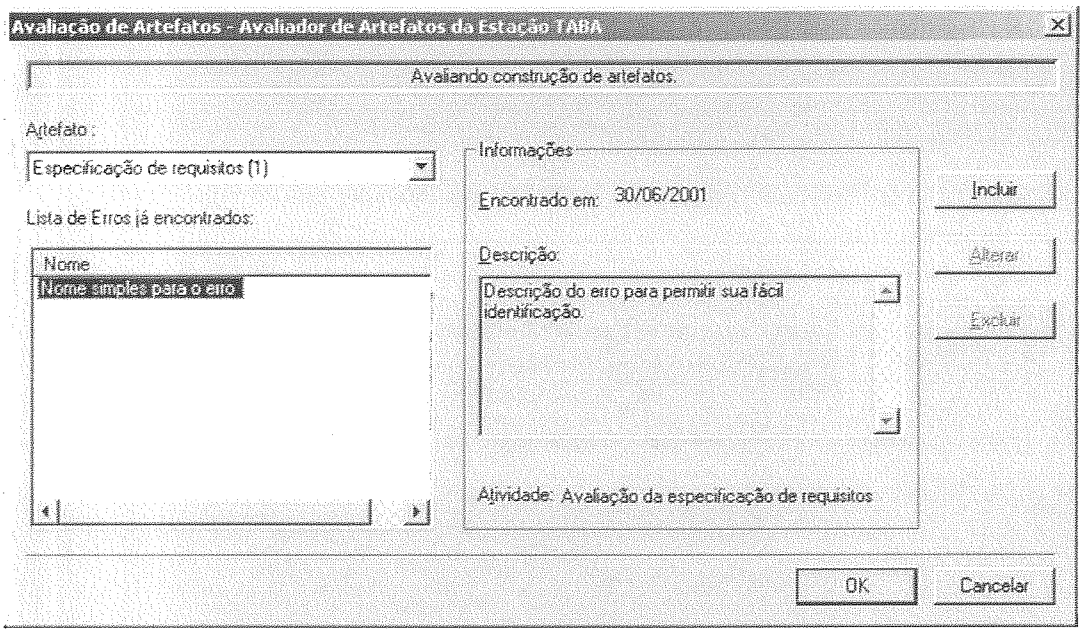


Figura 5.8: Cadastro de erros em artefatos.

Além dos erros encontrados nas atividades de avaliação, qualquer atividade pode encontrar problemas nos artefatos produzidos nas atividades anteriores. Sempre que isso ocorrer, deve-se cadastrar a modificação, utilizando, para isso, uma tela similar ao cadastro de erros que é chamada a partir da planilha de atividades. Desta forma, o desenvolvedor, ao realizar uma atividade e inclui-la nesta planilha, poderá cadastrar todas as modificações encontradas nos artefatos já validados durante a execução.

De acordo com a lista de documentos a serem preenchidos apresentada anteriormente, ainda falta o questionário sobre o contexto do projeto. Porém, como esta é uma informação muito específica utilizada somente no procedimento de decisão para o primeiro objetivo, foi decidido que este questionário seria preenchido somente no momento da avaliação do processo e, por isso, será visto mais à frente no próximo item deste capítulo.

Ao final do preenchimento destes documentos, todos os dados necessários se encontrarão disponíveis no ambiente instanciado de forma que possa ser realizada a segunda etapa do procedimento de avaliação do processo, a análise dos dados.

5.2.2 Análise dos Resultados da Medição

Como apresentado anteriormente, a avaliação de um processo de *software* baseada na análise do resultado de medições é uma tarefa complicada que necessita de real conhecimento para a sua execução. No entanto, atualmente este conhecimento ainda é bastante específico estando restrito a um conjunto relativamente pequeno de especialistas no assunto como, por exemplo, doutores na área de engenharia de software ou profissionais com grande experiência em gerência de projetos de diferentes tamanhos. Além disso, uma característica importante deste tipo de problema é que não existe uma única resposta correta e várias alternativas de solução podem ser sugeridas.

Estes são fatores que dificultam consideravelmente a construção de um sistema convencional para a realização automática desta tarefa. Assim, baseado nas características do problema e na grande necessidade de obtenção de conhecimento de especialistas para a sua resolução, foi decidido que seria construído um sistema baseado em conhecimento (SBC) para apoiar as equipes de desenvolvimento na execução desta etapa de avaliação dos processos de *software*.

Um SBC é um sistema capaz de aplicar o conhecimento de especialistas para a resolução de problemas do mundo real de difícil solução. Para isso, este se baseia em um conjunto de fatos observados no domínio e em regras que representam o conhecimento de como e quando estes fatos devem ser utilizados no procedimento de inferência. Esta base de conhecimento deve ser flexível permitindo que o conhecimento possa ser adicionado e modificado ao longo do tempo. Assim como os especialistas, um SBC utiliza heurísticas para encontrar respostas satisfatórias para o problema em questão e, desta forma, a solução obtida com o uso destes sistemas é de caráter inexato, porém com um grau de imprecisão dentro de limites tolerados. A tarefa de análise e raciocínio é realizada pela máquina de inferência, componente independente do domínio que trabalha com as informações da base de conhecimento para efetivamente solucionar o problema.

Como o conhecimento é o ponto principal dos SBC, a sua aquisição foi o primeiro passo para a construção do sistema proposto. Devido à sua importância, o estudo do

conhecimento humano é uma área bastante extensa e várias foram as técnicas propostas para a sua aquisição, dentre as quais vale destacar: o uso do meio bibliográfico, entrevistas realizadas com especialistas, ferramentas de edição de conhecimento, prototipação, etc. No entanto, como já havia sido realizado o levantamento de trabalhos relacionados ao problema a ser resolvido, apresentado no capítulo 2, foi considerado suficiente o levantamento do conhecimento prático de especialistas no assunto para complementar a base teórica selecionada.

Para a tarefa de aquisição do conhecimento foram utilizados questionários e entrevistas com alguns especialistas da área. Inicialmente, este grupo era composto por seis membros tanto do meio acadêmico quanto do empresarial. Mas, devido à dificuldade de encontro conjunto e da obtenção de consenso para o conhecimento a ser utilizado, este grupo precisou ser reduzido para somente duas pessoas do meio acadêmico. Desta forma, foi possível agilizar o procedimento de elicitação e verificar a viabilidade da abordagem proposta.

Com o andamento dos trabalhos e com o amadurecimento da abordagem teórica proposta, foi estabelecido o procedimento de definição da estrutura de decisão apresentada no item 4.4 do capítulo anterior que, baseada na abordagem GQM, utiliza o conhecimento de especialistas para organizar o conjunto de métricas coletadas em uma estrutura de forma que os níveis inferiores representam causas diretas que podem explicar os problemas encontrados nos níveis superiores. Este foi um processo exploratório lento e como resultado foi definida a estrutura de decisão apresentada na figura 4.8.

Depois disto, era necessário projetar o sistema e implementar a solução encontrada de forma integrada aos ambientes instanciados na Estação TABA. Assim como os demais sistemas convencionais, os sistemas baseados em conhecimento também possuem métodos de desenvolvimento que orientam na transformação da solução abstrata para um problema em uma representação computacional capaz de ser implementada em uma linguagem de programação adequada. Desta forma, foi utilizado o método chamado KADS-estendido proposto por WERNECK (1995) que, como o próprio nome diz, trata-se de uma extensão ao método KADS (*Knowledge Aquisition and Design Structure*) (VOB e KARBACH, 1993). Esta foi uma extensão proposta com o objetivo de diminuir as dificuldades encontradas para a realização do projeto do sistema a partir de sua especificação e, posteriormente, sua passagem para a etapa de implementação.

Este, como qualquer outro método de desenvolvimento de sistemas, possui um conjunto de modelos que podem ser utilizados a fim de diminuir a distância entre a concepção do problema a ser solucionado e a sua implementação computacional. Considerando a equipe reduzida e o pequeno escopo definido para o projeto, foram selecionados apenas os três principais modelos propostos pelo método para documentar o processo de análise do conhecimento: o modelo de Especialidade, o modelo Lógico e apenas um diagrama do modelo Físico, como pode ser visto no anexo 2.

É importante enfatizar que o objetivo principal deste trabalho não é a construção de um sistema baseado em conhecimento, mas a definição de uma abordagem para a avaliação de processos na qual o SBC está sendo utilizado como uma solução razoável destinada a apoiar uma de suas etapas. Neste momento, pretende-se mostrar apenas que é possível realizar esta tarefa de forma automática. Assim, algumas das decisões tomadas durante o seu desenvolvimento se referem mais a questões de facilidade de uso baseado na capacidade técnica e no tempo de execução que a questões como desempenho ou perfeita adequação.

Desta forma, primeiramente foi construído o diagrama da Estrutura do Domínio do Problema (que faz parte do Modelo de Especialidade). Para isto, foram consideradas as classes já existentes no modelo do ambiente instanciado a fim de manter coerência entre os modelos. Neste passo, foram definidas as principais classes do domínio que, de alguma forma, compõem a base do SBC. Em seguida, foi analisada a biblioteca de tarefas genéricas do KADS e, como o próprio título deste trabalho apresenta, foi selecionada a tarefa de Avaliação como a mais adequada para modelar o processo de raciocínio para resolução do problema. Baseado nesta decisão, foram desenvolvidos os diagramas da Estrutura de Inferência, Estrutura de Tarefas e a Árvore de Tarefas finalizando, assim, o Modelo de Especialidade.

Foi seguido, então, o procedimento de transformação da Estrutura de Inferência no Diagrama Heurístico do Raciocínio apresentado em (WERNECK, 1995) e, em seguida, foi feito o Diagrama do Domínio do Problema que compõem o modelo lógico. No entanto, foi na construção do Diagrama Estrutural da Base de Conhecimento, já no Modelo Físico, que foi possível visualizar completamente a solução para a implementação deste sistema. Com este diagrama, foi possível identificar três módulos principais que juntos compõem a base de conhecimento implementada.

O primeiro módulo contém os dados coletados durante o decorrer do projeto e seriam utilizados para compor as métricas e as tolerâncias utilizadas no processo de

decisão. Este é o Modulo de Fatos do sistema que tem o objetivo de retratar a realidade de cada projeto específico e, desta forma, basta alterar os dados deste módulo para realizar a avaliação de diferentes projetos seguindo a mesma estrutura de decisão.

O segundo módulo é o de regras que descrevem o comportamento apresentado esquematicamente pela estrutura de decisão da figura 4.8. Este é composto apenas de dois tipos de regras. A primeira retrata a relação de hierarquia descrita na estrutura, ou, segundo a nomenclatura utilizada na construção da estrutura de decisão, retrata quais são os sub-objetivos do objetivo analisado descrevendo a relação de causa entre eles. A segunda descreve como é determinado se um nó da estrutura possui ou não um problema, isto é, se existe algum desvio do objetivo previsto. Desta forma, este segundo módulo reúne o conhecimento adquirido com os especialistas e, por isso, foi chamado de Módulo de Conhecimento. Caso seja necessário modificar a estrutura de decisão, devido à descoberta de um relacionamento ainda não representado por esta, basta modificar este módulo para que o sistema automaticamente retrate este novo comportamento.

No entanto, somente com os dados medidos do projeto e com a representação da estrutura de decisão não seria possível para o sistema inferir as causas para os desvios. Era necessário introduzir na base o conhecimento de como construir a solução do problema para um projeto específico. Assim, um terceiro módulo capaz de implementar a análise das informações contidas no módulo de dados baseada na estrutura descrita no módulo de conhecimento foi criado. Este é o Módulo Solução que é o responsável por efetivamente resolver o problema analisando as medidas de cada métrica e caminhar na estrutura de decisão para encontrar as possíveis causas para a ocorrência de um desvio dos valores tolerados.

Organizando a base de conhecimento desta maneira, é possível avaliar vários projetos apenas modificando o módulo de dados e, além disso, criar novos módulos de conhecimento para a análise dos demais objetivos propostos neste trabalho. Isto permite, ainda, que o conhecimento inserido nesta ferramenta seja expandido a partir de pequenas alterações neste módulo. Como discutido anteriormente, não seria possível desenvolver a estrutura de decisão para os demais objetivos neste trabalho, mas tomou-se o cuidado de utilizar uma solução que permitisse que esta expansão fosse realizada sem a necessidade de qualquer reestruturação da base de conhecimento.

Para tornar possível a implementação deste sistema, era necessário utilizar uma linguagem que pudesse se comunicar, de alguma forma, com os ADS instanciados pela

Estação TABA implementados na linguagem C++. Para isso, foi selecionada a linguagem Prolog que já fazia parte da infra-estrutura do meta-ambiente. Prolog é uma linguagem simples, mas poderosa, capaz de validar a proposta de automatização da análise do resultado das medições. Assim, esta máquina Prolog foi disponibilizada, também, para os ADS instanciados.

Finalmente, após a implementação do sistema nesta linguagem, foi definida a interface para ser integrada aos ambientes instanciados. Para isso, foi incluída ao menu de Projeto a opção Avaliação do Processo que somente será habilitada quando todas as atividades do processo estiverem finalizadas e somente será permitida para o(s) gerente(s) de projeto.

Como pode ser visto na figura a seguir, inicialmente, é apresentado ao gerente uma tela onde este poderá informar qual o objetivo da avaliação do processo. Atualmente, são apresentados os três objetivos definidos nesta abordagem, porém, somente quando o primeiro estiver selecionado é que a opção Avançar é habilitada.

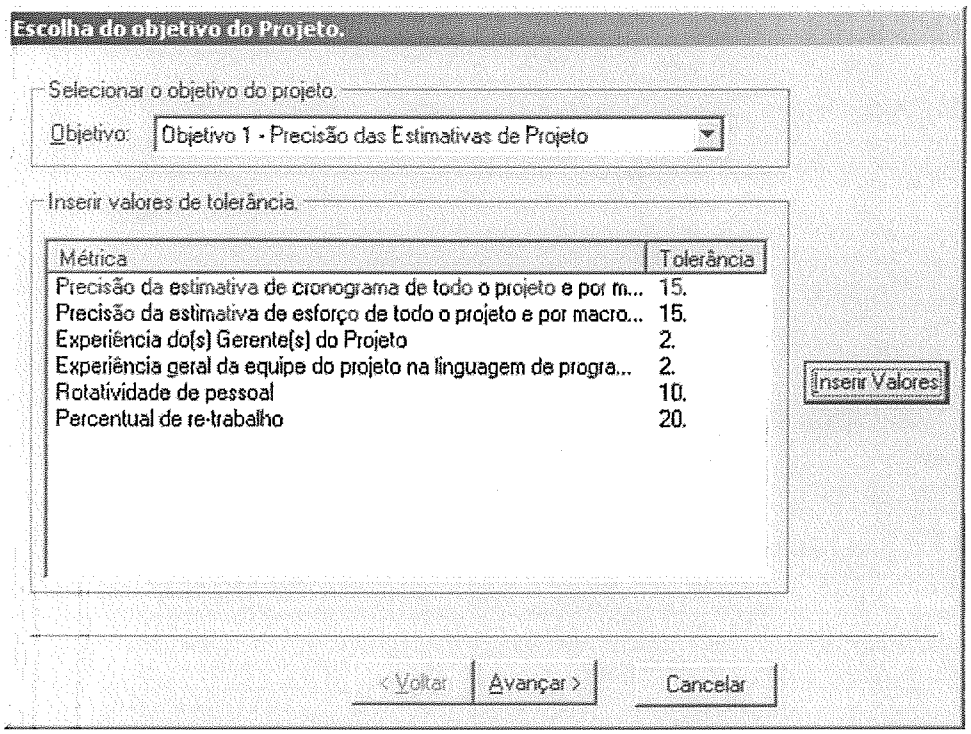


Figura 5.9: Definição do objetivo da avaliação e dos parâmetros de tolerância.

Ao selecionar um objetivo, são apresentadas as métricas que necessitam da definição de seu valor de tolerância utilizado pelo SBC e, somente após esta definição, é que será permitido o prosseguimento do processo de avaliação.

Como dito anteriormente, se o objetivo selecionado for o de "Melhorar a Precisão das Estimativas de Projeto", será apresentado um pequeno questionário para levantar informações a respeito do contexto do projeto de forma a compor a base de informações necessárias para a análise segundo este objetivo. Como pode ser visto na figura 5.10, este procura definir as características gerais dos usuários que auxiliaram o desenvolvimento do projeto e sua participação nas reuniões de avaliação. A escolha correta dos usuários para apoiar a produção de um *software* pode ser essencial para o sucesso do projeto e esta informação pode ser necessária para a avaliação. Além disso, questiona-se a flexibilidade do cronograma estipulado para o processo. Cronogramas flexíveis facilitam a execução de atividades para a garantia da qualidade do produto final e diminuem a pressão sobre a equipe do projeto, pressões estas que podem levar ao aumento de defeitos no produto desenvolvido. Como dito anteriormente, este questionário não foi apresentado na etapa de medição do processo por se tratar de questões bastante específicas desta estrutura de decisão.

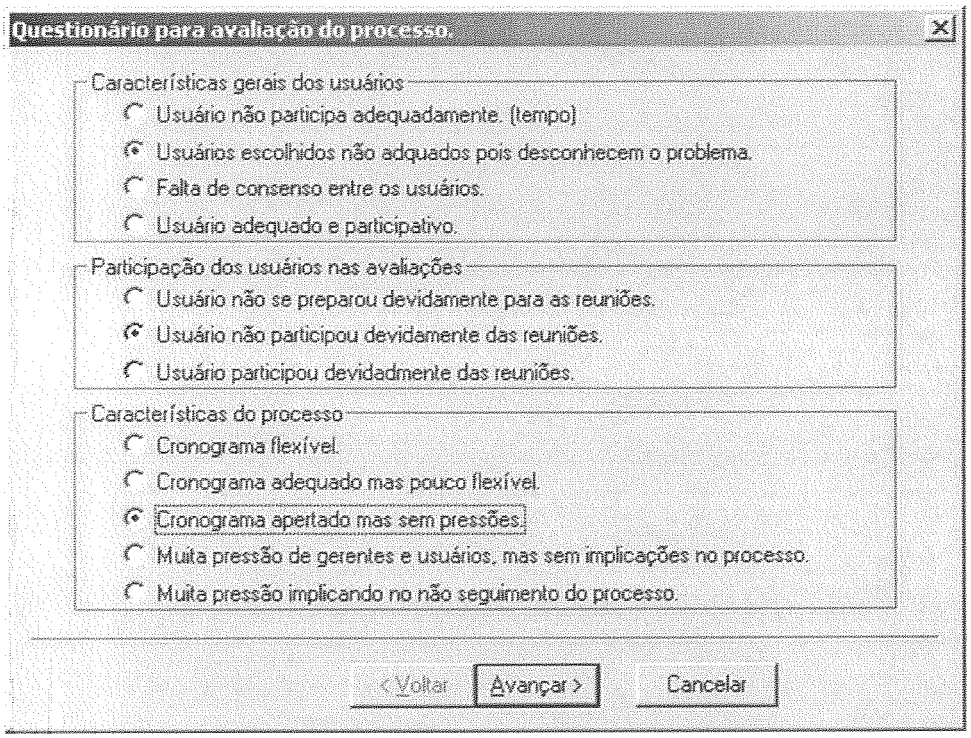


Figura 5.10: Questionário sobre o Contexto do Projeto.

Foi decidido implementar esta tela em seqüência após a tela de definição do objetivo e dos valores de tolerância para garantir que todos os passos necessários para a avaliação fossem seguidos. Desta forma, somente depois de informados todos estes dados é que o ADS será capaz de criar o Módulo de Fatos do SBC para o projeto atual. Feito isto, o passo seguinte é chamar o SBC para executar o procedimento de inferência retornando ao ADS uma estrutura semelhante àquela apresentada na figura 4.8, porém contendo apenas os nós que possuem algum problema comprovado pela análise dos dados.

Neste momento, o ADS apresenta esta resposta em uma estrutura de árvore de diretórios de forma que os níveis mais internos representam causas para os problemas encontrados nos níveis mais externos, como pode ser visto na figura 5.11. Assim, são apresentados ao gerente do projeto os principais problemas encontrados na análise e, caso haja interesse, é possível detalhar cada um destes desvios em suas possíveis causas. Baseado nestes dados, o gerente pode traçar planos para evitar que estes problemas aconteçam nos próximos projetos e, se considerado pertinente, modificar o próprio processo de forma a incorporar estas modificações para quaisquer outros projetos que o utilizarem.

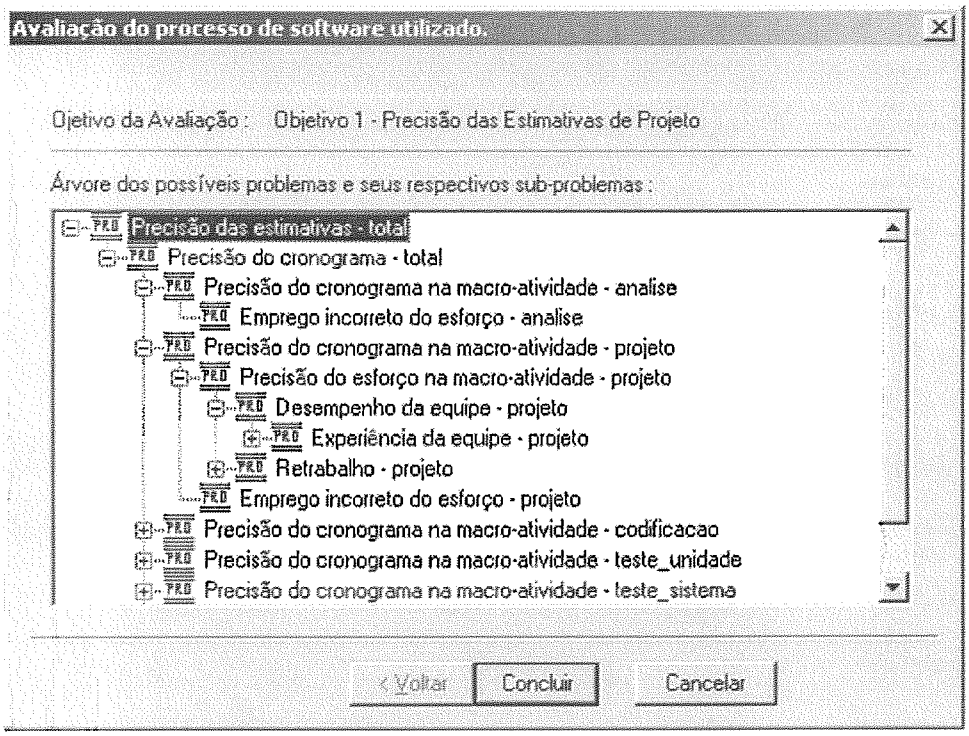


Figura 5.11: Resultado do procedimento de avaliação do processo.

5.3 Considerações Finais

Neste capítulo, foi apresentado o sistema construído para apoiar a abordagem teórica de avaliação de processos de *software* proposta neste trabalho. A construção das ferramentas é apenas uma forma de facilitar o trabalho da equipe de desenvolvimento e tentar garantir o correto emprego dos procedimentos aqui apresentados. Desta forma, os gerentes e engenheiros de *software* têm uma ferramenta capaz de apoiar a melhoria dos processos definidos no meta-ambiente TABA.

Porém, já foi constatado que o uso de ferramentas não é suficiente para o sucesso de nenhum procedimento ou abordagem. Na realidade, o aspecto menos problemático, geralmente, é a tecnologia, enquanto pessoas e processos são fatores críticos que podem fazer a diferença entre o sucesso e o fracasso (ZAHRAN, 1998).

Assim, é importante deixar claro que esta ferramenta sozinha não é capaz de melhorar o processo. Ela apenas orienta os engenheiros de *software* para os principais problemas que podem existir. Cabe a eles, analisar os problemas e modificar a forma de trabalhar a fim de tentar corrigir suas deficiências e aumentar seu desempenho. Somente com a ajuda das pessoas envolvidas que a melhoria de processos de *software* pode ser efetivamente alcançada.

Capítulo 6
Considerações Finais

Neste capítulo, são apresentados o processo de realização desta tese, suas principais conclusões e as perspectivas futuras para a continuidade deste trabalho.

6.1 Conclusões

A definição e a utilização de processos de *software* é um requisito básico para o alcance de índices mais altos de qualidade para os produtos desenvolvidos, característica esta que tem se mostrado cada vez mais necessária para a sobrevivência das empresas produtoras de *software* no competitivo mercado atual. No entanto, não basta somente definir e utilizar um processo, é preciso que este seja constantemente monitorado, avaliado e melhorado.

Para alcançar esta melhoria contínua, chegou-se à conclusão que o uso de dados quantitativos pode proporcionar aos engenheiros de *software* um detalhamento das características do processo que de outra forma não seriam observadas. Somente a mensuração provê os dados necessários sobre projetos, processos e produtos, levando em consideração as características específicas da organização e as metas de negócio a apoiar.

Baseado neste panorama, este trabalho apresentou uma abordagem para a seleção, definição e uso de métricas como ferramenta para orientar o procedimento de melhoria de processos de *software*. A partir desta abordagem teórica, foi desenvolvido um sistema para apoiar os procedimentos de medição e avaliação de processos definidos e instanciados pela Estação TABA, projeto no qual este trabalho está inserido. A etapa de análise dos resultados da medição está sendo apoiada por um sistema baseado em conhecimento capaz de inferir possíveis problemas existentes no processo utilizado.

O trabalho foi realizado em várias etapas cada uma delas com objetivos bem definidos e com publicações descrevendo seus resultados. Na primeira etapa, foi realizado um estudo sobre medições em processos de *software* e foram selecionadas as métricas que seriam utilizadas neste trabalho (GOMES *et al.*, 2000a). A segunda etapa consistiu na utilização das métricas e avaliação do processo de *software* de uma empresa durante um projeto (GOMES *et al.*, 2000b). A terceira etapa foi a especificação

e o desenvolvimento do sistema de apoio à avaliação de processos de *software* (GOMES *et al.*, 2001a; GOMES *et al.*, 2001b, GOMES *et al.*, 2001c).

Durante o decorrer deste trabalho, foram observados alguns pontos que devem ser mencionados. A fase inicial de implantação de um programa de medição, isto é, a seleção e definição das métricas, deve estar baseada nos principais objetivos da organização. Isto aumenta o comprometimento da alta gerência com o programa e este apoio pode ser definido como um dos fatores mais importantes, se não o mais importante, para o sucesso de um programa como este. Esta é a garantia de que todos os passos do procedimento serão corretamente seguidos pela equipe no decorrer do projeto.

Além disso, o conjunto selecionado de métricas deve ser simples para facilitar o seu uso, porém significativo o bastante para apoiar a tomada de decisão. Métricas isoladas podem apresentar uma visão distorcida do processo levando a interpretações equivocadas. Acredita-se que o seu uso de forma combinada é uma ferramenta poderosa ainda pouco explorada nas abordagens tradicionais.

Definidas as métricas, é importante que a obtenção dos dados seja feita durante o decorrer do projeto, pois, assim, pode-se construir uma base capaz de permitir a avaliação de cada passo do ciclo de vida. Para tornar possível a melhoria do processo, é necessário que cada uma de suas etapas seja perfeitamente entendida e melhorada.

No entanto, coletar corretamente estas informações é uma tarefa difícil. As equipes de desenvolvimento normalmente não estão habituadas ao procedimento de medição e, assim, qualquer problema encontrado durante o projeto se torna motivo para que este procedimento seja prejudicado ou, até mesmo, interrompido. Assim, tornar a medição parte do processo produtivo é um fator muito importante para diminuir o esforço gasto e garantir que as informações analisadas retratem realmente a realidade vivida no decorrer do projeto. Motivo este que levou à implementação do sistema para apoiar o levantamento destes dados de forma semi-automatizada.

E, finalmente, deve-se ter sempre em mente que ferramentas, procedimentos ou metodologia não são suficientes se as pessoas envolvidas com o seu uso não estiverem realmente engajadas com a finalidade principal de uma tecnologia que é desenvolver produtos de boa qualidade, atendendo aos requisitos estabelecidos, dentro do cronograma e orçamentos propostos.

6.2 Perspectivas Futuras

A fim de expandir esta abordagem e, assim, torná-la mais completa, algumas melhorias já podem ser destacadas para trabalhos futuros.

Inicialmente, um primeiro passo seria obter o conhecimento de especialistas para o desenvolvimento de uma estrutura semelhante àquela apresentada na figura 4.8 para o segundo e terceiro objetivos definidos nesta abordagem. A conclusão desta etapa é fundamental para permitir uma melhor visualização das semelhanças e diferenças entre as estruturas de decisão, o que pode levar a novas conclusões que somente neste momento poderão ser observadas.

Depois de terminada esta etapa de finalização desta abordagem, pode-se dar início a sua real expansão. Durante a construção da base de conhecimento do sistema, tomou-se o cuidado de selecionar as informações que pudessem ser utilizadas por uma variedade maior de projetos. Desta forma, não foram incluídas, por exemplo, regras que pudessem ser específicas de um determinado domínio ou empresa. Contudo, este ganho em generalidade implicou em uma sensível perda de poder de análise, pois regras que não pudessem ser utilizadas em projetos com determinadas características, mesmo que pertinentes para a maioria, não poderiam fazer parte do conhecimento.

Com o intuito de resolver este problema e melhorar a qualidade das inferências, pode-se prever a definição de mais dois grupos de métricas a serem utilizadas. O primeiro grupo deveria estabelecer "tipos" de projetos com características similares e o segundo seria composto de métricas que se pretende estudar melhor seu comportamento e relacionamento com as demais. Assim, deveria ser construída uma base de dados de métricas contendo dados obtidos em diferentes projetos de forma a tornar possível a realização de estudos empíricos comparando os resultados obtidos nos diferentes projetos de um mesmo "tipo". À medida que esta base aumentar, relações entre métricas poderão se mostrar pertinentes e poderão orientar o estabelecimento de novas regras a serem inseridas na base de conhecimento do SBC. Assim, conhecimentos específicos de determinadas áreas poderão compor a base do sistema melhorando a qualidade das inferências e permitindo que respostas cada vez mais precisas possam ser obtidas. Além disso, acredita-se que somente com a comparação dos resultados obtidos em vários projetos similares para a execução de um mesmo processo que será possível determinar modificações pertinentes para a sua definição.

A técnica utilizada para a definição dos parâmetros de tolerância no procedimento de avaliação também pode ser melhorada. Foi constatado que, inicialmente, os gerentes têm muita dificuldade em estabelecer estes limites, apesar de toda atenção dada a esta etapa (o que incluiu varias discussões e o levantamento de dados na literatura como forma de orientação). Seria interessante o uso de técnicas que permitissem o estabelecimento destes valores automaticamente pelo sistema. Para isso, pode-se imaginar o uso de técnicas como, por exemplo, o uso do Controle Estatístico do Processo (FLORAC e CARLETON, 1999). Desta forma, o sistema poderá analisar o comportamento padrão das métricas em projetos de um mesmo tipo e, assim, determinar os valores de tolerância a partir de uma análise estatística dos dados destes projetos retirando do usuário do sistema esta difícil tarefa. O uso de gráficos para a apresentação dos resultados das medições pode ser de grande ajuda para facilitar a visualização e o entendimento das tolerâncias estipuladas e dos problemas sugeridos pelo SBC.

Um outro aspecto importante que precisa ser destacado é que neste trabalho procurou-se implementar um conjunto de funcionalidades bastante simples apenas com o objetivo de registrar os dados necessários para o procedimento de avaliação. Neste contexto, já existem pesquisas em andamento que têm como objetivo: (i) desenvolver uma ferramenta para efetivamente apoiar a determinação de estimativas de esforço e cronograma para os projetos baseada no uso de métodos de estimativas; (ii) uma ferramenta para apoiar as avaliações de qualidade e, assim, registrar o resultado de reuniões de inspeção e *walkthrought* provendo como resultado o número de erros encontrados em cada artefato; (iii) uma ferramenta de apoio à alocação de pessoal em projetos a partir de sua experiência.

Finalmente, seria interessante que esta abordagem fosse utilizada em um número maior de projetos de forma a verificar a sua viabilidade. O uso dessa abordagem em experiências reais pode ser de grande ajuda para encontrar falhas, deficiências e melhorias para o aprimoramento destas idéias.

Referências Bibliográficas

ARAÚJO, M. A. P.,1998, *Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados pela Estação TABA*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BASILI, V. R., CALDIERA, G., ROMBACH, H.D., 1994, *Goal Question Metric Paradigm*, Encyclopedia of Software Engineering, 2 Volume Set, John Wiley & Sons, Inc.

BASILI, V. R., ROMBACH, H. D., 1981, "The TAME project: Towards Improvement-oriented Software Environments", *IEEE Transactions on Software Engineering*, 14(6), pp. 758-773.

BAUMERT J.H., 1992, *Software Measures and the Capability Maturity Model*, CMU/SEI-92-TR-25, ESD-TR-92-25, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

BROOKS, F.P.J., 1975, *The Mythical Man-Month*, Addison-Wesley

CARLETON, A.D., PARK, R.E, GOETHERT, W.B., FLORAC, W.A., BAILEY, E.K., PFLEEGER, S.L., 1992, *Software Measurement for DoD Systems: Recommendation for Initial Core Measures*, CMU/SEI-92-TR-19, ESC-TR-92-19, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

CONTE, S.D., DUNSMORE, H.D., SHEN, V.Y., 1986, *Software Engineering Metrics and Models*, Benjamin-Cummings, Menlo Park, CA.

DASKALANTONAKIS, M.K., 1992, "A Practical View of Software Measurement and Implementation Experiences Within Motorola", In: *Applying Software Metrics*, IEEE Computer Society Press, pp. 168-180.

DAVIES, E., WHYMAN, M., 2000, "ISO 9000:2000 new ISO, New Responsibilities for Top Management", *Engineering Management Journal*, pp. 244-248, Outubro.

EMAM, K. E., DROUIN, J. N., MELO, W., 1998, *SPICE – The Theory and Practice of Software Process Improvement and Capability Determination*, California, IEEE Computer Society.

EMAM, K.E., MADHAVJI, N.H., 1999, *Elements of Software Process Assessment & Improvement*, California, IEEE Computer Society.

FALBO, R. A., 1998, *Integração de Conhecimento em um Ambiente de Desenvolvimento*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FENTON, N.E., PFLEEGER, S.L, 1997, *Software Metrics – A Rigorous & Practical Approach*, Second Edition, Boston, MA, PWS Publishing Company.

FENTON, N.E., NEIL, M., 2000, "Software Metrics: Roadmap", In: *The Future of Software Engineering*, ACM Press, pp. 359-370.

FLORAC, W.A., 1992, *Software Quality Measurement: A Framework for Counting Problems, Failures and Faults*, CMU/SEI-92-TR-22, ESC-TR-92-22, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

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

FLORAC, W.A., PARK, R.E., CARLETON, A.D., 1997, *Practical Software Measurement: Measuring for Process Management and Improvement*, CMU/SEI-97-HB-003, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

FUGGETTA, A., 2000, "Software Process: A Roadmap", In: *The Future of Software Engineering*, ACM Press, pp. 25-33.

GOETHERT, W.B., BAILEY, E.K., BUSBY, M.B., 1992, *Software Effort & Schedule Measurement: A Framework for Counting Staff-hours and Reporting Schedule Information*, CMU/SEI-92-TR-21, ESC-TR-92-21, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

GOMES, A.G. J. *et al.*, 2000b, "Medição e Melhoria de Processos de Software", In: *WQS 2000 – Workshop de Qualidade de Software – XIV SBES*, pp. 306-316, João Pessoa, Brasil.

GOMES, A.G. J., MACHADO, L.F.C., OLIVEIRA, K.M., ROCHA, A.R., 2000a, "Software Process Improvement Through Measurement and Expert Judgement", In: *FESMA – The Third European Software Measurement Conference – AEMES 2000*, Madrid, Espanha.

GOMES, A.G. J., OLIVEIRA, K.M., ROCHA, A.R., 2001a, "Métricas para Avaliação e Melhoria de Processo", In: *QuaTIC'2001 – Quarto Encontro para a Qualidade nas Tecnologias de Informação e Comunicações*, pp. 71-77, Lisboa, Portugal.

GOMES, A.G. J., OLIVEIRA, K.M., ROCHA, A.R., 2001b, "Avalaição de Processos de Software Baseada em Medições", In: *WQS 2001 – Workshop de Qualidade de Software – XV SBES*, Rio de Janeiro, Brasil.

GOMES, A.G. J., MAFRA S. N., OLIVEIRA, K.M., ROCHA, A.R., 2001c, "Avalaição de Processos de Software na Estação Taba", In: *Workshop de Ferramentas – XV SBES*, Rio de Janeiro, Brasil.

GRABLE, R., JERNIGAN, J., POGUE, C., DIVIS, D., 1999, "Metrics for Small Projects: Experiences at the SED", *IEEE Software*, pp.21-29, Março/Abril.

GRADY, R.B., CASWELL, E.L., 1987, *Software Metrics: Establishing a Company-Wide Program*, New Jersey, Prentice Hall.

HUMPHREY, W.S., 1997, *Introduction to the Personal Software Process*, Addison-Wesley.

HUMPHREY, W.S. 1989, *Managing the Software Process*, Addison-Wesley

KAUTZ, K., 1999, "Making Sense of Measurement for Small Organizations", *IEEE Software*, pp. 14-20, Março/Abril.

LAVAZZA, L., 2000, "Providing Automated Support for the GQM Measurement Process", *IEEE Software*, pp. 56-62, Maio/Junho.

MACHADO, L.F.C. *et al*, 2000b, *Modelo para Definição, Especialização e Instanciação de Processos de software na Estação TABA*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MACHADO, L.F.C. *et al*, 2000a, "DEF-PRO: Uma Ferramenta para Apoiar a Definição do Processo Padrão", In: *XI CITS*, pp. 166-179, Curitiba, Brasil.

MAIDANTCHIK, C. L. L. M., 1999, *Gerência de Processos de Software para Equipes Geograficamente Dispersas*, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ.

MACKEY, K., 2000, "Mars versus Vênus", *IEEE Software*, pp. 14-15, Maio/Junho.

ISO/IEC 12207, 1995, *Information Technology – Software Life-Cycle Processes.*

IEEE P1045/D5.0, 1992, *Standard for Software Productivity Metrics [draft].*

OLIVEIRA, K. M., 1999, *Modelo para Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OLIVEIRA, K.M, ROCHA, A.R, TRAVASSOS, G., MENEZES, C., 1999, "Using Domain-Knowledge in Software Development Environments", In: *SEKE 99 - Software Engineering and Knowledge Engineering*; Kaiserlautern, Germany.

OMAN P., PFLEEGER S.L., 1997, *Applying Software Metrics*, IEEE Computer Society Press, California.

PARK, R.E., 1992, *Software Size Measurement: A Framework for Counting Source Statements*, CMU/SEI-92-TR-20, ESC-TR-92-20, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

PARK, R.E., GOETHERT, W.B., FLORAC, W.A., 1996, *Goal-Driven Software Measurement – A Guidebook*, CMU/SEI-96-HB-002, Pittsburgh, Software Engineering Institute, Carnegie Mellon University.

PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds), 1995, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley.

PBQP, 1999, *Qualidade e Produtividade no Setor de Software Brasileiro*, Ministério da Ciência e Tecnologia, Secretaria de Política de Informática e Automação.

PRESSMAN, R. S., 1997, *Software Engineering: A Practitioner's Approach*, 4th edition, McGraw-Hill.

ROBERTS, M.A., 1996, *Experiences in Analyzing Software Inspection Data*, Software Engineering Process Group, McDonnell Douglas Aerospace, McDonnell Douglas Corporation, St. Louis, MO.

ROCHA, A. R. C., AGUIAR, T. C., SOUZA, J. M., 1990, "TABA: A Heuristic Workstation for Software development", In: *Proceedings of COMPEURO 90*, Tel Aviv, Israel, Maio.

ROCHA, A.R., MALDONADO, J.C., WEBER, K.C., 2001, *Qualidade de Software – Teoria e Prática*, 1ª ed., Prentice Hall, São Paulo.

TAJIMA, D., MATSUBARA, T., 1981, "The Computer Software Industry in Japan", IEEE Computer, 14(5), pp. 89-96.

TRAVASSOS, G. H., 1994, *O Modelo de Integração de Ferramentas da Estação TABA*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

VOB, A., KARBAH,W., 1993, *Impementing KADS Expertise Models with Model-K*, IEEE Expert, 74 – 81.

WANGENHEIM, C.G., RODRIGUES, M.R., 2000, "Planejamento de Programas de Mensuração Baseado em Reutilização", In: *XI CITS*, pp. 44-57, Curitiba, Brasil.

Weber, K.C., Rocha, A.R.C., Nascimento, C.J., 2001, *Qualidade e Produtividade em Software*, 4ª ed., São Paulo, Makron Books.

WELLER, E.F., 2000, "Practical Applications of Statistical Process Control", *IEEE Software*, pp. 48-55, Maio/Junho.

WERNECK, V. M., 1995, *Ambiente para Desenvolvimento de Software Baseado em Conhecimento*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ZAHRAN, S., 1998, *Software Process Improvement*, Addison Wesley.

ZLOT, F., SANTOS, G., 1999, *Definição e Instanciação e Ambientes na Estação TABA*. Projeto Final de Curso, UFRJ, Rio de Janeiro, RJ, Brasil.

Anexo 1

Métricas Utilizadas

Neste anexo, são apresentadas as métricas selecionadas para a análise do processo com suas respectivas definições e formas de cálculo.

Métrica 1	**Tempo real de todo o projeto**
Cálculo	data de sua aceitação – data de seu início
Definição	Foi definido que o tempo seria medido em número de dias corridos do início de uma atividade até a data da sua aceitação. Assim, o tempo real de todo o projeto é o número de dias desde o começo dos trabalhos até a aceitação de sua última atividade (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 2	**Tempo real por macro-atividade**
Cálculo	data de sua aceitação – data de seu início
Definição	De forma semelhante, foi definido que o tempo real de cada macro-atividade é medido em número de dias corridos do início da sua execução até a data de sua aceitação (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 3	**Tempo estimado para o projeto**
Cálculo	data prevista para sua aceitação – data prevista para seu início
Definição	O tempo estimado é o número de dias corridos entre a data prevista para o início de sua primeira macro-atividade até a data da previsão de término da última macro-atividade (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 4	**Tempo estimado por macro-atividade**
Cálculo	data prevista para sua aceitação – data prevista para seu início
Definição	De forma semelhante, foi definido que o tempo estimado por macro-atividade é o número de dias corridos entre a data prevista para seu inicio e a data prevista para o seu término (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 5	**Precisão da estimativa de cronograma em todo o projeto**
Cálculo	Tempo real de todo o projeto (M1) / tempo estimado para o projeto (M3) (DASKALANTONAKIS, 1992).
Escala	Racional

Métrica 6	**Precisão da estimativa de cronograma por macro-atividade**
Cálculo	Tempo real por macro-atividade (M2) / tempo estimado por macro-atividade (M4) (DASKALANTONAKIS, 1992).
Escala	Racional

Métrica 7	**Esforço real de todo o projeto**
Cálculo	somatório do número de homens-hora empregados durante todo o projeto
Definição	Foi definido que o esforço seria medido em número de homens-hora empregados na execução de uma atividade. Assim, o esforço real de todo o projeto é o somatório do número de horas de trabalho de cada membro da equipe desde o início dos trabalhos até a sua data de aceitação (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 8	**Esforço real por macro-atividade**
Cálculo	somatório do número de homens-hora empregados por macro-atividade
Definição	De forma semelhante, foi definido que o esforço por macro-atividade é o somatório do número de horas de trabalho de cada membro da equipe na execução desta macro-atividade até a sua data de aceitação (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 9	**Esforço estimado para o projeto**
Cálculo	número de homens-hora estimados para todo o projeto
Definição	Foi definido que o esforço estimado é o número de homens-hora previsto para todo o projeto (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 10	**Esforço estimado por macro-atividade**
Cálculo	número de homens-hora empregados por macro-atividade
Definição	De forma semelhante, foi definido que o esforço estimado é o número de homens-hora previsto para a macro-atividade (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 11	Precisão da estimativa de esforço em todo o projeto
Cálculo	esforço real de todo o projeto (M7) / esforço estimado para o projeto (M9) (DASKALANTONAKIS, 1992).
Escala	Racional

Métrica 12	Precisão da estimativa de esforço por macro-atividade
Cálculo	esforço real por macro-atividade (M8) / esforço estimado por macro-atividade (M10) (DASKALANTONAKIS, 1992).
Escala	Racional

Métrica 13	Experiência da equipe no domínio da aplicação
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 14	Experiência da equipe no domínio da aplicação.
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 15	Experiência da equipe na ferramenta de desenvolvimento.
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 16	Experiência da equipe na linguagem de programação.
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 17	Experiência da equipe no método de desenvolvimento.
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 18	Experiência da equipe no processo de *software*
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 19	Experiência da equipe em projetos de porte similar
Cálculo	mediana dos valores obtidos para cada membro da equipe.
Definição	Para diminuir a subjetividade desta métrica foram estabelecidos 5 valores possíveis que são: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 20	Rotatividade de Pessoal
Cálculo	somatório do número de pessoas que saíram ou entraram no decorrer do projeto / número total de pessoas que participaram do projeto.
Definição	A rotatividade de pessoal é a relação entre o somatório do número de pessoas que entraram ou saíram durante os trabalhos e o número total de pessoas que trabalharam no projeto (FENTON e PFLEEGER, 1997).
Escala	Absoluta

Métrica 21	Esforço em retrabalho
Cálculo	somatório do número de homens-hora empregados em retrabalho durante todo o projeto.
Definição	Foi definido que retrabalho é qualquer atividade de correção, modificação ou melhoria realizada sobre um artefato que já deveria ter sido finalizado. Assim, não é considerado retrabalho, por exemplo, os problemas encontrados por um programador durante o desenvolvimento de um módulo ou função, pois este ainda não foi validado e considerado finalizado. Somente após a realização de uma atividade de avaliação do documento, feita, de preferência, por outra pessoa que não o próprio desenvolvedor, que qualquer atividade realizada para sua correção, modificação ou melhoria será considerada retrabalho. Assim, esforço em retrabalho é o somatório do número de horas empregadas em retrabalho por cada membro da equipe durante todo o projeto (PARK *et al.*, 1996).
Escala	Racional

Métrica 22	Esforço em retrabalho por macro-atividade
Cálculo	somatório do número de homens-hora empregados em retrabalho durante uma macro-atividade.
Definição	Esforço em retrabalho em uma macro-atividade é o somatório do número de horas empregadas no retrabalho nesta macro-atividade por cada membro da equipe durante todo o projeto (PARK *et al.*, 1996).
Escala	Racional

Métrica 23	Percentual de retrabalho
Cálculo	esforço em retrabalho (M21) * 100 / esforço real de todo o projeto (M7) (PARK *et al.*, 1996).
Escala	Racional

Métrica 24	Percentual de retrabalho por macro-atividade
Cálculo	esforço em retrabalho por macro-atividade (M22) * 100 / esforço real na macro-atividade (M8) (PARK *et al.*, 1996).
Escala	Racional

Métrica 25	Número de erros por artefato
Cálculo	somatório do número de erros em cada artefato produzido durante o projeto encontrados em reuniões de revisão de aprovação do artefato em questão.
Definição	Foi definido que erro é qualquer problema novo encontrado em um artefato no momento da sua aprovação (ROBERTS, 1996).
Escala	Absoluta

Métrica 26	Número de modificações por artefato por macro-atividade
Cálculo	somatório do número de modificações em cada artefato produzido durante o projeto encontrados após sua aceitação, organizadas por macro-atividade.
Definição	Quando um novo problema é encontrado em um artefato que já tenha sido aprovado anteriormente, é necessário modificá-lo para que este atenda corretamente aos requisitos estipulados. Sempre que isto for necessário, deve-se documentar as modificações realizadas de forma a tornar possível sua posterior contagem de forma semelhante à feita com os erros. O número de modificações por artefato é sub-totalizado por cada macro-atividade onde estas modificações foram encontradas (ROBERTS, 1996).
Escala	Absoluta

Métrica 27	Tamanho do sistema
Cálculo	número total de linhas de código exceto as contendo somente comentários ou linhas em branco.
Definição	Esta sendo utilizado o número total de linhas de código exceto as contendo comentários ou linhas em branco para facilitar a medição do tamanho do sistema (CARLETON *et al.*, 1992).
Escala	Racional

Métrica 28	Densidade de defeitos
Cálculo	(número de erros por artefato (M25) + número de modificações por artefato por macro-atividade (M26)) / tamanho do sistema(M27).
Definição	Soma-se o número de erros de todos os artefatos ao número de modificações de todos os artefatos em todas as macro-atividades e compara-se com o tamanho do sistema (atualmente medido em linhas de código) (PARK *et al.*, 1996).
Escala	Racional

Métrica 29	Esforço empregado na correção de problemas após a liberação do *software*
Cálculo	somatório do número de homens-hora empregados para a correção de problemas encontrados no *software* após sua aceitação por parte do usuário.
Definição	Foi definido que qualquer trabalho realizado após a liberação do *software* para o usuário necessário devido a problemas encontrados deve ser contado como esforço empregado na correção de problemas após a liberação do *software*. Isto não inclui manutenções evolutivas (TAJIMA e MATSUBARA, 1981).
Escala	Racional

Métrica 30	Deterioração do *Software*
Cálculo	esforço empregado na correção de problemas após a liberação do *software* (M29) / esforço real de todo o projeto.
Definição	Acredita-se que modificações efetuadas em um *software* após a sua liberação para uso aumentam a possibilidade de inclusão de novos problemas diminuindo a qualidade do produto final (manutenibilidade, eficiência, etc).Desta forma, foi definido Deterioração do *software* como a relação entre o esforço empregado na correção de defeitos após a liberação do *software* e o esforço real de todo o projeto (TAJIMA e MATSUBARA, 1981).
Escala	Racional

Métrica 31	Forma da Estimativa
Cálculo	o número de 1 a 3 de acordo com a definição da métrica
Definição	A forma da estimativa deve ser um dos seguintes valores: (i) adhoc; (ii) baseada na experiência de projetos anteriores; (iii) método formal (FENTON e PFLEEGER, 1997).
Escala	Nominal

Métrica 32	Característica do Cronograma
Cálculo	o número de 1 a 5 de acordo com a definição da métrica
Definição	A característica do cronograma deve ser um dos seguintes valores: (i) cronograma flexível; (ii) cronograma adequado, mas pouco flexível; (iii) cronograma apertado, mas sem pressões; (iv) muita pressão de gerentes e usuários mas sem implicações no processo; (v) muita pressão implicando no não seguimento do processo (FENTON e PFLEEGER, 1997).
Escala	Ordinal

Métrica 33	Característica Geral dos Usuários
Cálculo	O número de 1 a 4 de acordo com a definição da métrica
Definição	A características dos usuários deve ser um dos seguintes valores: (i) usuário não participa adequadamente (com relação ao tempo); (ii) usuários escolhidos não adequados, pois desconhecem o problema; (iii) falta de consenso entre os usuários; (iv) usuário adequado e participativo (FENTON e PFLEEGER, 1997).
Escala	Nominal

Métrica 34	Participação dos Usuários nas Avaliações
Cálculo	o número de 1 a 3 de acordo com a definição da métrica
Definição	A participação dos usuários nas avaliações deve ser um dos seguintes valores: (i) usuários não se prepararam devidamente para as reuniões; (ii) usuários não participaram devidamente das reuniões; (iii) usuários participaram devidamente das reuniões (FENTON e PFLEEGER, 1997).
Escala	Nominal

Anexo 2

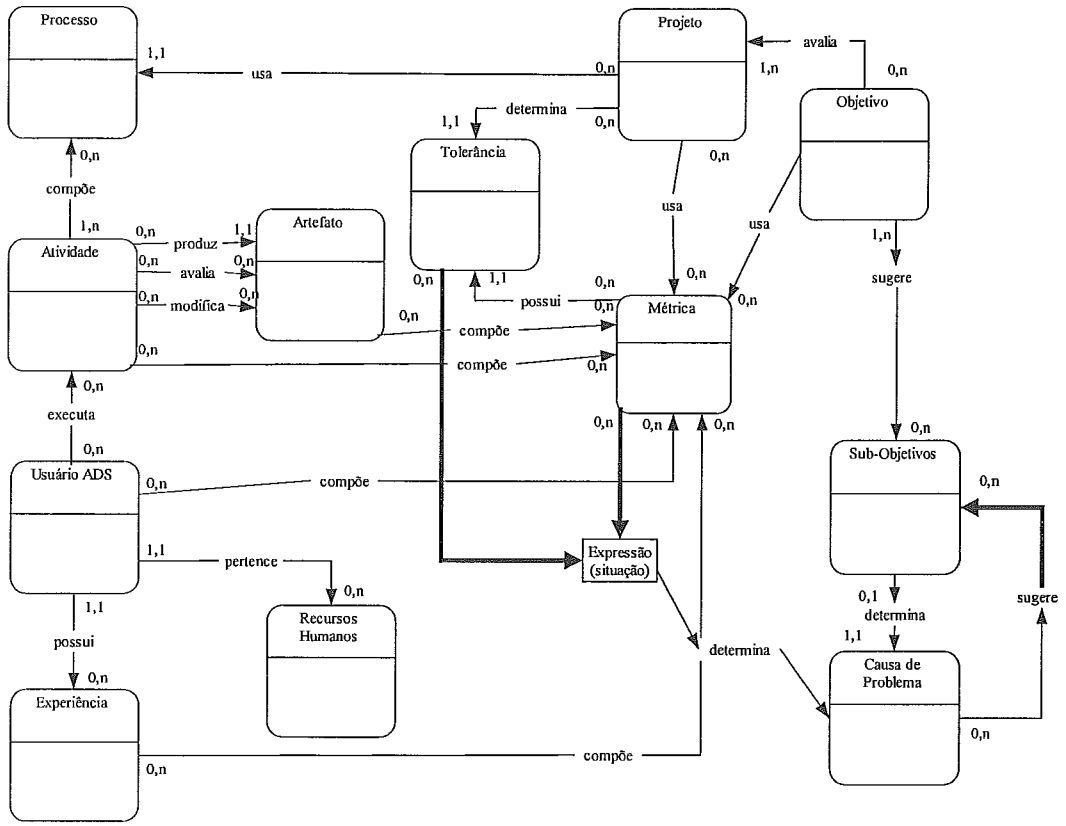
Modelagem do Sistema Baseado em Conhecimento

Neste anexo, são apresentados todos os modelos gerados através do uso do método KADS-estendido para o desenvolvimento do sistema baseado em conhecimento.

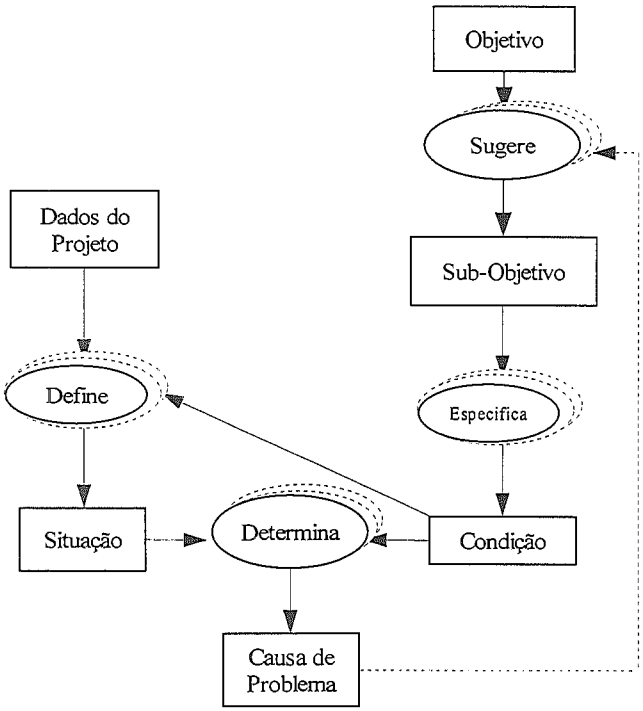
A modelagem do sistema baseado em conhecimento foi realizada utilizando-se o método KADS-estendido proposto por WENECK (1995) que, como o próprio nome diz, trata-se de uma extensão ao método KADS (*Knowledge Aquisition and Design Structure*) (Vob e Karbach, 1993). Esta extensão teve como objetivo diminuir as dificuldades encontradas para a realização do projeto do sistema a partir de sua especificação e de sua passagem para a etapa de implementação.

Como qualquer outro método de desenvolvimento de sistemas, este possui um conjunto de modelos que podem ser utilizados a fim de diminuir a distância entre a concepção do problema a ser solucionado e a sua implementação computacional. Considerando a equipe reduzida e o pequeno escopo definido para o projeto, foram selecionados apenas os três principais modelos propostos pelo método para documentar o processo de análise do conhecimento: o modelo de Especialidade, o modelo Lógico e apenas um diagrama do modelo Físico. Desta forma, neste anexo são apresentados apenas os modelos que foram utilizados para a construção do sistema baseado em conhecimento para a avaliação de processos de *software* baseada na estrutura de decisão apresentada no capítulo 4.

ESTRUTURA DO DOMÍNIO DO PROBLEMA



ESTRUTURA DE INFERÊNCIA



ESTRUTURA DE TAREFAS

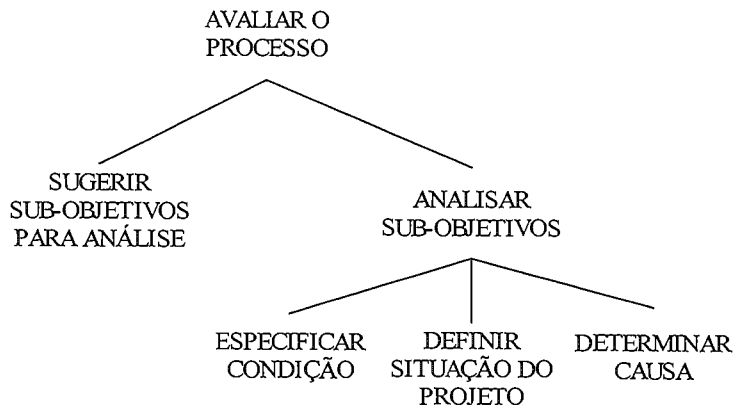
Para o OBJETIVO informado :

Sugerir os SUB-OBJETIVOS a serem analisados.

Enquanto existir SUB-OBJETIVOS não analisado faça:

- Especificar a CONDIÇÃO para a análise deste SUB-OBJETIVO.
- Definir SITUAÇÃO a partir da CONDIÇÃO e dos DADOS.
- Se a SITUAÇÃO atende à CONDIÇÃO e então:
  - Determinar CAUSA.

ÁRVORE DE TAREFAS



CLASSE	DESCRIÇÃO
Objetivo	Objetivos de avaliação de processo
Sub-objetivos	Sub-objetivos derivados do objetivo principal para avaliar um processo. São as possíveis causas que devem ser analisadas para um objetivo.
Condição	Condição que determina se o Sub-objetivo é uma Causa de Problema
Dados do projeto	Dados coletados do projeto sobre o qual se deseja avaliar o processo
Situação	Situação em que se encontra o processo. Combina o valor coletado para métricas e tolerâncias estabelecidas na Condição.
Causa de problema	Causa do problema encontrado no Objetivo

INFERÊNCIA	DESCRIÇÃO
Sugere	Subdivisão do Objetivo em seus Sub-objetivos.
Especifica	Especificação da Condição para a análise do Sub-objetivo. Se esta condição for verificada então o Sub-Objetivo é uma Causa de Problema.
Define	Coleta dos Dados do Projeto para comparar com a Condição.
Determina	Comparação do valor coletado do projeto com a Condição. A partir desta comparação determina se o Sub-objetivo é realmente Causa de Problema.

DIAGRAMA HEURÍSTICO DO RACIOCÍCIO

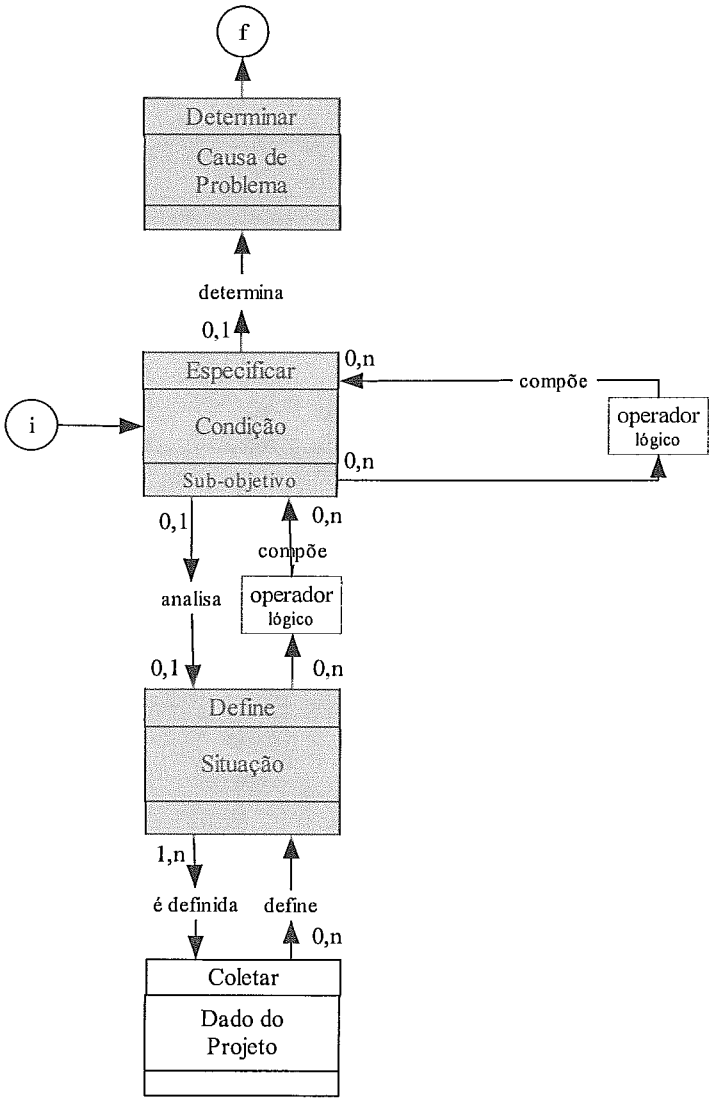


DIAGRAMA DO DOMÍNIO DO PROBLEMA

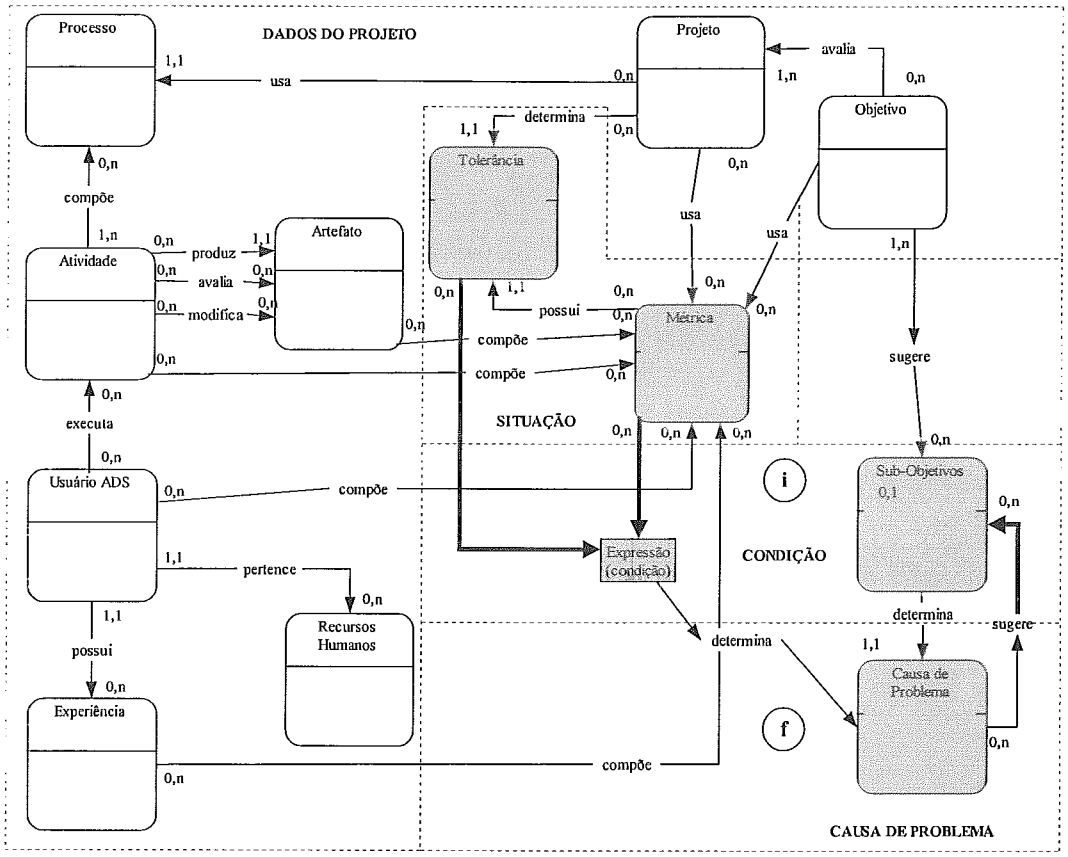
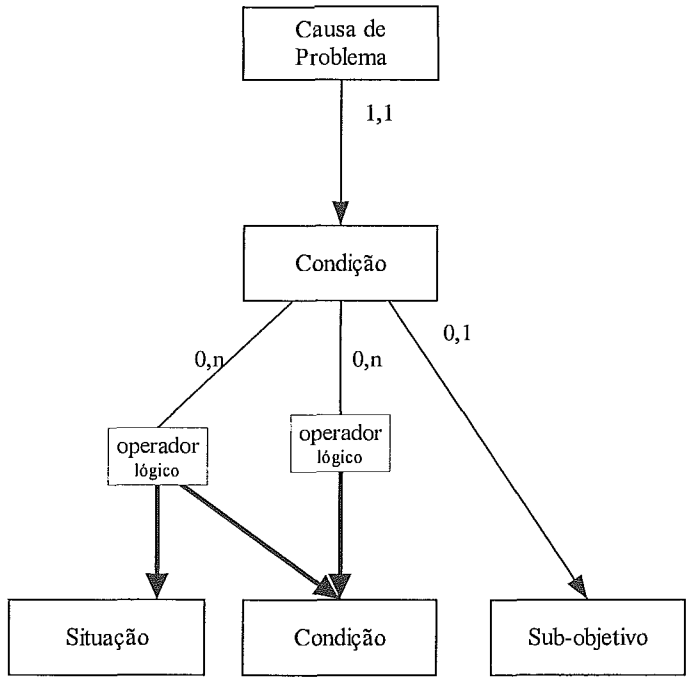


DIAGRAMA ESTRUTURAL DA BASE DE CONHECIMENTO



Regras:

Se Sub-objetivo
Então Condição

Se Condição
Então Causa de Problema

Se Situação (operador lógico) Condição
Então Condição

Se Condição (operador lógico) Condição
Então Condição

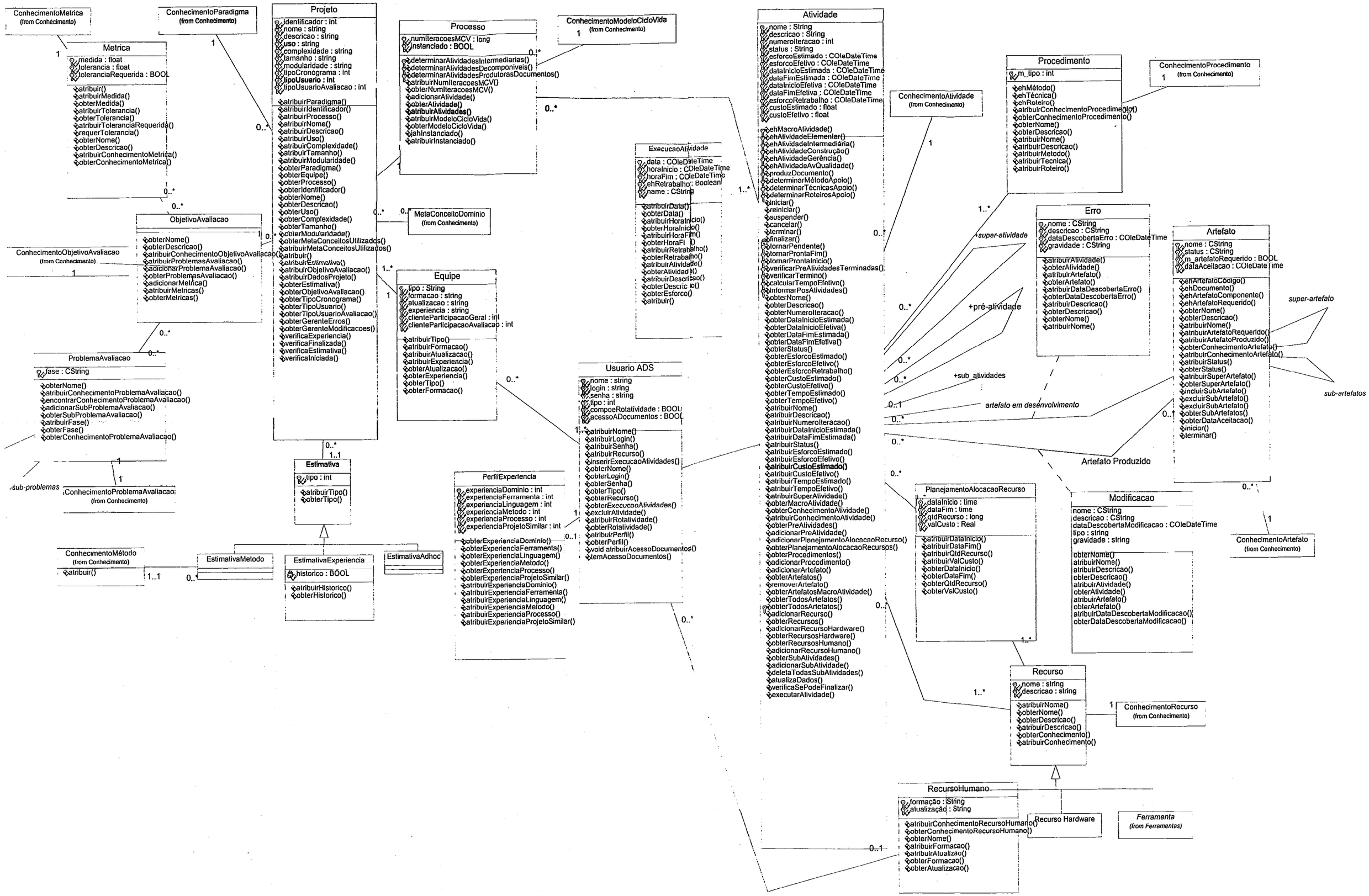
Anexo 3

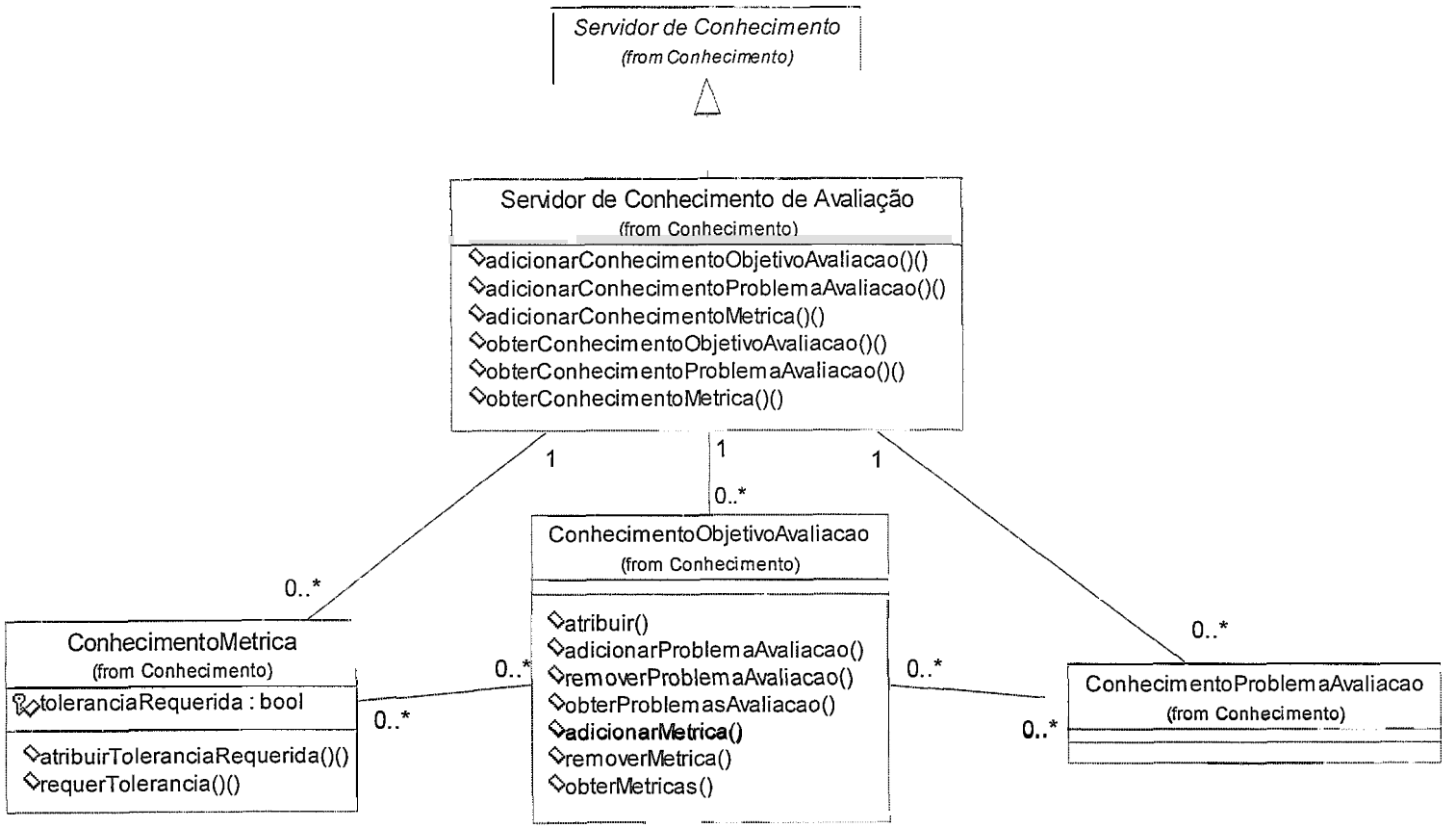
Alterações no Modelo TABA

Este anexo apresenta os modelos de classes da Estação TABA com as extensões incorporadas pelo sistema proposto nesta tese.

CLASSE	DESCRIÇÃO
ServidordeConhecimentodeAvaliação	Servidor de conhecimento responsável por armazenar o conhecimento necessário para a avaliação de processos.
ConhecimentoObjetivoAvaliação	Objetivos de avaliação que podem ser utilizados na análise do processo. São os objetivos definidos com os especialistas nesta abordagem.
ObjetivoAvaliação	Objetivo da avaliação utilizado para avaliar um processo específico.
ConhecimentoMetrica	Métricas que são utilizadas na avaliação de processos segundo algum dos objetivos definidos.
Métrica	Métrica coletada para a avaliação do processo segundo o objetivo escolhido.
ConhecimentoProblemaAvaliação	Todos os problemas que podem ser apontados como causas para o desvio ocorrido em algum dos objetivos definidos.
ProblemaAvaliação	Estrutura que será a solução do problema de avaliação do processo segundo o objetivo escolhido. Esta será a árvore que será apresentada para o usuário com as causas para o desvio ocorrido no objetivo.
UsuárioAds	Usuário do ambiente instanciado. Inclui tanto o Super Usuário quanto os desenvolvedores e futuros usuários que não participam diretamente do projeto, como, por exemplo, os clientes.

CLASSE	DESCRIÇÃO
Desenvolvedores	Usuário do sistema que participa do desenvolvimento do projeto.
PerfilExperiencia	Classe que é utilizada para representar a experiência de cada desenvolvedor da equipe do projeto.
ExecuçãoAtividade	Relacionamento existente entre os Desenvolvedores e as Atividades armazenando as informações sobre a sua execução.
FormaEstimativa	Forma como as estimativas de projeto foram realizadas.
EstimativaAdhoc	Estimativa realizada de forma adhoc.
EstimativaExperiencia	Estimativa realizada baseada na experiência dos gerentes e responsáveis pelo projeto. Pode ser feita baseada em dados históricos ou não.
EstimativaMetodo	Estimativa feita utilizando algum tipo de método formal de estimativa de projeto.
Erro	Erro encontrado em um artefato durante uma atividade do tipo avaliação
Modificação	Modificações encontradas em artefatos já validados em atividades anteriores.


ConhecimentoMetrica
(from Conhecimento)
ConhecimentoParadigma
(from Conhecimento)
Metrica
medida : float
tolerancia : float
toleranciaRequerida : BOOL
atribuir()
atribuirMedida()
obterMedida()
atribuirTolerancia()
obterTolerancia()
atribuirToleranciaRequerida()
requerTolerancia()
obterNome()
obterDescricao()
atribuirConhecimentoMetrica()
obterConhecimentoMetrica()
Projeto
identificador : int
nome : string
descricao : string
uso : string
complexidade : string
tamanho : string
modularidade : string
tipoCronograma : int
tipoUsuario : int
tipoUsuarioAvaliacao : int
atribuirParadigma()
atribuirIdentificador()
atribuirProcesso()
atribuirNome()
atribuirDescricao()
atribuirUso()
atribuirComplexidade()
atribuirTamanho()
atribuirModularidade()
obterParadigma()
obterEquipe()
obterProcesso()
obterIdentificador()
obterNome()
obterDescricao()
obterUso()
obterComplexidade()
obterTamanho()
obterModularidade()
obterMetaConceitosUtilizados()
atribuirMetaConceitosUtilizados()
atribuir()
atribuirEstimativa()
atribuirObjetivoAvaliacao()
atribuirDadosProjeto()
obterEstimativa()
obterObjetivoAvaliacao()
obterTipoCronograma()
obterTipoUsuario()
obterTipoUsuarioAvaliacao()
obterGerenteErros()
obterGerenteModificacoes()
verificaExperiencia()
verificaFinalizada()
verificaEstimativa()
verificaIniciada()
Processo
numIteracoesMCV : long
instanciado : BOOL
determinarAtividadesIntermediarias()
determinarAtividadesDecomponiveis()
determinarAtividadesProdutorasDocumentos()
atribuirNumIteracoesMCV()
obterNumIteracoesMCV()
adicionarAtividade()
obterAtividade()
atribuirAtividades()
atribuirModeloCicloVida()
obterModeloCicloVida()
jahInstanciado()
atribuirInstanciado()
ConhecimentoModeloCicloVida
(from Conhecimento)
Atividade
nome : String
descricao : String
numeroIteracao : int
status : String
esforcoEstimado : COleDateTime
esforcoEfetivo : COleDateTime
dataInicioEstimada : COleDateTime
dataFimEstimada : COleDateTime
dataInicioEfetiva : COleDateTime
dataFimEfetiva : COleDateTime
esforcoRetrabalho : COleDateTime
custoEstimado : float
custoEfetivo : float
ehMacroAtividade()
ehAtividadeElementar()
ehAtividadeIntermediaria()
ehAtividadeConstrução()
ehAtividadeGerência()
ehAtividadeAvQualidade()
produzDocumento()
determinarMétodoApoio()
determinarTécnicasApoio()
determinarRoteirosApoio()
iniciar()
reiniciar()
suspender()
cancelar()
terminar()
finalizar()
tornarPendente()
tornarProntaFim()
tornarProntaInicio()
verificarPreAtividadesTerminadas()
verificarTermino()
calcularTempoEfetivo()
informarPosAtividades()
obterNome()
obterDescricao()
obterNumeroIteracao()
obterDataInicioEstimada()
obterDataInicioEfetiva()
obterDataFimEstimada()
obterDataFimEfetiva()
obterStatus()
obterEsforcoEstimado()
obterEsforcoEfetivo()
obterEsforcoRetrabalho()
obterCustoEstimado()
obterCustoEfetivo()
obterTempoEstimado()
obterTempoEfetivo()
atribuirNome()
atribuirDescricao()
atribuirNumeroIteracao()
atribuirDataInicioEstimada()
atribuirDataFimEstimada()
atribuirStatus()
atribuirEsforcoEstimado()
atribuirEsforcoEfetivo()
atribuirCustoEstimado()
atribuirCustoEfetivo()
atribuirTempoEstimado()
atribuirTempoEfetivo()
atribuirSuperAtividade()
obterMacroAtividade()
obterConhecimentoAtividade()
atribuirConhecimentoAtividade()
obterPreAtividades()
adicionarPreAtividade()
adicionarPlanejamentoAlocacaoRecurso()
obterPlanejamentoAlocacaoRecursos()
obterProcedimentos()
adicionarProccdimento()
adicionarArtefato()
obterArtefatos()
removerArtefato()
obterArtefatosMacroAtividade()
obterTodosArtefatos()
obterTodosArtefatos()
adicionarRecurso()
obterRecursos()
adicionarRecursoHardware()
obterRecursosHardware()
obterRecursosHumano()
adicionarRecursoHumano()
obterSubAtividades()
adicionarSubAtividade()
deletaTodasSubAtividades()
atualizaDados()
verificaSePodeFinalizar()
executarAtividade()
ConhecimentoAtividade
(from Conhecimento)
Procedimento
m_tipo : int
ehMétodo()
ehTécnica()
ehRoteiro()
atribuirConhecimentoProcedimento()
obterConhecimentoProcedimento()
obterNome()
obterDescricao()
atribuirNome()
atribuirDescricao()
atribuirMetodo()
atribuirTecnica()
atribuirRoteiro()
ConhecimentoProcedimento
(from Conhecimento)
ExecucaoAtividade
data : COleDateTime
horaInicio : COleDateTime
horaFim : COleDateTime
ehRetrabalho : Boolean
name : CString
atribuirData()
obterData()
atribuirHoraInicio()
obterHoraInicio()
atribuirHoraFim()
obterHoraFi ()
atribuirRetrabalho()
obterRetrabalho()
atribuirAtividade()
obterAtividad ()
atribuirDescricao()
obterDescric io()
obterEsforco()
atribuir()
Erro
nome : CString
descricao : CString
dataDescobertaErro : COleDateTime
gravidade : CString
atribuirAtividade()
obterAtividade()
atribuirArtefato()
obterArtefato()
atribuirDataDescobertaErro()
obterDataDescobertaErro()
atribuirDescricao()
obterDescricao()
obterNome()
atribuirNome()
Artefato
nome : CString
status : CString
m_artefatoRequerido : BOOL
dataAceitacao : COleDateTime
ehArtefatoCódigo()
ehDocumento()
ehArtefatoComponente()
ehArtefatoRequerido()
obterNome()
obterDescricao()
atribuirNome()
atribuirArtefatoRequerido()
atribuirArtefatoProduzido()
obterConhecimentoArtefato()
obterConhecimentoArtefato()
atribuirStatus()
obterStatus()
atribuirSuperArtefato()
obterSuperArtefato()
incluirSubArtefato()
excluirSubArtefato()
excluirSubArtefato()
obterSubArtefatos()
obterDataAceitacao()
iniciar()
terminar()
ConhecimentoArtefato
(from Conhecimento)
ConhecimentoObjetivoAvaliacao
(from Conhecimento)
ObjetivoAvaliacao
obterNome()
obterDescricao()
atribuirConhecimentoObjetivoAvaliacao()
atribuirProblemasAvaliacao()
adicionarProblemaAvaliacao()
obterProblemasAvaliacao()
adicionarMetrica()
atribuirMetricas()
obterMetricas()
MetaConceitoDominio
(from Conhecimento)
Equipe
tipo : String
formacao : string
atualizacao : string
experiencia : string
clienteParticipacaoGeral : int
clienteParticipacaoAvaliacao : int
atribuirTipo()
atribuirFormacao()
atribuirAtualizacao()
atribuirExperiencia()
obterAtualizacao()
obterExperiencia()
obterTipo()
obterFormacao()
Usuario ADS
nome : string
login : string
senha : string
tipo : int
compoeRotatividade : BOOL
acessoADocumentos : BOOL
atribuirNome()
atribuirLogin()
atribuirSenha()
atribuirRecurso()
inserirExecucaoAtividades()
obterNome()
obterLogin()
obterSenha()
obterTipo()
obterRecurso()
obterExecucaoAtividades()
excluirAtividade()
atribuirRotatividade()
obterRotatividade()
atribuirPerfil()
obterPerfil()
void atribuirAcessoDocumentos()
temAcessoDocumentos()
ProblemaAvaliacao
fase : CString
obterNome()
atribuirConhecimentoProblemaAvaliacao()
encontrarConhecimentoProblemaAvaliacao()
adicionarSubProblemaAvaliacao()
obterSubProblemaAvaliacao()
atribuirFase()
obterFase()
obterConhecimentoProblemaAvaliacao()
sub-problemas
ConhecimentoProblemaAvaliacao
(from Conhecimento)
Estimativa
tipo : int
atribuirTipo()
obterTipo()
PerfilExperiencia
experienciaDominio : int
experienciaFerramenta : int
experienciaLinguagem : int
experienciaMetodo : int
experienciaProcesso : int
experienciaProjetoSimilar : int
obterExperienciaDominio()
obterExperienciaFerramenta()
obterExperienciaLinguagem()
obterExperienciaMetodo()
obterExperienciaProcesso()
obterExperienciaProjetoSimilar()
atribuirExperienciaDominio()
atribuirExperienciaFerramenta()
atribuirExperienciaLinguagem()
atribuirExperienciaMetodo()
atribuirExperienciaProcesso()
atribuirExperienciaProjetoSimilar()
ConhecimentoMétodo
(from Conhecimento)
atribuir()
EstimativaMetodo
EstimativaExperiencia
historico : BOOL
atribuirHistorico()
obterHistorico()
EstimativaAdhoc
PlanejamentoAlocacaoRecurso
dataInicio : time
dataFim : time
qtdRecurso : long
valCusto : Real
atribuirDataInicio()
atribuirDataFim()
atribuirQtdRecurso()
atribuirValCusto()
obterDataInicio()
obterDataFim()
obterQtdRecurso()
obterValCusto()
Modificacao
nome : CString
descricao : CString
dataDescobertaModificacao : COleDateTime
tipo : string
gravidade : string
obterNome()
atribuirNome()
atribuirDescricao()
obterDescricao()
atribuirAtividade()
obterAtividade()
atribuirArtefato()
obterArtefato()
atribuirDataDescobertaModificacao()
obterDataDescobertaModificacao()
Recurso
nome : string
descricao : string
atribuirNome()
obterNome()
obterDescricao()
atribuirDescricao()
obterConhecimento()
atribuirConhecimento()
ConhecimentoRecurso
(from Conhecimento)
RecursoHumano
formação : String
atualização : String
atribuirConhecimentoRecursoHumano()
obterConhecimentoRecursoHumano()
obterNome()
atribuirFormacao()
atribuirAtualizao()
obterFormacao()
obterAtualizacao()
Recurso Hardware
Ferramenta
(from Ferramentas)
+super-atividade
+pré-atividade
+sub_atividades
artefato em desenvolvimento
Artefato Produzido
super-artefato
sub-artefatos



Modelo de Classes do Pacote ServidorConhecimentoAvaliacão.