
Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

# UTILIZAÇÃO DE MÉTRICAS EM UMA FERRAMENTA DE APOIO À MELHORIA CONTÍNUA DE PROCESSOS DE SOFTWARE

Pedro Osandy Alves Matos Júnior

TRABALHO DE GRADUAÇÃO

Recife
9 de fevereiro de 2006

Universidade Federal de Pernambuco
Centro de Informática

Pedro Osandy Alves Matos Júnior

# UTILIZAÇÃO DE MÉTRICAS EM UMA FERRAMENTA DE APOIO À MELHORIA CONTÍNUA DE PROCESSOS DE SOFTWARE

*Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.*

Orientador: *Alexandre Marcos Lins de Vasconcelos*
Co-orientador: *Sandro Ronaldo Bezerra Oliveira*

Recife
9 de fevereiro de 2006

*À minha família por todo o apoio que ela me deu para que chegasse até aqui.*

# AGRADECIMENTOS

Agradeço a meus pais, pela vida e pelo apoio que me deram. Agradeço ao prof. Alexandre Vasconcelos e a Sandro Ronaldo por toda a paciência e interesse que demonstraram enquanto me orientaram neste trabalho. Agradeço também a todos os professores e funcionários do CIn-UFPE por constituírem esse excelente ambiente de aprendizado que muito contribuiu para o meu desenvolvimento pessoal. Deixo também minha gratidão a todos os meus amigos, colegas de trabalho e de estudo, por todos os momentos de descontração e experiências obtidas. Por fim, agradeço a todos os que não foram aqui citados, mas que contribuíram para o meu sucesso.

*É mais fácil desintegrar um átomo que um preconceito.*

— ALBERT EINSTEIN

# RESUMO

Nos últimos anos, as empresas produtoras de software têm investido cada vez mais na qualidade de seus processos de desenvolvimento, esperando aumentos de produtividade e ganhos na qualidade final dos produtos criados. Surgiram várias abordagens para a melhoria dos processos de software, entre elas, o modelo, que visa a melhoria contínua do processo. Por outro lado, a utilização de métricas de software representa uma ferramenta importante para a avaliação de produtos e processos e para o controle e validação da execução de atividades. Este trabalho contempla a extensão de uma ferramenta de melhoria de processos de software, baseada no modelo IDEAL, para a utilização de métricas. Dessa forma, permite-se a utilização de métricas para validar se os objetivos de uma determinada melhoria foram alcançados e analisar o esforço de melhoria a fim de propor ações para as futuras melhorias realizadas no processo.

**Palavras-chave:** processos, métricas, melhoria contínua

# ABSTRACT

In the last years, software companies have been doing great investments on their software development processes, expecting productivity and quality improvements. Many approaches appeared to perform process improvements, among them, the IDEAL model, which focus on the continuous process improvement. On the other hand, the utilization of software metrics is a well recognized tool for performing processes and products analysis as well as for controlling and validating tasks executions. This work describes an extension to a software process improvement tool, based on the IDEAL model, to support metrics control. Thus, a user of this tool will be able to use metrics to validate the accomplishment of improvements goals. It is also possible to analise the improvement execution in order to propose actions for future improvements on the process.

**Keywords:** processes, metrics, continuous improvement

# SUMÁRIO

# LISTA DE FIGURAS

CAPÍTULO 1

# INTRODUÇÃO

Nas ultimas décadas, ocorreu um grande aumento na complexidade dos produtos de software desenvolvidos, demandando equipes de desenvolvimento cada vez maiores e degradando a qualidade dos produtos gerados e a produtividade das empresas de software. Estudos indicam que cerca de 70% dos investimentos da área possuem como objetivo a manutenção de produtos desenvolvidos anteriormente [18]. Diante deste cenário, para permanecerem de forma competitiva no mercado, as empresas desenvolvedoras de software tiveram que investir cada vez mais na qualidade dos produtos desenvolvidos e na produtividade de suas equipes de funcionários, dando origem aos processos de software.

Aumentando-se a qualidade de um processo, espera-se aumentar também a qualidade dos produtos gerados através desse processo. Várias abordagens e modelos de melhoria de processos de software foram surgindo. Entre eles, um modelo que se tornou bastante importante foi o modelo IDEAL (Initiating, Diagnosing, Establishing, Acting and Learning) [12, 24]. O IDEAL é um modelo que visa a melhoria contínua do processo de software e que aborda desde a motivação e concepção de uma mudança no processo, até a sua implantação, teste de eficácia e aprendizado obtido.

Devido à importância que os processos de software tomaram, ferramentas de apoio à definição e execução de processos de software começaram a ser desenvolvidas. Em um trabalho de graduação desenvolvido por Rafael Seabra [8], foi proposta e desenvolvida uma ferramenta que provê automação do modelo IDEAL, o *ProImprove*. Tal ferramenta foi desenvolvida de forma a se adaptada ao contexto de um projeto de doutorado que visa o desenvolvimento de um Ambiente de Implementação Progressiva de Processo de Software, o *ImPProS* [9]. Essa ferramenta permite que a organização possa manter um controle de todas as fases do modelo IDEAL em todas as melhorias executas em seus processos de software.

Entretanto, o *ProImprove* não fornecia suporte para a utilização de métricas para validar se os objetivos da melhoria haviam sido alcançados. A ferramenta também não oferecia suporte de métricas para identificar os pontos negativos ocorridos durante uma melhoria do processo, de forma a permitir que o próprio processo de melhoria seja melhorado.

A ferramenta possuía ainda alguns pontos onde poderia ser melhorada, conforme descrito em [8]. Entre os trabalhos futuros propostos estavam a necessidade de integração com outras ferramentas do ambiente, implantação de melhorias em vários ciclos e um controle sistematizado do fluxo de atividades.

O presente trabalho de graduação tem como objetivo a análise de como o *ProImprove* pode ser modificado a fim de oferecer suporte a métricas para a validar se uma dada melhoria atingiu seus objetivos e avaliar também o processo de melhoria em si. Como resultado desse trabalho, foi implementado o suporte a métricas nessa ferramenta. Como

trabalho adicional, os trabalhos futuros propostos em [8] também forma contemplados.

O restante deste trabalho está organizado da seguinte maneira: O Capítulo 2 fornece uma visão geral do modelo IDEAL no qual a ferramenta *ProImprove* se baseia. O Capítulo 3 contém uma breve discussão sobre métricas de software e sobre a abordagem GQM (*Goal Question Metric*), que serviu de base para este trabalho. No Capítulo 4 há uma breve discussão sobre processos de software e ferramentas de apoio, e são apresentados o ambiente *ImPProS* e a ferramenta *ProImprove*. O Capítulo 5 explica como ocorreu a adaptação da ferramenta *ProImprove* para utilização de métricas, que é o foco principal deste trabalho. No Capítulo 6 são apresentados os demais resultados obtidos. Por fim, o Capítulo 7 conclui o presente trabalho e apresenta uma proposta de ações futuras.

CAPÍTULO 2

# O MODELO IDEAL

Nos últimos anos, as empresas produtoras de software passaram a reconhecer cada vez mais a necessidade de uma metodologia sistemática para efetuar implantação de mudanças em seus processos de software. A fim de satisfazer tal necessidade, o Software Engineering Institute (SEI) desenvolveu o modelo IDEAL.

Inicialmente, o modelo IDEAL foi concebido visando a melhoria de processos de software, com base no modelo CMM (*Capability Maturity Model*) [16]. Com o passar do tempo, o modelo IDEAL foi sendo refinado e tornou-se robusto o suficiente para ser aplicado em qualquer esforço de melhoria, como a adoção de uma nova tecnologia, por exemplo [23]. Em [24] a pintura de um prédio é tomada como exemplo para ilustrar que esse modelo possui aplicações fora da área tecnológica.

O modelo IDEAL encontra-se dividido em cinco fases: Iniciação (*Initiating*), Diagnóstico (*Diagnosing*), Estabelecimento (*Establishing*), Ação (*Acting*) e Aprendizado (*Learning*). Essas fases formam um ciclo de vida de melhoria contínua que não só permite que a organização execute mudanças de forma organizada, como também permite que a organização esteja cada vez mais aumentando sua eficiência em realizar melhorias internas. A Figura 2.1 ilustra o ciclo de vida do modelo IDEAL.

As seções seguintes trazem uma visão geral das atividades envolvidas em cada uma dessas fases.

## 2.1 A FASE DE INICIAÇÃO

Essa fase tem como objetivo a criação de uma estrutura básica para que a melhoria possa ser executada. É nessa fase onde os objetivos da mudança são claramente definidos e documentados. É também nessa fase onde é reunido todo o apoio administrativo e infra-estrutura necessários para que a mudança ocorra. As atividades seguintes constituem essa fase.

**Estímulo para a Mudança**
É necessário que a motivação para a mudança esteja claramente definida. Uma mudança sem objetivos claros dificilmente termina em bons resultados. A motivação pode partir de estudos ou fatos visíveis dentro da organização, previsão de eventos futuros, ou decisões estratégicas. É importante que todos os membros da organização estejam cientes dos motivos que levaram à decisão da mudança. Dessa forma, os membros da organização tenderão a aceitar a mudança mais facilmente e essa terá mais changes de obter sucesso.

**Definir Contexto**
Após identificar claramente os motivos para realizar uma mudança, deve-se analisar como

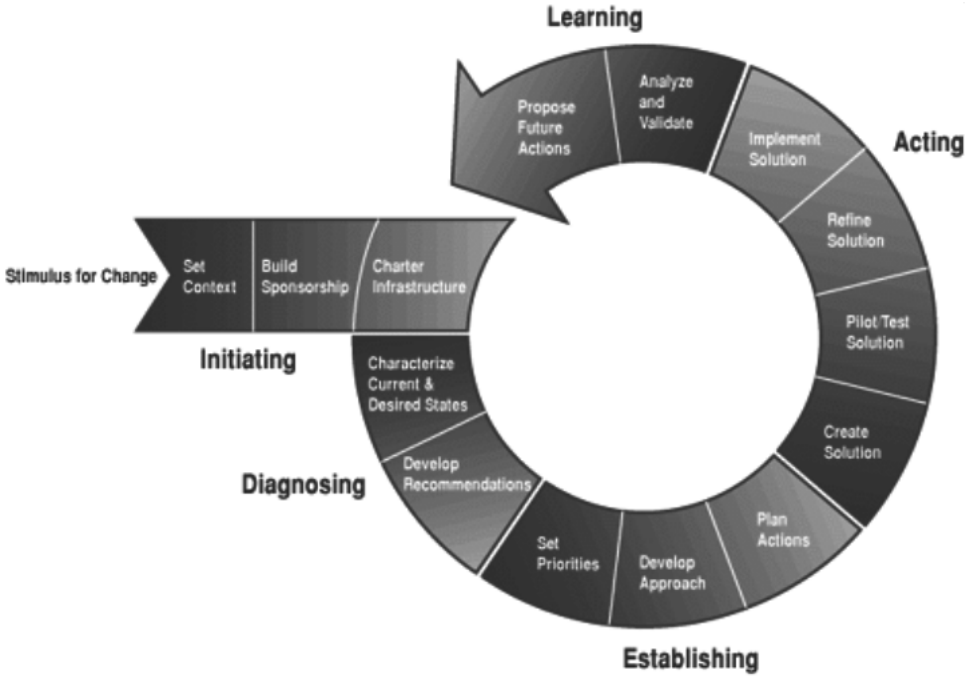



**Figura 2.1** O ciclo de vida do model IDEAL [12]

a mudança irá se encaixar dentro do contexto da organização. Isso envolve identificar quais áreas da organização serão afetadas pela mudança e qual o impacto que esta trará em cada área específica. É importante também analisar qual o impacto da mudança nas estratégias da organização e quais ações poderão ser beneficiadas ou viabilizadas através da sua implantação. Mais uma vez, é importante que cada funcionário da organização esteja ciente de como a mudança irá impactar o seu trabalho e quais o benefícios que ela trará para o contexto de seu trabalho e para a organização como um todo.

**Construir Patrocínio**
O apoio da alta administração da organização é um dos fatores que possuem maior influência para o sucesso de uma mudança. Se a mudança não for vista como algo essencial pela administração da organização, essa será facilmente descartada em momentos de escassez de recursos de tempo e financeiros. Além disso, o apoio direto do alta administração fornecerá um incentivo muito importante para que os seus demais membros vejam a mudança como uma prioridade.

**Estabelecer Infra-Estrutura**
Antes que a mudança seja executada, toda a infra-estrutura necessária deve ser estabelecida, seja ela permanente ou temporária. Caso a infra-estrutura não seja estabelecida adequadamente, a implantação da mudança pode ser interrompida ou até mesmo cancelada. Essa atividade envolve a elaboração de documentos formais e contratos que especifiquem claramente as expectativas de cada recurso alocado e delegue responsabilidades às partes envolvidas na mudança.

## 2.2 A FASE DE DIAGNÓSTICO

Essa fase possui como objetivo a obtenção de um entendimento mais aprofundado sobre a melhoria. É nessa fase onde é feita uma definição mais formal de como ficará a organização após a mudança.

**Caracterização dos Estados Atual e Desejado**
Essa fase documenta o estado atual da organização e o estado que deseja-se alcançar após a mudança. No caso específico de melhorias em processos de software, essa atividade envolveria uma descrição do estado atual do processo, como por exemplo os níveis de produtividade e qualidade alcançados atualmente e uma descrição do estado esperado do processo após a implantação da melhoria.

**Desenvolver Recomendações**
Nessa atividade, são definidas recomendações a serem seguidas durante a mudança. Essas recomendações devem ser elaboradas por pessoas com um alto grau de experiência das áreas afetadas e devem influir nas tomadas de decisões das pessoas responsáveis pela implantação da mudança.

## 2.3 A FASE DE ESTABELECIMENTO

É nessa fase onde a implantação da mudança é planejada. O plano de ações a ser elaborado deve levar em conta as recomendações efetuadas na fase de diagnóstico, além das restrições organizacionais envolvidas. As atividades seguintes fazem parte dessa fase.

**Definir Prioridades**
Essa atividade define prioridades entre as ações envolvidas com a mudança. Essas prioridades podem ser motivadas por restrições existentes na organização, forças externas ou dependência entre as ações. A definição de prioridade também deve ser fortemente influenciada pelas estratégias organizacionais.

**Desenvolver Abordagem**
Nessa atividade, o conhecimento aprofundado sobre o problema e o trabalho a ser realizado (obtido na fase de diagnóstico) é combinado com a definição de prioridades e é estabelecida uma estratégia para efetuar a mudança.

**Planejar Ações**
A partir da estratégia estabelecida pela atividade anterior, é elaborado um plano detalhado para a implantação da mudança. Esse plano deve conter todas as atividades necessárias, organizadas em um cronograma. O cronograma deve definir ainda os marcos e pontos de decisão existentes. O planejamento inclui ainda a lista de recursos necessários e a atribuição de responsabilidades. Também faz parte do planejamento a definição de métricas que permitam acompanhar a implantação da mudança e um plano para controle de riscos. Outros elementos podem ser adicionados ao planejamento, caso seja interes-

sante para a organização.

## 2.4 A FASE DE AÇÃO

É nessa fase onde as ações planejadas previamente são postas em prática. Geralmente, essa fase consome a maior parte dos recursos alocados para o esforço de mudança. Essa fase é composta pelas atividades seguintes.

**Criar Solução**
Essa atividade envolve a criação de uma solução para os problemas que serviram de estímulo para a mudança, seguindo o plano criado na fase de estabelecimento e obedecendo às restrições identificadas. Essa solução teórica deve ser criada por um grupo de técnicos experientes.

**Testar Solução**
É muito provável que ocorram problemas quando a solução proposta na atividade anterior for posta em prática. Mesmo contanto com uma ótima equipe de profissionais, o número de fatores envolvidos em um ambiente real faz com que seja essencial testar a solução antes de a por em prática. Vários tipos de testes podem ser feitos em um ambiente real, como a aplicação da solução em apenas um dos setores da organização. No caso de melhorias em processos de software, é bastante comum a utilização de um projeto piloto para testar a melhoria.

**Refinar Solução**
A experiência obtida com os testes deve ser usada pra refinar a solução, corrigindo os problemas identificados. A solução refinada deve então ser testada novamente a fim de identificar possíveis novos problemas. O ciclo teste-refinamento se repete até que a solução esteja pronta para ser aplicada.

**Implementar Solução**
Nessa atividade a solução criada e validada é implantada de fato na organização. A implantação pode se dar de várias formas. Algumas mudanças podem concentrar esforços para que sejam implantadas de forma rápida em toda a organização. Outras mudanças pode ser implantadas de forma lenta e gradual, ou até sob demanda. A abordagem utilizada para a implantação deve ser aquela que melhor se adapte ao contexto da organização.

## 2.5 A FASE DE APRENDIZADO

O modelo IDEAL é um modelo de melhoria contínua que visa a melhoria do processo de melhoria em si. Espera-se que após a execução de uma melhoria, a organização possa adquirir experiência que possibilitará que essa execute melhorias futuras de forma mais eficiente. É na fase de aprendizado onde são coletadas informações de como implementar mudanças de forma mais eficiente, completando assim o ciclo de melhoria contínua. As atividades seguintes fazem parte dessa fase.

**Analisar e Validar**
Nessa atividade, são analisados os fatos relevante observados durante todas as fases anteriores. Para cada atividade realizada, são analisados os pontos que foram executadas com sucesso, os pontos que resultaram em problemas e os pontos onde observou-se possibilidade de melhora. É verificado também se os problemas que motivaram a mudança foram atendidos em sua totalidade. Essas informações são analisadas e documentadas.

**Propor Ações Futuras**
A partir dos dados obtidos na atividade anterior, são propostas modificações que visam aumentar a capacidade da organização de efetuar mudanças de forma eficiente. Essas sugestões são documentadas a fim de serem utilizadas quando uma próxima melhoria tiver que ser executada.

## 2.6 CONSIDERAÇÕES FINAIS

Este capítulo introduziu os conceitos do modelo IDEAL, uma abordagem que pode ser utilizada para realizar o melhoria contínua de processos de software. O modelo IDEAL, além de prover o apoio necessário para executar uma melhoria específica, também foca na evolução da próprio procedimento de realização de melhorias. Dessa forma, espera-se que, ao longo do tempo, não somente a qualidade do processo de software aumente, como também que este possa ser modificado de forma mais eficiente, sempre que necessário.

O próximo capítulo introduzirá o conceito de medição de software e sua importância para a melhoria contínua de processos de desenvolvimento.

CAPÍTULO 3

# MÉTRICAS

As métricas de software desempenham um papel muito importante na gerência de projetos e no planejamento estratégico de empresas desenvolvedoras de software. Métricas também podem ser de grande importância para realizar a melhoria contínua de um processo.

Esse capítulo fornece uma visão geral sobre medição de software e sua importância. Por fim, é introduzida a abordagem GQM (*Goal Question Metric*) para utilização de métricas baseada em objetivos.

## 3.1 MEDIÇÃO DE SOFTWARE

Em geral, medição é o processo pelo qual números ou símbolos são atribuídos a atributos de entidades do mundo real de forma a descrevê-las de acordo com regras bem definidas [10]. Esse procedimento pode ser aplicado tanto para produtos quanto para processos de software.

Medição de software é o processo contínuo de definição, coleta e análise de dados no processo de desenvolvimento de software e seus produtos resultantes, com o objetivo de entendê-los, controlá-los e obter informação relevante para melhorá-los.

A medição em um processo de desenvolvimento de software se dá através da aplicação de métricas de software específicas. A medição pode envolver diferentes métricas. A literatura especializada, define algumas categorizações de métricas [17, 10, 11]. Abaixo listamos os principais tipos de métricas existentes:

- **Métricas de produto e Métricas de processo:** As métricas de produto trazem medições relativas somente ao produto final de software, ou produtos intermediários, como, por exemplo, número de linhas de código de um produto. Já as métricas de processo, fornecem dados sobre o processo de desenvolvimento como um todo, como, por exemplo número de defeitos introduzidos por hora de desenvolvimento.

- **Métricas objetivas e Métricas subjetivas:** Métricas objetivas fornecem dados absolutos sobre o objeto em estudo, tal como número de linhas de código de um produto ou número de falhas encontradas em um procedimento de testes. Já as métricas subjetivas fornecem dados que envolvem julgamento humano, como complexidade esperada de um produto ou grau de conformidade em relação a um padrão.

- **Métricas de diretas e Métricas de indiretas:** Métricas diretas são aquelas que contém dados sobre uma característica que não depende da medição de nenhuma outra característica do objeto em estudo, como, por exemplo . De maneira contrária, as métricas indiretas são aquelas cuja característica analisada depende da medição

de uma ou mais características do objeto de estudo. Por exemplo, a quantidade de defeitos por linha de código depende da medição do número de linhas de código.

## 3.2 A IMPORTÂNCIA DE MÉTRICAS EM MELHORIAS DE PROCESSOS DE SOFTWARE

As métricas fornecem um apoio valioso para entender os efeitos das ações que são implementadas para melhorar o processo de desenvolvimento de software. Segundo [15] e [17], os benefícios obtidos com a utilização de métricas são, entre outros:

- um melhor entendimento do processo de desenvolvimento;
- um maior controle do processo de desenvolvimento;
- aumento da capacidade de melhorar o processo de desenvolvimento;
- possibilidade de criar estimativas mais próximas da realidade;
- avaliações mais precisas das mudanças ocorridas no processo;
- redução de custos e tempo de desenvolvimento devido ao aumento de produtividade;
- aumento da satisfação e confiança dos clientes devido a uma maior qualidade do produto final.

Segundo [22], as informações interpretadas a partir das métricas em um processo de software podem ser aplicados para três propósitos:

1. entender o processo
2. controlar o processo
3. melhorar o processo

Em uma primeira etapa, as métricas fornecem uma visão do processo de desenvolvimento e das características dos produtos de software criados. Dessa forma, torna-se mais fácil entender o processo de software, determinando as variáveis existentes durante a sua execução. Conhecendo as variáveis envolvidas e os relacionamentos entre elas, torna-se possível controlar a execução do processo, permitindo que sejam tomadas ações corretivas e preventivas durante a execução. A próxima etapa seria utilizar o conhecimento adquirido para efetuar melhorias no processo. Através do conhecimento profundo do processo, é possível identificar pontos passíveis de melhoria em sua execução. A melhoria de uma processo se dá alterando as variáveis existentes em sua execução e os relacionamentos entre elas.

As métricas de software também podem auxiliar a execução de melhorias utilizando o modelo IDEAL, descrito no Capítulo 2. Inferências obtidas através da análise de métricas podem ser utilizadas como justificativa para a melhoria na atividade de definição do estímulo da mudança. As métricas também podem ter um papel muito importante na

fase de ação, durante a validação da implantação da melhoria. As informações inferidas a partir de métricas do processo, coletadas após a implantação da melhoria, podem indicar se o processo alcançou o estado desejado, definido na fase de diagnóstico. Outra aplicação de métricas de dá na fase de aprendizado. A aplicação eficiente de métricas pode ajudar a entender o processo de melhoria em si, identificando pontos fracos e forte na execução da melhoria. Dessa forma, é possível propor ações futuras a fim de aumentar a qualidade do processo de melhoria em si.

## 3.3 MEDIÇÃO DE SOFTWARE ORIENTADA A OBJETIVOS

De acordo com [3], a aplicação de métricas, para ser efetiva, deve ser focada em objetivos específicos e sua interpretação deve levar em consideração o contexto do ambiente organizacional e os objetivos vislumbrados. Dessa forma, a medição deve ser definida de forma *top-down*, isto é, a partir dos objetivos e estratégias organizacionais.

Uma estratégia de utilização de métricas *bottom-up*, partindo das métricas para os objetivos, dificilmente obtém sucesso, dado o enorme número de características observáveis em produtos e processos de software. Além disso, a escolha de que métricas usar em determinadas situações, e como interpretar os seus valores, não será muito bem definida. Assim, a organização arcará com os custos de um programa de coleta de métricas e irá usufruir pouco ou nada dos benefícios de um programa de métricas eficiente. Além disso, o fato de o programa de métricas não trazer resultados visíveis, fará com que os membros da organização percam o crédito por ele, comprometendo mais ainda a sua eficiência.

Essa seção fornece uma visão geral sobre a abordagem GQM (*Goal Question Metric*) [2, 4, 5, 6], que fornece uma metodologia para um controle de métricas baseado em objetivos organizacionais. Segundo [3], o modelo de medição QGM define três níveis de informação:

1. **Nível conceitual (Objetivo):** É definido o objetivo almejado pela organização.
2. **Nível operacional (Questão):** Um conjunto de questão são definidas para caracterizar de que maneira o objetivo poderá ser alcançado.
3. **Nível quantitativo (Métrica):** Um conjunto de métricas é associado a cada questão, com o objetivo de respondê-la de forma quantitativa.

Assim, o modelo GQM forma uma estrutura hierárquica onde primeiro é especificado o objetivo da medição, depois são feitas perguntas e por último, definidas as métricas, como ilustrado na Figura 3.1. Uma mesma métrica pode ser utilizada para responder mais de uma questão para um mesmo objetivo, da mesma forma que objetivos distintos podem definir questões e métricas em comum. Mesmo assim, os pontos de vista e contexto de cada objetivo devem ser levados em consideração no momento de efetuar a medição.

De acordo com [22], a implantação do modelo GQM em uma organização é realizada em quatro fases:

1. A fase de planejamento, onde é construído um planejamento para a aplicação do programa de medições, resultando em um plano de projeto.

2. A fase de definição onde o programa de medições é definido. Objetivos, questões, métricas e hipóteses são definidos e documentados.

3. A fase de coleta, onde os dados são realmente coletados.

4. A fase de interpretação, na qual os dados coletados são processados em relação às métricas definidas, provendo respostas para as questões. Por fim, o objetivo vislumbrado é avaliado.

A Figura 3.2 ilustra as quatro fases do modelo.

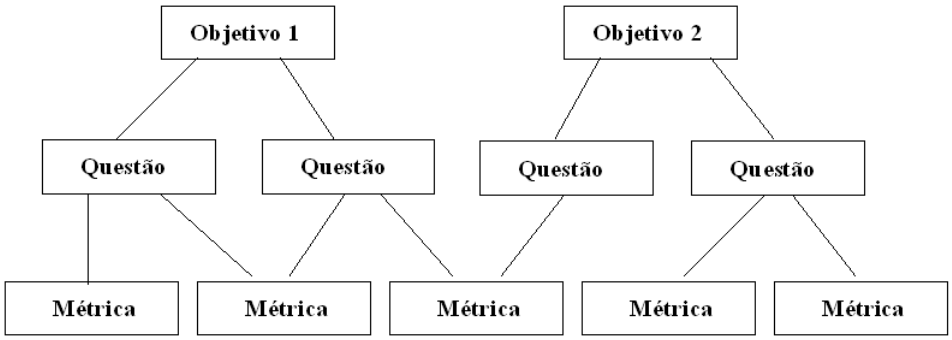



**Figura 3.1** Estrutura para definição de métricas do método GQM

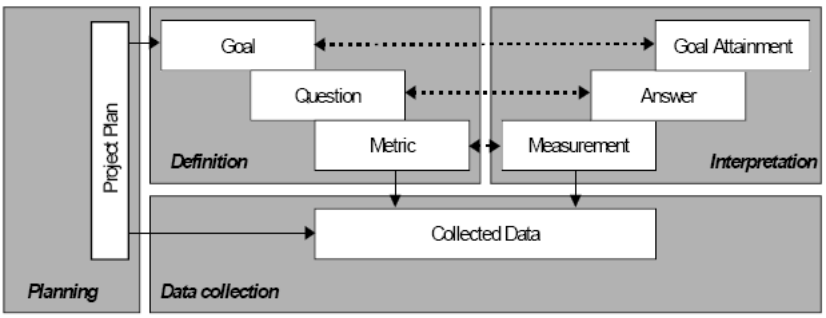



**Figura 3.2** As quatro fases do modelo GQM [22]

## 3.4 CONSIDERAÇÕES FINAIS

Este capítulo foi introduzido o conceito de medições de software, e apresentados alguns tipos de métricas existentes. Discutiu-se também sobre a importância que as métricas possuem na execução de melhorias de processos de software.

Apesar de todas as vantagens citadas, a utilização de métricas não deve ser feita de forma descontrolada, pois dessa maneira, corre-se o risco de adicionar um custo extra para a organização, sem agregar valores significativos. Com base nessa idéia, foi apresentada a abordagem GQM, que prevê a utilização de métricas, sem perder o foco nos objetivos organizacionais.

CAPÍTULO 4

# IMPPROS - UM AMBIENTE DE IMPLEMENTAÇÃO DE PROCESSO DE SOFTWARE

Esse capítulo descreve brevemente a tecnologia de processo de software e os ambientes de desenvolvimento centrados no processo. Por fim, é apresentado o ImPProS, um ambiente de implementação progressiva de processos de software que serviu com base para este trabalho.

## 4.1 TECNOLOGIA DE PROCESSO DE SOFTWARE

Segundo Carla Reis em [21], um processo de software é o conjunto de todas as atividades necessárias para criar um produto de software a partir dos requisitos de um usuário. Um processo é organizado em etapas parcialmente ordenadas, que estão relacionadas a artefatos, pessoas, recursos, estruturas organizacionais e restrições.

As etapas de execução de um processo podem ser classificadas como atividades ou tarefas. As primeiras são etapas bem gerenciadas, já as ultimas são etapas elementares, que combinadas levam à execução de uma atividade.

As atividades possuem como objetivo criar ou alterar um conjunto de artefatos. Elas podem ainda possuir relacionamentos entre si, estando associadas também a papéis e ferramentas. Atividades podem ser executas por agentes humanos, com o apoio de ferramentas, ou executadas sem intervenção humana, de maneira totalmente automatizada.

Artefatos são quaisquer produtos criados ou modificados durante a execução do processo. Artefatos podem servir como requisito para a execução de uma tarefa, ou como resultado da execução desta.

Um modelo de processo de software é uma descrição abstrata do processo de software. Vários tipos de informação devem ser integradas em um modelo de processo de software para indicar quem, quando, onde, como e por que os passos são realizados.

Um modelo do processo instanciado ou processo executável é um modelo de processo pronto para execução. Por sua vez, um projeto, segundo [18], é a instância de um processo, com objetivos e restrições específicos.

## 4.2 AMBIENTES DE DESENVOLVIMENTO DE SOFTWARE CENTRADOS NO PROCESSO

O surgimento da tecnologia CASE (Computer Aided Software Engineering) - Engenharia de Software Auxiliada por Computador, provocou um grande impacto na área de engenharia de software. Surgiram várias ferramentas capazes de auxiliar a execução de processo de produção de software. Como uma evolução dessa tecnologia, surgiu o conceito de ADS (Ambiente de Desenvolvimento de Software). Esse novo conceito baseia-se na

integração das ferramentas de apoio, fornecendo um suporte contínuo em todas as etapas do ciclo de vida de um processo de desenvolvimento.

O principal objetivo de um ADS é prover um ambiente no qual produtos de software de grande porte possam ser desenvolvidos através da integração de um conjunto de ferramentas que suportam métodos de desenvolvimento, apoiados por uma estrutura que permite a comunicação e cooperação entre as ferramentas [20]. O conceito de ADS é considerado bem mais amplo que o de ferramentas CASE por prover uma estrutura unificadora de serviços onde várias ferramentas podem ser integradas.

Uma evolução significativa nos ADS foi conseguida com a tecnologia de processos de software. A automação do processo de software foi incorporada aos ADSs mais recentes tornando-os ADS centrados em processo (ou orientados a processo), também conhecido na literatura como PSEE - Process-Centered Software Engineering Environment. Estes ambientes constituem uma nova geração de ADS que suportam além da função de desenvolvimento de software, também as funções associadas à gerência e garantia da qualidade durante o ciclo de vida do software. Um ADS centrado em processo baseia-se em uma definição explícita do processo de desenvolvimento de software. Por isso o processo de software utilizado na organização deve estar formalizado e ser obedecido.

Segundo [19], uma das implicações da utilização de ambientes orientados a processos é a necessidade de definir a execução do processo de forma mais rigorosa. Isso permite uma melhor comunicação entre os membros da organização no que diz respeito ao processo, permitindo que pessoas que desconheçam o processo tenham uma maior facilidade em seu aprendizado. Mesmo os desenvolvedores mais experientes, podem ter seu trabalho facilitado uma vez que, durante a execução do processo, o ambiente provê informações que podem guiá-lo em suas atividades. Além disso, o ambiente pode realizar algumas atividades mais simples de forma automática, permitindo que os desenvolvedores ocupem seu tempo em outras atividades, aumentando assim a produtividade da organização. Pode-se ainda, configurar o ambiente para armazenar informações sobre o desenvolvimento, tais como métricas do processo, que podem ser consultadas, quando necessário.

Outra grande vantagem da utilização de um ADS centrado no processo é a possibilidade de reunir as definições do processo em uma biblioteca reusável. Assim, pode haver a padronização de um processo organizacional, que é adaptado para cada projeto específico. Esta característica faz com que a organização não somente economize em recursos, mas também possa atingir o nível 3 de maturidade do modelo CMM, descrito em [13]. Outro recurso disponível é a coleta automática de métricas [7].

Segundo [20], os ambientes centrados no processo devem oferecer as seguintes funcionalidades:

- suporte a múltiplos usuários concorrentemente;
- prover um módulo responsável por controlar o acesso e a evolução de objetos compartilhados
- apoio à edição cooperativa de documentos e ítens de software de forma gerenciada;
- suporte à modelagem e execução de processos de software;

- permitir extensão do ambiente através da inclusão de novas ferramentas;
- integração dos módulos em todos os níveis.

## 4.3 IMPLEMENTAÇÃO DE PROCESSO DE SOFTWARE

A mudança no processo de desenvolvimento de software em uma organização não é um processo trivial e pode levar algum tempo até que os resultados sejam percebidos. Esta mudança de um processo de software é um procedimento muito mais complexa do que a utilização de uma nova ferramenta de desenvolvimento [1]. Para utilizar uma nova ferramenta, basta aprender a instalá-la e entender as instruções de como operá-la. Já a implantação de um novo processo de desenvolvimento afeta a maneira como os indivíduos trabalham, como eles vêem, e dão valor ao resultado de seu trabalho. Tal mudança afeta os indivíduos e a empresa muito mais profundamente do que a mudança de tecnologia ou ferramentas. Portanto, os resultados dessa mudança só são percebidos a longo prazo, o que reforça a necessidade de que essa mudança seja cuidadosamente planejada e gerenciada. Uma abordagem de adoção gradual do processo de desenvolvimento e ferramentas de apoio, onde cada passo seja planejado, executado e avaliado com critério, permite que a implantação do processo seja validada em etapas, permitindo a correção de erros e minimizando os riscos associados.

Um projeto de implementação de um processo de software pode ser dividido em quatro etapas, conforme ilustrado na Figura 4.1

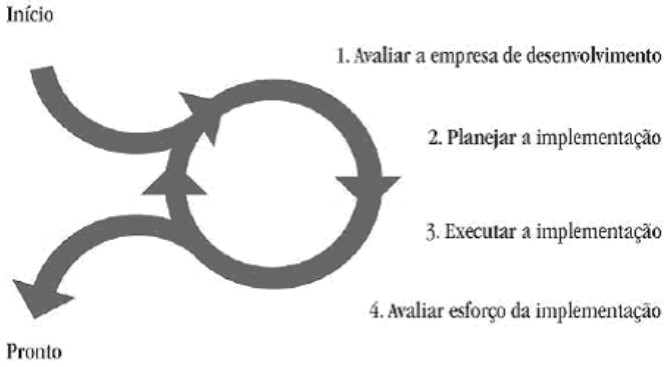



**Figura 4.1** Passos para implementação de um processo de software [9]

Na primeira etapa, o intuito é coletar informações de pessoas-chave internas ou externas à organização para obter uma lista dos problemas existentes e entender como essas pessoas vêem esses problemas e os priorizam. Este levantamento significa identificar as áreas com problemas na empresa que possuem maior prioridade para serem mudadas. Afinal, pode não ser o foco da empresa mudar o processo todo e implantar todas as ferramentas de uma só vez, mas sim fazê-lo de forma interativa e incremental, começando com as áreas que têm a maior necessidade e maior potencial para melhorias, podendo, desta forma, provar aos patrocinadores a necessidade de mudança e benefícios esperados, e estabelecer o nível de capacitação técnica que as pessoas na empresa necessitam.

Em uma segunda etapa, a implantação do novo processo deve ser cautelosamente planejada. Um planejamento efetivo deve conter: uma definição clara dos objetivos a serem alcançados e escopo de atuação; um plano de gerência de riscos eficiente; a utilização de um projeto piloto para testar a implantação; estabelecer um plano de treinamentos, visando a capacitação dos desenvolvedores; definir pessoas para servir de disseminadores de conhecimento.

Na etapa seguinte, o processo é efetivamente implantado. Isso significa executar alguns projetos de software escolhidos para adotar o processo e as ferramentas. Do ponto de vista organizacional, isso significa: monitorar os projetos de desenvolvimento de software; gerenciar a adoção de processo e ferramentas nos projetos; monitorar a criação e uso de um ambiente de desenvolvimento organizacional. A escolha de como deve-se implementar o processo e as ferramentas em uma empresa depende dos problemas que foram identificados e priorizados para o projeto e qual a capacidade de mudança que a equipe pode suportar. Deve-se ter a capacidade de atacar os maiores riscos desde o início, e de observar se estes estão sendo eliminados de forma efetiva. Uma vez concluído o projeto piloto, o uso do processo e ferramentas deve então ser avaliado com o intuito de serem adotados pelos demais projetos da empresa.

A última etapa é feita a validação de todo o esforço de implantação, isto é, os resultado são validados em relação ao plano proposto no passo na etapa *Planejar a implementação*. O ciclo pode ser reiniciado, caso uma análise dos resultados proponha uma maneira mais efetiva e eficiente de se trabalhar.

## 4.4 O IMPPROS

Como descrito nas seções anteriores, para apoiar a modelagem e a execução de processo, têm sido propostos ambientes de desenvolvimento de software centrados no processo, os quais englobam, além das ferramentas de apoio ao desenvolvedor, ferramentas que permitem modelagem do processo de software e execução do mesmo. Dessa forma, o próprio ambiente armazena conhecimento sobre o processo e pode auxilar desenvolvedores na execução de suas tarefas ou realizar algumas tarefas específicas de forma automática.

Apesar dos benefícios já citados, que esse tecnologia traz, percebe-se ao é comum que alguns processos, implementados não se adaptam às características da organização, ou não são adequados para o projeto para o qual estão sendo implementados.

Esse fato ocorre, pois muitas vezes os responsáveis pela definição do processo não conhecem profundamente as reais necessidades do ambiente onde o processo estará inserido, e acabam indicando, de forma arbitrária, as melhores práticas a serem instanciadas a partir de um processo padrão.

Assim, para ajudar uma organização na implementação progressiva de um processo de software, é útil fornecer apoio automatizado por meio de um ambiente capaz de suportar as fases que a literatura especializada propõe como necessárias. O *ImPProS* (Ambiente de Implementação Progressiva de Processos de Software) [9] é um ambiente proposto em um plano de doutorado submetido por Sandro Ronaldo Bezerra Oliveira e aprovado pelo Centro de Informática da Universidade Federal de Pernambuco. O termo *progressiva* decorre do fato de que a implementação do processo é aperfeiçoado com as experiências

aprendidas na sua definição, simulação, execução e avaliação.

O ImPProS está sendo concebido a fim de apoiar a implementação de um processo de software em uma organização. Seus principais objetivos são:

- Especificar um meta-modelo de processo de software a fim de definir uma terminologia única entre os vários modelos de qualidade de processo de software existentes, para uso do ambiente em seus serviços providos. Essa necessidade decorre da existência de vários padrões de modelos diferentes para especificar processos de software, que são encontrados na literatura.
- Apoiar a definição de um processo de software para organização.
- Permitir a modelagem e instanciação deste processo.
- Permitir a simulação do processo a partir das características instanciadas para um projeto específico.
- Dar apoio à execução do processo de software.
- Permitir avaliar se o processo atende aos critérios da organização.
- Apoiar a melhoria contínua do processo de software e o reuso através da realimentação e coleta das experiências aprendidas.

A Figura 4.2 ilustra a arquitetura do ambiente proposto, apresentando quatro tipos de usuários distintos:

- **Projetista do Processo:** responsável pela definição do processo e coleta de experiência acerca da execução de projetos. Este tipo de usuário interage com o ambiente recebendo orientações e identificando melhorias para processos existentes ou em concepção;
- **Gerente de Processo:** acompanha a simulação e avaliação do processo a fim de prover conhecimento que possibilite o reúso e a melhoria contínua dos processos;
- **Gerente de Projetos:** atua nas fases de instanciação do processo para um projeto específico, acompanhando a execução do processo e a sua avaliação para posterior coleta de experiências;
- **Equipe de Desenvolvimento:** todos os perfis relacionados à execução de um projeto de software.

O mecanismo de interação com o usuário, composto pelo Módulo de Interação e Visualização do Ambiente, possui como objetivo prover, aos usuários envolvidos com os serviços do ambiente, diferentes visões da mesma informação sendo definida e especificada. Cada perfil deve ter uma visão da informação que lhe seja mais útil.

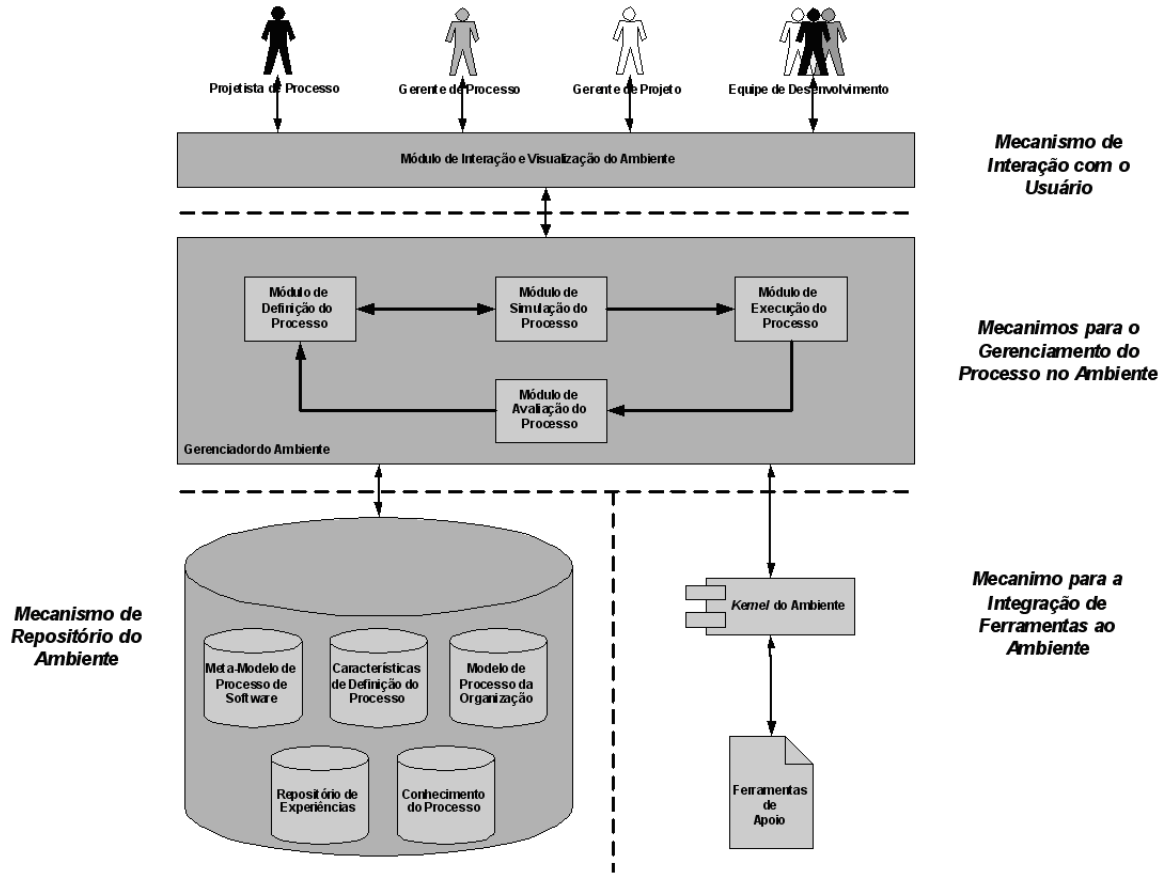



**Figura 4.2** Arquitetura do ImPProS

O mecanismo para o gerenciamento do processo no ambiente é composto pelo módulo de definição do processo, módulo de simulação do processo, módulo de execução do processo e pelo módulo de avaliação do processo. Este mecanismo possui a responsabilidade de prover serviços para ao ambiente de forma automatizada, ou seja, possibilitar que os usuários do ambiente executem suas funções tendo como referencial um guia que possibilite monitorar as suas definições.

O mecanismo de repositório provê ao ambiente um sistema de gerenciamento dos seus objetos a partir de bases de dados que permitam o controle de evolução e manutenção dos componentes do processo de software.

No mecanismo de integração de ferramentas do ambiente, o objetivo é prover a integração do ambiente com outras ferramentas de apoio tanto ao processo de software, quanto à execução do projeto de software, possibilitando desta forma a automação de atividades definidas no processo de software.

A próxima sessão descreve o *ProImprove*, uma ferramenta para melhoria contínua de processos de software que faz parte do ambiente *ImPProS*.

## 4.5 A FERRAMENTA PROIMPROVE

Em um outro trabalho de graduação [8], desenvolvido por Rafael Correia, o modelo IDEAL foi adaptado para ser utilizado em um ambiente de implementação de processo de software. Ainda como parte desse trabalho, foi desenvolvida a ferramenta *ProImprove*, um protótipo dessa adaptação. Esse capítulo fornece uma visão geral do *ProImprove* que é uma das bases do presente trabalho. A Seção 4.5.1 traz uma visão geral de como foi realizada a adaptação do modelo IDEAL, após isso, a Seção 4.5.2 fornece uma visão do fluxo das atividades envolvidas na adaptação.

### 4.5.1 Adaptação do Modelo IDEAL

A adaptação do modelo IDEAL proposta em [8] consistiu de uma análise de todas as fases do modelo visando a contextualização dessas para o ambiente de implementação de processos de software. A adaptação teve dois objetivos principais a utilização dos conceitos de melhoria contínua no contexto do ambiente e estudar como o modelo IDEAL se comportaria na implantação de melhorias em processos de software dentro desse contexto.

A adaptação proposta definiu três visões distintas:

**Gerenciamento:** responsável por planejar como as informações seriam manipuladas no ambiente pelos usuários.

**Uso (Execução):** responsável pela por manter informações relativas à execução das melhorias.

**Avaliação:** responsável por todas as atividades onde é necessário avaliar alguma informação a fim de inferir um resultado.

A adaptação definiu ainda dois perfis de pessoas responsáveis pela melhoria de um processo de software:

**Projetista de Processo:** responsável por descrever a melhoria, identificar a motivação para que esta seja executada, definir o contexto organizacional, alocar a infraestrutura necessária, definir os procedimentos necessários e fornecer uma análise e validação da solução implantada.

**Gerente de Processo:** responsável por definir um plano de ações, criar a solução, testá-la, refiná-la e por fim implantá-la

A próxima seção fornece uma rápida visão do fluxo das atividades envolvidas na adaptação proposta.

### 4.5.2 Fluxo de Atividades do ProImprove

A adaptação pode ser visualizada como um fluxo de atividades distribuídas entre o projetista do processo, o gerente do processo e o patrocinador que fornecerá o apoio necessário para que melhoria ocorra. A Figura 4.3 ilustra esse fluxo de atividades.

A atividade *Cadastrar Razão* corresponde à fase de definição do estímulo da mudança no modelo IDEAL. É nessa fase onde o projetista do processo especificará os motivos organizacionais que motivaram o início da melhoria do processo de software. Já na atividade *Definir Contexto*, o projetista descreverá como a melhoria se encaixa dentro do contexto da organização, isto é, como ela afetará os trabalhos existentes na organização e quais os benefícios que ela trará.

Na atividade *Construir Apoio* algum membro da alta administração da organização deve assumir o papel de patrocinador, formalizando o seu interesse pela melhoria e garantindo o apoio necessário para que a mesma seja executada.

Após receber o apoio necessário, o projetista do processo deverá executar a atividade *Alocar Infra-Estrutura*, preparando todos os recursos necessários para que a execução da melhoria seja iniciada e delegando responsabilidades às partes envolvidas. Uma vez montada a infra-estrutura, o projetista terá de definir os estados atual e desejado do processo, através da atividade *Caracterizar Estados*. O foco dessa atividade é definir os estados atual e desejado para cada prática do processo software que sofrerá modificação. Depois de definir claramente os estados atuais e desejados para cada prática, o projetista executará a atividade *Definir Procedimentos*, na qual especificará quais os procedimentos

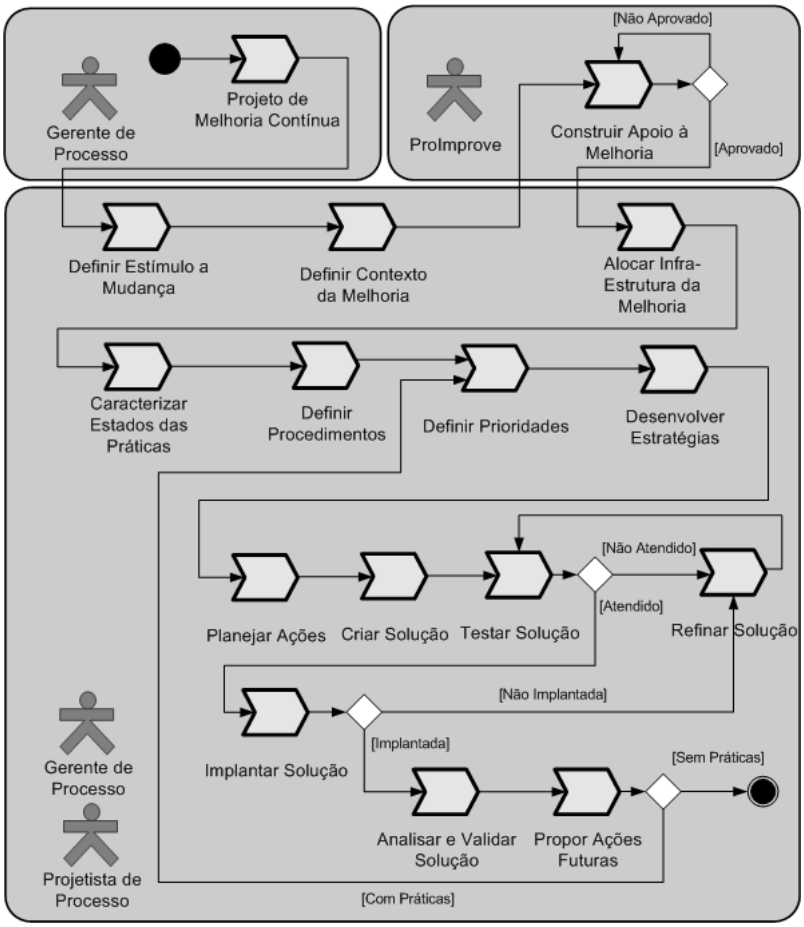



**Figura 4.3** Fluxo de Atividades do ProImprove

necessários para que os estados desejados sejam alcançados.

A próxima etapa a ser executada pelo projetista é a definição de prioridades. Nessa etapa, o projetista deve selecionar o subconjunto de práticas do processo que possuem maior prioridade em alcançar os seus estados desejados. Após definir prioridades, o projetista deverá criar um plano estratégico de execução da melhoria.

Uma vez definido o plano estratégico, o gerente do processo poderá dar início à atividade *Planejar Ações*. O foco dessa atividade é a definição de um plano de ações para implantação da melhoria, contendo um cronograma de tarefas, milestones e pontos de decisão. O plano de ações inclui ainda um gerenciamento de recursos, atribuição de responsabilidades, definição de métricas, mecanismos de *tracking*, planejamento de estratégias e gerenciamento de riscos.

Após a definição do plano de ações, o gerente poderá iniciar a atividade *Criar Solução*. A solução é composta pelas ferramentas e processos que serão utilizados; conhecimentos e habilidades necessárias; ajuda externa que será requerida e quaisquer outras informações adicionais. Após a criação da solução esta deverá ser testada e refinada. Quando a solução for finalmente validada, esta será implantada na atividade *Implantar Solução*.

Uma vez implantada a solução, o projetista do processo deverá analisar e validar a melhoria, registrando as lições aprendidas durante a sua execução. Essas lições formarão uma base de conhecimento da método de implantação de melhorias, permitindo que boas práticas sejam reconhecidas e erros sejam evitados no futuro. Se julgar necessário, o projetista poderá ainda propor ações futuras que irão trazer mais eficiência e confiabilidade à execução de melhorias futuras.

## 4.6 CONSIDERAÇÕES FINAIS

Como discutido neste capítulo, os ambientes de desenvolvimento de software centrados no processo representam um grande avanço para a engenharia de software. O ambiente *ImPProS* vêm como uma proposta da utilização dessa tecnologia na implementação progressiva de processos de software. Como parte desse ambiente, a ferramenta ProImprove é responsável por controlar a execução de melhorias realizadas no processo de software, baseando-se no conceito de melhoria contínua proposto pelo modelo IDEAL. O capítulo seguinte, descreve como o ProImprove foi adaptado permitir que a atividade de melhoria possa contar com o apoio de métricas de software.

CAPÍTULO 5

# UTILIZAÇÃO DE MÉTRICAS NO PROIMPROVE

O objetivo principal deste trabalho foi identificar como a utilização de métricas poderia aumentar a eficiência das melhorias efetuadas em um processo de software, utilizando a ferramenta ProImprove. A Seção 5.1 descreve como foi contemplada a utilização de métricas para a avaliação de uma melhoria executada. Já a Seção 5.2 descreve como foi adicionado o controle de métricas para identificar se uma melhoria fez com que o processo chegasse ao um estado desejado. Por fim, a Seção 5.3 fornecesse uma breve descrição de como essas funcionalidades foram implementadas.

## 5.1 MÉTRICAS PARA AVALIAR O ESFORÇO DE MELHORIA

Um dos focos desse trabalho foi a utilização de métricas para avaliar o esforço de melhoria. Essa avaliação tem como objetivo analisar o processo de melhoria em si, identificando os pontos de sucesso e de fracasso. Dessa forma, podem ser propostas ações futuras que permitirão que as próximas melhorias do processo sejam efetuadas de forma mais eficiente.

Para adicionar tal funcionalidade, a atividade *Analisar e Validar Solução* definida na Figura 4.3 (*Fluxo de Atividades do ProImprove*) foi estendida. Essa extensão permite que o responsável pela melhoria atribua valores a determinadas características do esforço de melhoria. A partir dos dados informados, são inferidas informações que podem auxiliar a identificação de pontos de deficiência e pontos positivos no procedimento de melhoria executado.

Na concepção dessa funcionalidade, foi utilizado o padrão de definição de métricas proposto pelo GQM, especificado na Seção 3.3. Tomou-se como objetivo: *Aumentar a qualidade e eficiência do procedimento executado para realizar melhorias no processo de desenvolvimento de software, sob o ponto de vista do responsável pelo processo.* A partir desse objetivo, foram definidas nove questões a fim de identificar os pontos de sucesso e fracasso na execução da melhoria, e assim permitir que ações futuras fossem propostas.

Para cada uma das questões definidas, foi estabelecido um conjunto de métricas associadas(denominadas de características do esforço de melhoria). A extensão da atividade *Analisar e Validar Solução* se dá pela possibilidade de o usuário atribuir valores a essas características. Dessa forma, o ambiente poderá inferir respostas para as questões definidas e auxiliar a validação do esforço de melhoria.

A lista completa de questões e características associadas e metodologia de inferência encontra-se no apêndice A.

## 5.2 MÉTRICAS PARA VALIDAR A CHEGADA AO ESTADO DESEJADO

Outra modificação na ferramenta foi realizada, visando permitir a utilização de métricas para validar a chegada ao estado desejado de cada prática considerada pela melhoria.

Conforme descrito na Seção 4.5.2, a atividade *Caracterizar Estados*, definida no fluxo de atividades do ProImprove, consiste na identificação das práticas do processo que serão afetadas pela melhoria. Para cada prática identificada é descrito o seu estado atual e especificado o seu estado desejado, isto é, como espera-se que a prática se encontre após a execução da melhoria. Essa atividade foi alterada de forma que o usuário pudesse especificar um conjunto de métricas cuja análise permitiria inferir se uma determinada prática chegou ao estado desejado. Dessa forma, para cada prática definida, o usuário possui a opção de cadastrar uma conjunto de métricas e o seu universo de valores possíveis. Para cada valor, ou conjunto de valores, pertencente ao universo de possíveis valores da métrica, o usuário da ferramenta define se este indica que a prática está validada[1] ou não.

A próxima etapa, é a definição de um percentual de validação. Esse percentual indica qual a porcentagem mínima de métricas cuja análise resulte um valor que indique que a prática foi validada. Quando todas as métricas forem analisadas, o ambiente irá verificar se o percentual mínimo foi atingido. Em caso positivo, indicará ao usuário uma sugestão de que a prática está validada.

Outra alteração, se deu na atividade *Analisar e Validar Solução*. Nessa atividade, o usuário do ambiente deve indicar para cada prática escolhida para a melhoria, se esta foi validada ou não. Essa atividade foi estendida para possibilitar que o usuário forneça dados relativos à coleta das métricas definidas na atividade *Caracterizar Estados*. Uma vez informados todos os valores para as métricas associadas a uma prática, o ambiente poderá inferir se essa prática foi validada ou não. Essa inferência é apenas uma sugestão, ficando a cargo do usuário acatá-la ou não.

## 5.3 IMPLEMENTAÇÃO DO CONTROLE DE MÉTRICAS

Na a implementação das nova funcionalidades, foram utilizadas os seguintes softwares e ferramentas:

- Java Development Kit versão 1.4
- Eclipse versão 3.1.1
- MySQL versão 5.0

Para a implementação da interface com o usuário foi utilizada a biblioteca Swing, por não depender de código nativo. A Seção 5.3.1 fornece uma visão geral sobre a implementação da validação do esforço de melhoria baseada em métricas. Por fim, a Seção 5.3.1 fornece uma visão de como foi implementado a utilização de métricas para validar se uma determinada prática chegou ao seu estado desejado.

[1] Diz-se que uma prática foi validada quando esta alcança o seu estado desejado.

### 5.3.1 Validação do esforço de melhoria

Para implementar a validação do esforço de melhoria, foram criadas cinco novas tabelas na base de dados da ferramenta, conforme ilustrado no modelo de dados do *ProImprove*, contido no apêndice B. A tabela *CaracteristicasMelhoria* armazena todas as características de todas as questões definidas no apêndice A. A tabela *ValoresMelhoria* armazena os possíveis valores de cada característica. Já a tabela *AnaliseMelhoria* permite armazenar o valor atribuído às características no que diz respeito a uma melhoria específica. A tabela *QuestoesMelhoria* armazena todas as nove questões definidas no apêndice A. Por sua vez, a tabela *InferenciaMelhoria* armazena a inferência obtida para cada questão no que diz respeito a uma melhoria específica.

Para que a alteração na atividade *Analisar e Validar Solução* fosse contemplada na ferramenta, foi adicionado o botão *Métricas* à tela *Analisando e Validando a Execução da Sub-Melhoria*, conforme ilustrado na Figura 5.1.

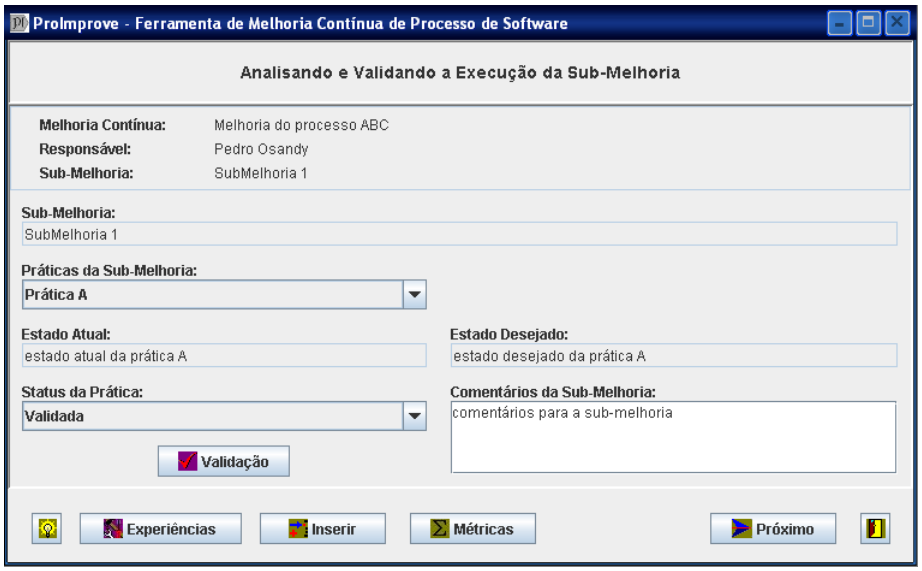



**Figura 5.1** Tela Analisando e Validando a Execução da Sub-Melhoria

Ao clicar no botão métricas, a tela *Analisando Características de Execução da Sub-Melhoria*, ilustrada na Figura 5.2, é exibida. Nessa tela, o usuário deve fornecer um valor válido para cada uma das características.

Após atribuir valores a todas as características, o usuário deverá pressionar o botão *Próximo* para continuar. Será exibida a tela *Validando Características de Execução da Sub-Melhoria*, ilustrada na Figura 5.3. Nessa tela, o usuário irá visualizar as respostas inferidas para cada uma das questões de validação da melhoria, baseadas nos valores informados para as características.

### 5.3.2 Validação dos estados das práticas

Para implementar a validação dos estados das práticas, foram criadas três novas tabelas na base de dados da ferramenta, conforme ilustrado no modelo de dados do *ProImprove*,

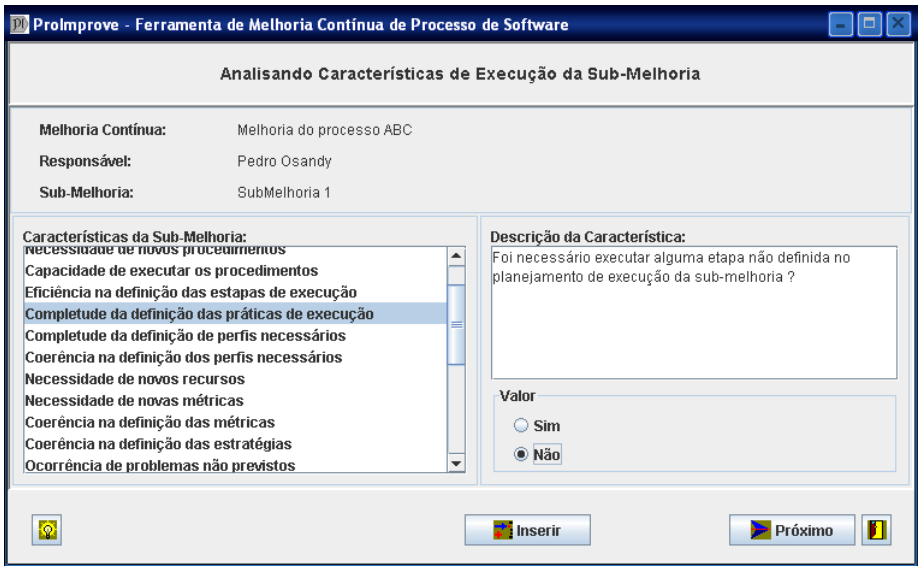



**Figura 5.2** Tela Analisando Características de Execução da Sub-Melhoria

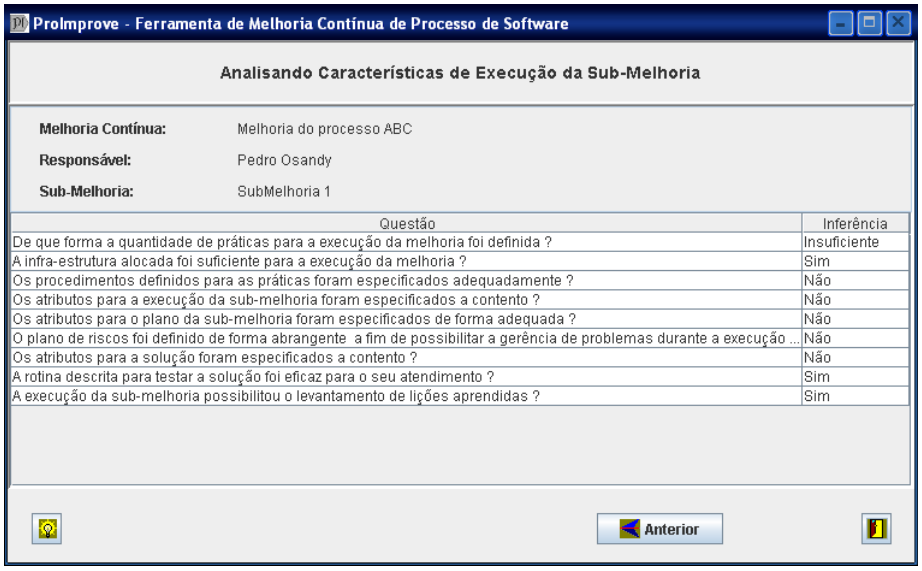



**Figura 5.3** Tela Validando Características de Execução da Sub-Melhoria

contido no apêndice B. A tabela *QuestoesPratica*, é responsável por armazenar todas as questões cadastradas para uma determinada prática. A tabela *ItemOpcao* armazena as possíveis respostas para cada uma das questões cadastradas. Já a tabela *Validacao* é responsável por armazenar o percentual mínimo de questões validadas para que a prática seja considerada validada, o percentual obtido após a coleta das métricas e o resultado inferido.

A fim contemplar a alteração da tarefa *Caracterizar Estados*, proposta na Seção 5.2, foi adicionado o botão *Validação* na tela *Caracterizando Estados das Práticas*, conforme ilustrado na Figura 5.4. Nessa figura, é tomado como exemplo a definição dos estados

atual e desejado para a prática *Ritmo Saudável*, que obriga que a equipe possua uma jornada de trabalho saudável, evitando insatisfação e perda de produtividade.

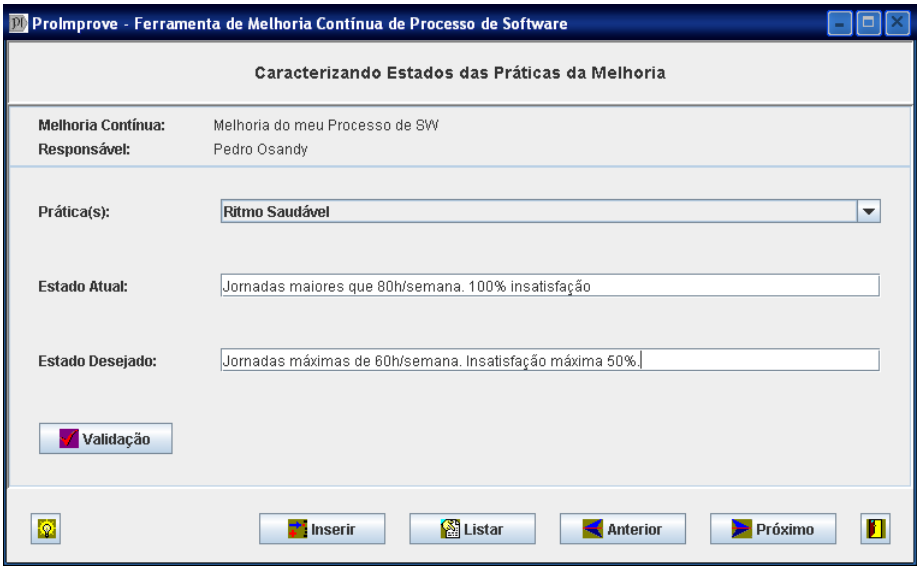



**Figura 5.4** Tela Caracterizando Estados das Práticas

Ao pressionar o botão *Validação*, a tela *Configurando Questões para validação da Prática* é exibida. Nessa tela, o usuário poderá cadastrar questões cujas respostas influenciarão na validação da prática. A Figura 5.5 e a Figura 5.6 ilustram a utilização dessa tela para cadastrar as questões que irão ajudar a identificar se o estado desejado para a prática *Ritmo Saudável* foi alcançado.

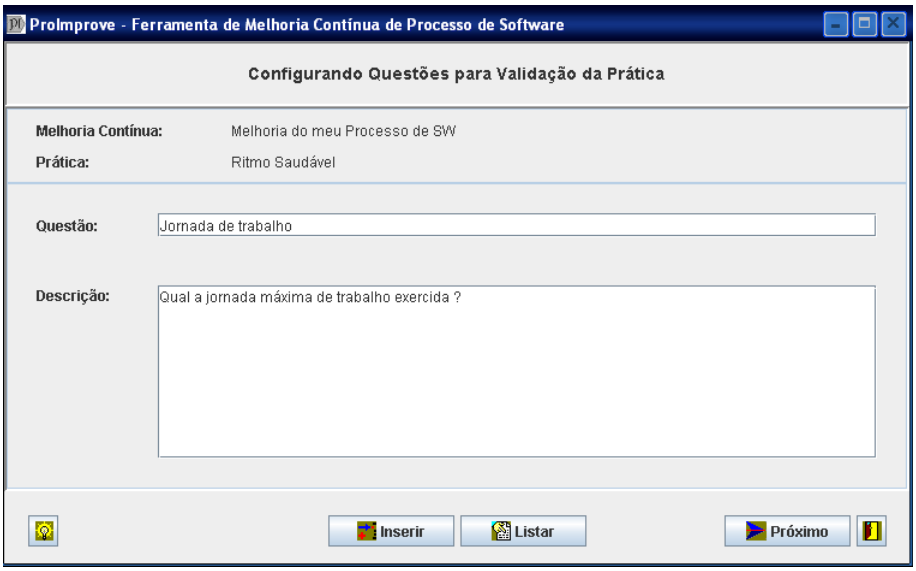



**Figura 5.5** Tela Configurando Questões para validação da Prática - Cadastrando questão *Jornada de trabalho*

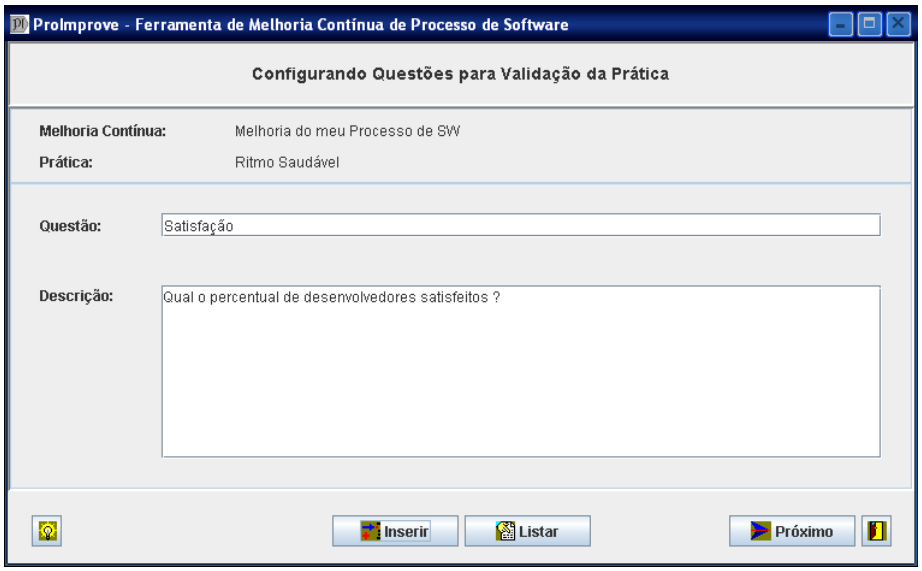



**Figura 5.6** Tela Configurando Questões para validação da Prática - Cadastrando questão *Satisfação*

Após cadastrar todas as questões, o usuário deverá pressionar o botão *Próximo*. Será exibida a tela *Configurando Análise das Questões*. Nessa tela, o usuário deverá cadastrar todos os valores possíveis para todas as questões inseridas na etapa anterior. Esses valores podem ser qualidades, no caso de métricas qualitativas, ou podem ser números ou intervalos numéricos, no caso de métricas quantitativas. Para cada valor inserido, deve ser definida também uma interpretação para a métrica: *Prática Validada* ou *Prática Não Validada*. A Figura 5.7 exemplifica a utilização dessa tela para cadastrar um possível valor de resposta para a questão *Jornada de Trabalho*.

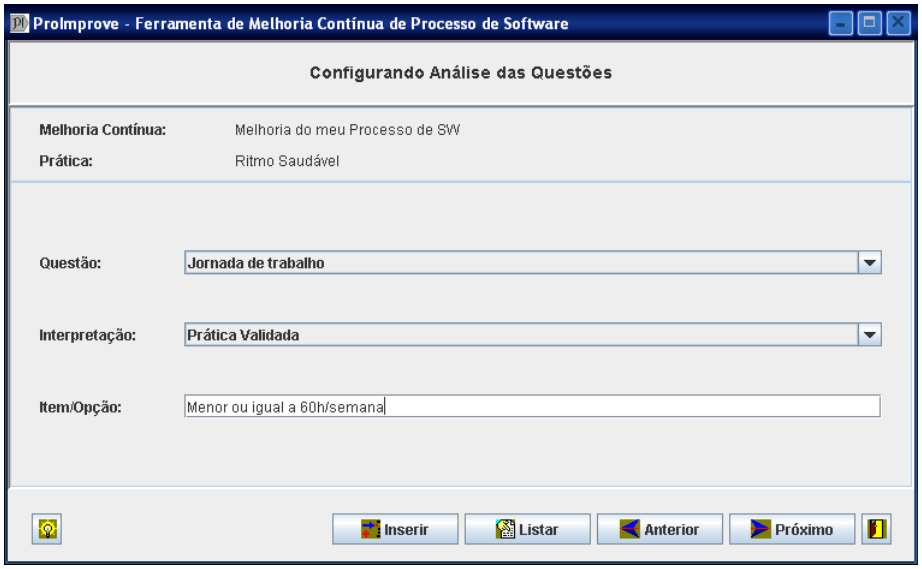



**Figura 5.7** Tela Configurando Análise das Questões

Após cadastrar todas as possíveis respostas para as questões, o usuário deverá pressionar o botão *Próximo*. Será exibida a tela *Configurando Validação da Prática*, ilustrada na Figura 5.8, na qual o usuário deverá escolher o percentual mínimo de questões cuja inferência deve ser *Prática Valida* para que o ambiente possa efetivamente inferir que a prática foi validada.

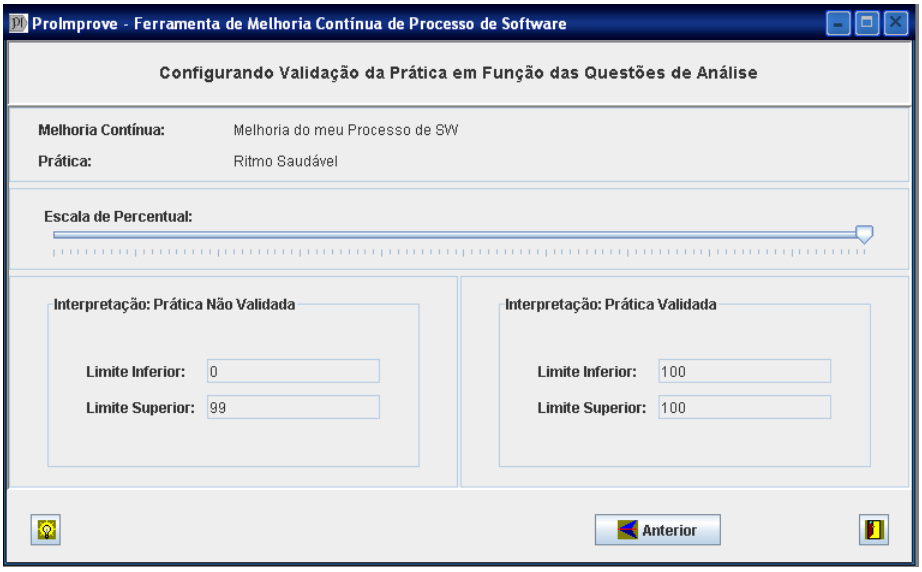



**Figura 5.8** Tela Configurando Validação da Prática

Conforme discutido na Seção 5.2, a atividade *Analisar e Validar Solução* também necessitou ser alterada para incluir o suporte de métricas na validação das práticas. Para contemplar essa alteração, foi adicionado o botão *Validação* na tela *Analisando e Validando a Execução da Sub-Melhoria*. Esse botão tem a função de realizar a validação da prática selecionada de acordo com as questões cadastradas durante a atividade *Caracterizar Estados*. A Figura 5.9 ilustra a utilização dessa tela para efetuar a validação da prática *Ritmo Saudável*.

Ao pressionar o botão *Validação*, a tela *Analisando Questões de Validação da Prática* é exibida. Nessa tela, é possível responder a cada uma das questões, utilizando um dos possíveis valores cadastrados. No exemplo ilustrado na Figura 5.10, o valor *Menor ou igual a 60h/semana* é atribuído como resposta à questão *Jornada de Trabalho*.

Após atribuir um valor válido a cada uma das questões, o usuário deverá pressionar o botão *Próximo*. Será exibida a tela *Sugerindo Resultado de Validação da Prática*, ilustrada na Figura 5.11. Essa tela exibe uma a inferência da validação da prática com relação a cada questão individualmente, e por fim, o resultado final inferido, baseado no percentual mínimo definido na tela *Configurando Validação da Prática*.

## 5.4 CONSIDERAÇÕES FINAIS

Neste capítulo, é descrito como a ferramenta ProImprove foi adaptada para oferecer suporte à utilização de métricas de software. A utilização de métricas para validar os

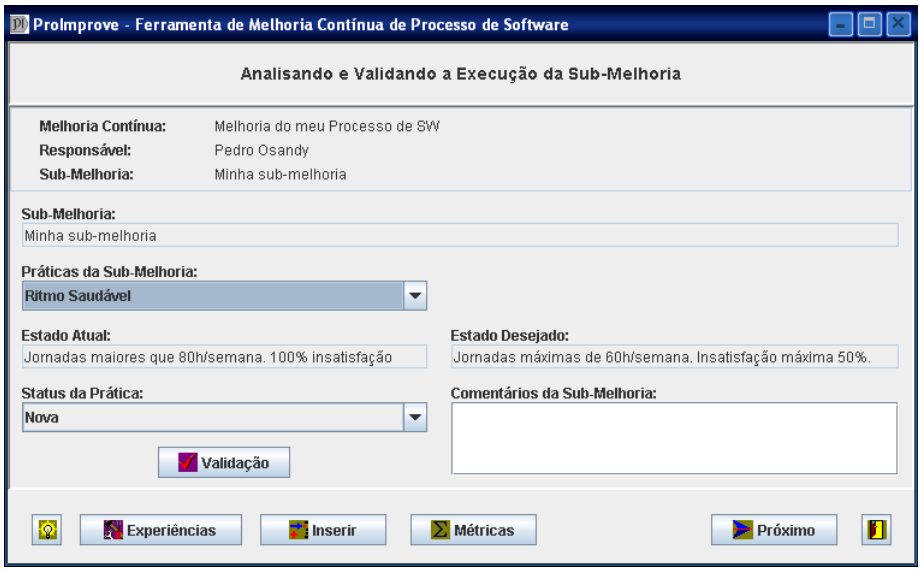



**Figura 5.9** Tela Analisando e Validando a Execução da Sub-Melhoria - Validando a prática *Ritmo Saudável*

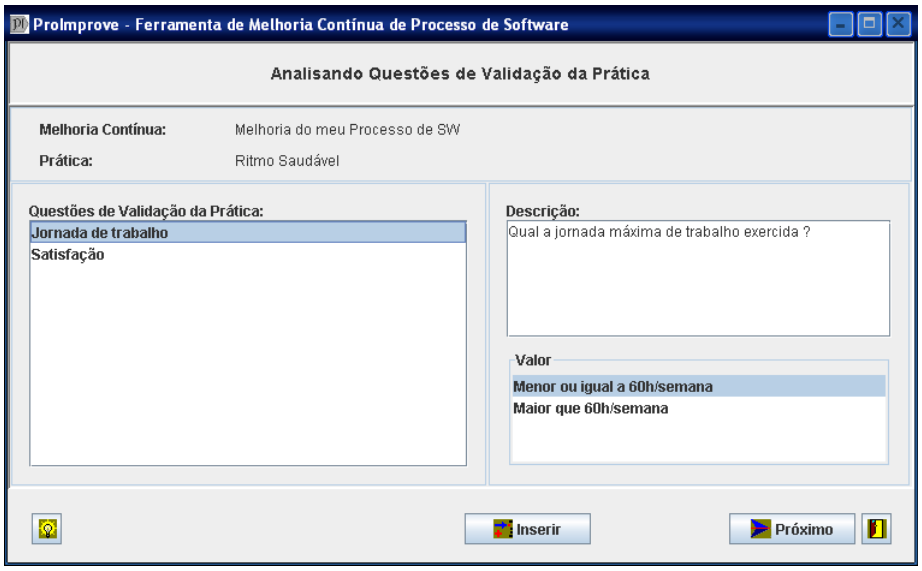



**Figura 5.10** Tela Analisando Questões de Validação da Prática

estados das práticas é um resultado importante, pois permite que os responsáveis pela melhoria possam avaliar o seu sucesso de forma mais sistematizada, apoiando-se em métricas definidas antes da melhoria ser implantada. Por outro lado, a utilização de métricas para análise do esforço de melhoria permite identificar pontos do procedimento de melhoria que podem ser melhorados, ou que devem ser mantidos. Isso permite que o responsável pela melhoria proponha ações futuras, otimizando o processo de melhoria, que é um dos objetivos vislumbrados pelo modelo IDEAL.

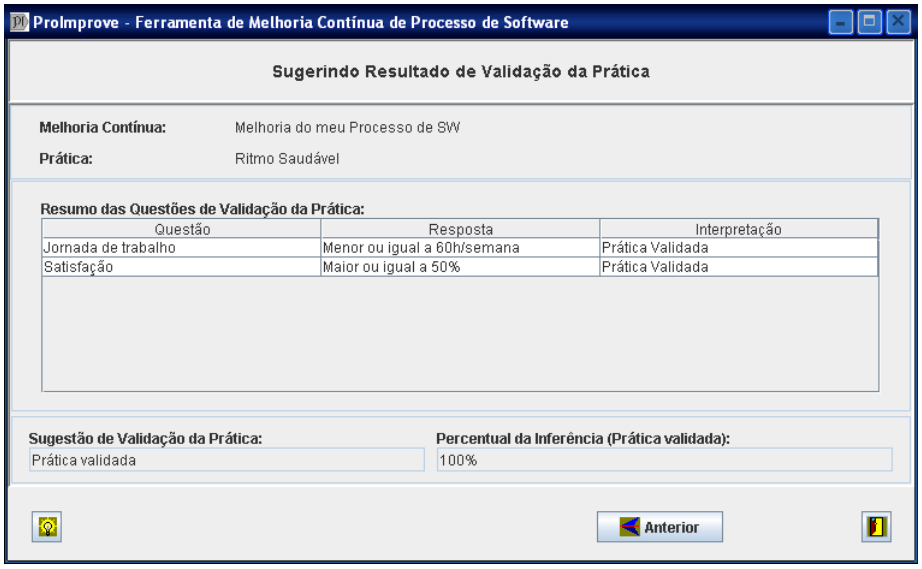



**Figura 5.11** Tela Sugerindo Resultado de Validação da Prática

CAPÍTULO 6

# OUTROS RESULTADOS

Nesse capítulo, são apresentados as demais contribuições do presente trabalho. Essas contribuições, que envolveram reestruturação e extensões da ferramenta, não faziam parte da proposta inicial, sendo implementadas como trabalho adicional. Todas essas modificações foram propostas como trabalhos futuros em [8].

## 6.1 ADEQUAÇÃO DA ESTRUTURA DE DIRETÓRIOS AO PADRÃO DO IMPPROS

A ferramenta ProImprove, apesar de desenvolvida para se integrar ao ambiente ImPProS, não estava seguindo o padrão de estrutura de diretórios definido para as ferramentas do ImPProS. Uma das contribuições deste trabalho de graduação foi a reestruturação dos diretórios da ferramenta para se adequar a esse padrão. O padrão em questão é ilustrado pela Figura 6.1.

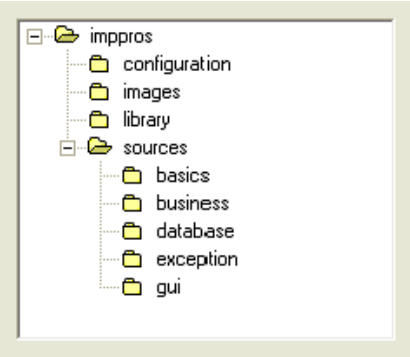



**Figura 6.1** Estruturação dos Diretórios do ImPProS e das Ferramentas de Suporte

O diretório ***imppros*** na Figura 6.1, corresponde à raiz do projeto, o qual conterá subdiretórios que organizam os arquivos fontes da automação do ImPPros e suas ferramentas de suporte. O nome deste diretório-raiz muda conforme o nome do projeto em questão, podendo assumir: ProKnowledge, ferramenta de aquisição do conhecimento; ProReuse, ferramenta de reuso de processo de software; ProImprove, ferramenta de melhoria contínua do processo de software; entre outros.

Os demais subdiretórios encontram-se definidos da seguinte maneira:

- **configuration:** diretório que mantém os arquivos que proverão a troca de mensagens entre as ferramentas. Estes arquivos possuem a configuração que permite a integração das ferramentas a partir do nível de controle.

- **images:** diretório que mantém os arquivos de imagem do projeto.
- **library:** diretório que mantém as bibliotecas de controle a serviços necessários para a automação do ImPProS e suas ferramentas de suporte como, por exemplo, controle de acesso ao banco de dados.
- **sources:** diretório que armazena os códigos-fonte provenientes da automação do ImPProS e suas ferramentas de suporte.
- **basics:** diretório que mantém as classes básicas da aplicação, ou seja, classes que representam características comuns dos objetos (entidades manipuladas pela aplicação).
- **business:** diretório que mantém as classes de negócio da aplicação.
- **database:** diretório que mantém as classes de acesso ao banco de dados.
- **exception:** diretório que mantém as classes de tratamento das exceções geradas pela aplicação.
- **gui:** diretório que mantém as classes de interface gráfica da aplicação.

## 6.2 INTEGRAÇÃO COM OUTRAS FERRAMENTAS DO AMBIENTE

Outra extensão realizada na ferramenta ProImprove foi a possibilidade de configurar a ferramenta para reconhecer as demais ferramentas do ambiente ImPProS. Essa configuração se dá a partir de um cadastro de ferramentas, como ilustrado na Figura 6.2.

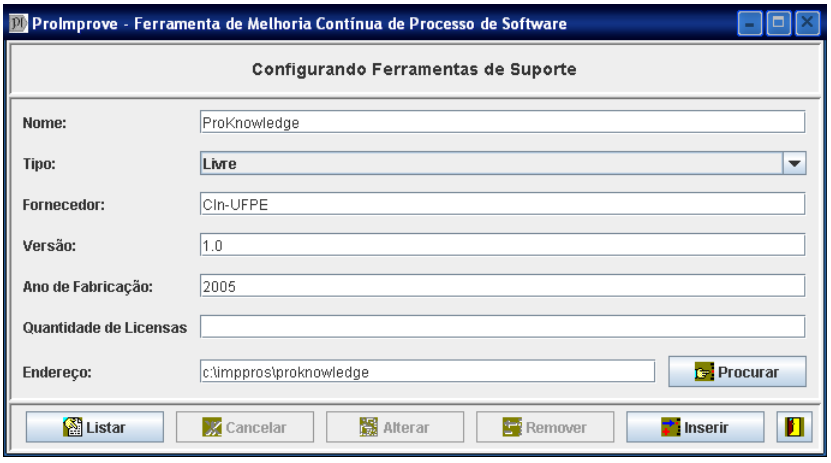



**Figura 6.2** Tela Configurando Ferramentas de Suporte

Esse cadastro permite que o ProImprove tenha conhecimento das demais ferramentas instaladas no ambiente e em quais diretórios estas estão instaladas. O padrão de estrutura de diretórios descrito na sessão 6.1, permite a interação do ProImprove com essas ferramentas, pois a ferramenta tem conhecimento de onde estão localizados os arquivos

de configuração das demais ferramentas de suporte. Além disso, o ProImprove poderá invocar uma ferramenta específica, sempre que necessário.

## 6.3 INTEGRAÇÃO COM PROKNOWLEDGE

O cadastro de ferramentas descrito na sessão anterior forneceu suporte para que fosse implementada a integração do ProImprove com a ferramenta de aquisição de conhecimento, o ProKnowledge.

Um dos problemas das organizações atuais, não restrito a empresas produtoras de software, é a falta de um mecanismo para administrar o conhecimento adquirido. A cada identificação de problema ou realização de uma atividade de forma bem sucedida, os funcionários da organização adquirem conhecimento que poderão auxiliar a realização de atividades futuras. Um problema existente é que esse conhecimento fica em posse dos funcionários. Quando os funcionários se afastam da organização, ou do setor onde trabalhava, esse conhecimento é perdido. Outro problema é que o conhecimento adquirido por um funcionários nem sempre é compartilhado com os demais. Assim, vários funcionários podem se ocupar por muitas horas para tentar solucionar um problema cuja solução já havia sido obtida por um outro membro da organização.

A ferramenta ProKnowledge provê uma solução para esse problema, permitindo que as lições aprendidas no dia-a-dia sejam armazenadas em uma base de dados. Dessa forma, o conhecimento adquirido passa a ser um patrimônio da organização, podendo ser consultado por todos os seus membros.

Essa integração foi realizada através da adição de um botão *Experiências*, contendo um ícone de uma lâmpada, no canto inferior esquerdo de todas as telas do ProImprove que estão associadas a atividades que podem possuir conhecimentos associados, cadastrados no ProKnowledge. Caso seja necessário consultar essas informações, o usuário poderá pressionar o botão *Experiências* para que a ferramenta ProKnowledge seja executada e o usuário possa efetuar a consulta. Para tanto, é necessário que a ferramenta ProKnowledge seja configurada, conforme explicado na sessão 6.2.

A implementação dessa integração se dá a partir de um arquivo XML criado pelo ProImprove e armazenado no diretório de configurações do ProKnowledge. Esse arquivo XML contém informações que será interpretadas pela ferramenta ProKnowledge, para que essa, ao ser executada, exiba uma determinada tela. No caso da integração com o ProImprove, são informados três parâmetros:

- **Função**: é informado o valor *Consulta* para que a ferramenta exiba uma tela de consulta
- **Contexto**: é informado o valor *Melhoria de Processo de Software* para que a tela de consulta exibida se adapte ao contexto de melhorias no processo
- **Palavra-Chave**: O valor informado depende da tela pela qual o usuário acessou a funcionalidade de consultar experiências. Para cada tela, existe um código associado. Dessa forma, a ferramenta ProKnowledge pode exibir o conhecimento armazenado sobre a atividade com a qual a tela está relacionada.

## 6.4 CONTROLE DE ACESSO POR PERFIL

Outra extensão realizada na ferramenta foi a adição do controle de acesso das funcionalidades baseado no perfil de usuário. A cada usuário deve ser associado um perfil. O perfil serve para categorizar o usuário e indicar a quais funções do sistema ele terá acesso. Foram definidas onze funções de acesso à ferramenta. A tabela 6.1 fornece a lista de funções disponíveis. Para cada função, existe um conjunto de perfis sugeridos que poderiam acessá-la. Essa sugestão pode não ser acatada, se necessário. Novos perfis podem ser criados e funções atribuídas a eles de forma arbitrária.

| Funções do ProImprove | Sugestão de Perfis |
|---|---|
| Controle de Acesso | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Alteração de Senha | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Nova Melhoria Contínua | Gerente de Processo/Administrador do ProImprove |
| Abrir Melhoria Contínua | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Manutenção - Usuários do ProImprove | Administrador do ProImprove |
| Manutenção - Configurando Ferramentas de Suporte | Administrador do ProImprove |
| Melhoria Contínua - Execução | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Melhoria Contínua - Visualização | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Suporte | Gerente de Processo/Projetista do Processo/Administrador do ProImprove |
| Ajuda | Todos |
| Sobre o ProImprove | Todos |

**Tabela 6.1** Funções do ProImprove

## 6.5 IMPLANTAÇÃO DA MELHORIA EM VÁRIOS CICLOS

Outro resultado importante foi a possibilidade de implantar a melhoria em vários ciclos independentes como previsto no fluxo de atividades do ProImprove, ilustrado na Figura 4.3. Na atividade *Definir Prioridades* é criada uma nova sub-melhoria, que irá contemplar a melhoria no que diz respeito somente a um subconjunto das práticas do processo. As práticas selecionadas para compor a sub-melhoria são aquelas de maior prioridade. O fluxo de atividade é então executado normalmente para a sub-melhoria criada até finalizar a atividade *Propor Ações Futuras*. Caso ainda haja práticas da melhoria

que ainda não chegaram ao seu estado desejado, a atividade *Definir Prioridades* deve ser executada novamente a fim de contemplar essas práticas, criando uma nova sub-melhoria que contemple um subconjunto dessas práticas. Dessa forma, podem ser criadas quantas sub-melhorias forem necessárias, até que todas as práticas da melhoria tenham sido validadas, finalizando assim o ciclo. A Figura 6.3 ilustra a tela *Definindo Prioridade de Execução das Práticas da Melhoria*, onde uma nova sub-melhoria é criada, escolhendo-se um subconjuntos das práticas não validadas da melhoria.

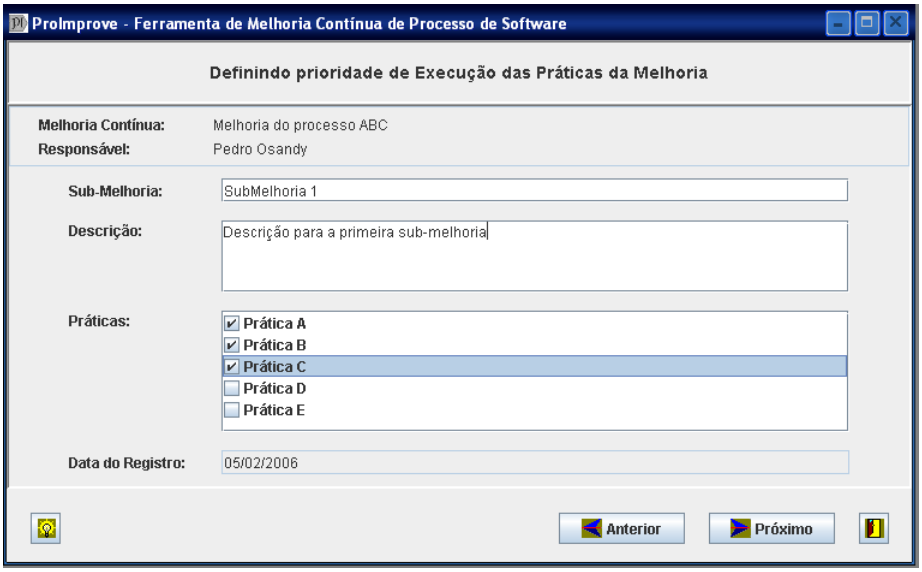



**Figura 6.3** Tela Definindo Prioridade de Execução das Práticas da Melhoria

## 6.6 CONTROLE SISTEMATIZADO DO FLUXO ATIVIDADES

Foi implementado também um controle sistematizado do fluxo de atividades desempenhadas durante a utilização da ferramenta. Antes dessa modificação, as telas relacionadas às atividades de execução da melhoria eram acessadas a partir do menu principal da aplicação. Dessa forma, a aplicação não orientava o usuário de forma sistematizada através das fases de execução da melhoria, aumentando a complexidade de utilização da aplicação por um usuário inexperiente.

A modificação consistiu em centralizar a execução da melhoria em uma única tela. Essa tela possui um painel central onde o usuário pode consultar e inserir informações referentes à atividade corrente. Além disso, a tela possui um botão *Próximo*. Esse botão é pressionado sempre que o usuário julgar a atividade como concluída, partindo para a atividade seguinte.

Como o estado da execução da melhoria é persistido na base de dados da ferramenta, o usuário poderá fechar a aplicação, e ao abrir a tela de execução da melhoria, será exibido o painel referente à última atividade não concluída.

## 6.7 CONSIDERAÇÕES FINAIS

Além dos trabalho inicialmente propostos, este trabalho apresenta resultados adicionais. Esses resultados são importantes pois facilitam a utilização da ferramenta, além de fornecer a integração com o ambiente ImPProS, onde a mesma está inserida.

CAPÍTULO 7

# CONCLUSÃO

A utilização de métricas é de grande importância para os empresas desenvolvedoras de software. Em particular, a realização da melhoria contínua de um processo de software pode ser fortemente auxiliada pela utilização de métricas.

Neste trabalho, foi identificado como uma ferramenta para melhoria contínua de processos de software poderia ser estendida para utilizar métricas na validação dos resultados de uma solução de melhoria e para análise do próprio processo de melhoria a fim de expor pontos onde este possa ser melhorado. A ferramenta estudada foi estendida para contemplar os resultados obtidos. A partir desses resultados, é possível utilizar a ferramenta para inferir se os objetivos de um determinada melhoria foram alcançados, e assim dizer que a mesma foi finalizada ou se deve ser executada novamente. Tornou-se possível também, obter apoio da ferramenta para propor otimizações nos procedimentos para execução de melhorias utilizados por uma organização, uma atividade importante proposta pelo modelo IDEAL.

Como trabalho adicional, foram solucionadas outras deficiências da ferramenta, identificadas anteriormente. Essas deficiências dizem respeito à usabilidade da ferramenta, falta de um controle sistematizado do fluxo de atividades executadas e pouca integração com o ambiente ImPProS. Propomos ainda, como trabalho futuro, a aplicação da ferramenta em um estudo de caso que envolva a utilização de métricas na análise e validação da melhoria.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] Ricardo Balduino. Implementação de um processo de desenvolvimento de software: uma abordagem passo-a-passo. Rational Software White Paper, 2002.

[2] Victor Basili. *Software Modeling and Measurement: The Goal Question Metric Paradigm*. 1992.

[3] Victor Basili, Gianluigi Caldiera, and Dieter Rombach. *The Goal Question Metric Approach. Encyclopedia of Software Engineering*. Wiley and Sons, 1994.

[4] Victor Basili and Dieter Rombach. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, 1988.

[5] Victor Basili and Richard Selby. Data collection and analysis in software research and management. *Proceedings of the American Statistical Association and Biomeasure Society*, 1984.

[6] Victor Basili and David Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, 1984.

[7] Alan Christie. *Software Process Automation: The Technology and its adoption*. Springer Verlag, 1997.

[8] Rafael Correia. Avaliação (adaptação) do modelo IDEAL para um ambiente de implementação de processo de software. Trabalho de conclusão da graduação em Ciência da Computação, Universidade Federal de Pernambuco, 2005.

[9] Sandro Oliveira e Alexandre Vasconcelos. Proposta de um ambiente de implementação de processo de software, 2005. Proposta de doutorado submetida ao Centro de Informática da Universidade Federal de Pernambuco.

[10] Norman Fenton and Shari Pfleeger. *Software metrics (2nd ed.): a rigorous and practical approach*. PWS Publishing Co, 1997.

[11] Robert Grady. *Practical software metrics for project management and process improvement*. Prentice Hall, 1992.

[12] Jennifer Gremba and Chuck Myers. The IDEAL model: A practical guide for improvement. Software Engineering Institute (SEI), 1997. Disponível em http://www.sei.cmu.edu/ideal/ideal.bridge.html. Último acesso: 28 de janeiro de 2006.

[13] Watts Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.

[14] Robert McFeeley. *IDEAL$^{SM}$: A User's Guide for Software Process Improvement, Software Engineering Institute Handbook*. Carnegie Mellon University, 1996.

[15] Möller and Paulish. *Software Metrics: a Practitioner's Guide to Improved Product Development*. Chapman and Hall, 1993.

[16] Mark Paulk, Bill Curtis, Mary Chrissis, and Charles Weber. *Capability maturity model for software, version 1.1*. Software Engineering Institute, 1993.

[17] Shari Pfleeger. *Software Engineering: the Production of Quality Software*. Macmillan, 1991.

[18] Roger Pressman. *Engenharia de Software*. Makron Books, 2002.

[19] Carla Reis. Ambientes de desenvolvimento de software e seus mecanismos de execução de processos de software. PPGC-UFRGS. Exame de Qualificação do Doutorado.

[20] Carla Reis. Reutilização de processos de software. PPGC-UFRGS. Exame de Qualificação do Doutorado.

[21] Carla Reis. Um gerenciador de processos de software para o ambiente PROSOFT. Master's thesis, PPGC-UFRGS, 1998.

[22] Van Solingen and Egon Berghout. *The goal/question/metric method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[23] Jordan Vause. The IDEAL$^{SM}$ transition framework: Speeding managed change book 'em case study. Software Engineering Institute (SEI), 1997. Disponível em http://www.sei.cmu.edu/activities/ideal/ideal.case.study.html. Último acesso: 28 de janeiro de 2006.

[24] Jordan Vause. Painting a building: A conceptual reconstruction based upon the IDEAL$^{SM}$ model. Software Engineering Institute (SEI), 1997. Disponível em http://www.sei.cmu.edu/ideal/ideal.paint.html. Último acesso: 28 de janeiro de 2006.

APÊNDICE A

# QUESTÕES PARA AVALIAÇÃO DA MELHORIA

Esse apêndice descreve as questões definidas para validar a execução da melhoria. A partir dessas questões é possível inferir se a melhoria foi bem sucedida ou não, além de identificar pontos fracos e fortes, que podem ser aprimorados em melhorias futuras. Cada questão possui um conjunto de características (métricas) associadas. Para cada questão, é especificado o procedimento para inferir sua resposta a partir dos valores atribuídos às suas características relacionadas.

## A.1 QUESTÃO 1

### Como avalia-se a quantidade de práticas definidas para a melhoria?

| Característica | Pergunta |
|---|---|
| Eficiência da implantação das práticas | Os resultados obtidos após a implantação das práticas foram satisfatórios para suprir as necessidades identificadas na definição dos objetivos da melhoria? |

**Inferência:**

Se o valor *SIM* for atribuído à característica *Eficiência da implantação das práticas*, então a inferência da questão será **SUFICIENTE**; caso contrário, a inferência da questão será **INSUFICIENTE**. Essa inferência é válida, pois a melhoria só pode ser considerada como concluída e avaliada quando

## A.2 QUESTÃO 2

### A infra-estrutura alocada foi suficiente para a execução da melhoria?

| Característica | Pergunta |
|---|---|
| Condições de Infra-Estrutura | A implantação da melhoria foi interrompida for falta de infra-estrutura? |

**Inferência:**

Se o valor *SIM* for atribuído à característica *Condições de Infra-Estrutura*, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.3 QUESTÃO 3

### Os procedimentos definidos para as práticas foram especificados adequadamente?

| Característica | Pergunta |
|---|---|
| Necessidade de novos procedimentos | Foi necessária a adoção de procedimentos para as práticas da sub-melhoria que não haviam sido previstos anteriormente? |
| Capacidade de executar os procedimentos | Algum dos procedimentos definidos para as práticas não pôde ser seguido? |

**Inferência:**

Se o valor *SIM* for atribuído a qualquer uma das características listadas, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.4 QUESTÃO 4

### Os atributos para a execução da sub-melhoria foram especificados a contento?

| Característica | Pergunta |
|---|---|
| Eficiência na definição das etapas de execução | Algum passo definido para a execução da sub-melhoria não pôde ser seguindo? |
| Completude da definição das práticas de execução | Foi necessário executar alguma etapa não definida no planejamento de execução da sub-melhoria? |
| Completude da definição de perfis necessários | Foi necessário algum perfil profissional não previsto para a execução da sub-melhoria? |
| Coerência na definição dos perfis necessários | Algum perfil previsto para a execução da sub-melhoria não foi necessário? |

**Inferência:**

Se o valor *SIM* for atribuído a qualquer uma das características listadas, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.5 QUESTÃO 5

### Os atributos para o plano da sub-melhoria foram especificados de forma adequada?

| Característica | Pergunta |
|---|---|

| | |
|---|---|
| Necessidade de novos recursos | Foram necessários recursos não previstos no planejamento da sub-melhoria? |
| Necessidade de novas métricas | Foram requeridas métricas não previstas para a sub-melhoria? |
| Coerência na definição das métricas | Alguma métrica prevista para a sub-melhoria não foi coletada? |
| Coerência na definição das estratégias | As estratégias definidas no plano da sub-melhoria foram realmente seguidas? |

**Inferência:**
Se o valor *NÃO* for atribuído à característica *Coerência na definição das estratégias* ou o valor *SIM* for atribuído a qualquer uma das demais características, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.6 QUESTÃO 6

### O plano de riscos foi definido de forma abrangente a fim de possibilitar a gerência de problemas durante a execução da solução?

| Característica | Pergunta |
|---|---|
| Ocorrência de problemas não previstos | Ocorreu alguma situação problemática não prevista pelo plano de riscos? |
| Completude do plano de riscos | Algum risco potencial foi descoberto somente após a finalização da fase de planejamento do controle de riscos? |

**Inferência:**
Se o valor *SIM* for atribuído a qualquer uma das características listadas, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.7 QUESTÃO 7

### Os atributos para a solução foram especificados a contento?

| Característica | Pergunta |
|---|---|
| Necessidade de novas ferramentas | Foi utilizada alguma ferramenta não especificada no planejamento da solução? |
| Coerência na definição das ferramentas | Alguma ferramenta especificada no planejamento da solução não foi utilizada? |
| Coerência na definição dos processos para a solução | Algum processo definido para a solução não foi utilizado? |

| | |
|---|---|
| Completude da definição dos processos para a solução | Foi utilizado algum processo para a solução que não havia sido definido em seu plano? |
| Necessidade de novas habilidades | Foi necessária alguma habilidade não prevista no planejamento da solução? |
| Coerência na definição das habilidades necessárias | Alguma habilidade prevista no planejamento da solução não foi necessária à sua implantação? |
| Necessidade de ajuda externa adicional | Foi requisitada alguma ajuda externa não prevista no planejamento da solução? |
| Necessidade de conhecimentos adicionais | Foi necessário algum conhecimento não especificado no planejamento da solução? |

**Inferência:**

Se o valor *SIM* for atribuído a qualquer uma das características listadas, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.8 QUESTÃO 8

### A rotina descrita para testar a solução foi eficaz para o seu atendimento?

| Característica | Pergunta |
|---|---|
| Eficiência do plano de testes | Foram encontrados problemas na solução proposta para a sub-melhoria que não forma contemplados em seu plano de testes? |

**Inferência:**

Se o valor *SIM* for atribuído à característica *Eficiência do plano de testes*, então a inferência da questão será **NÃO**; caso contrário, a inferência da questão será **SIM**.

## A.9 QUESTÃO 9

### A execução da sub-melhoria possibilitou o levantamento de lições aprendidas?

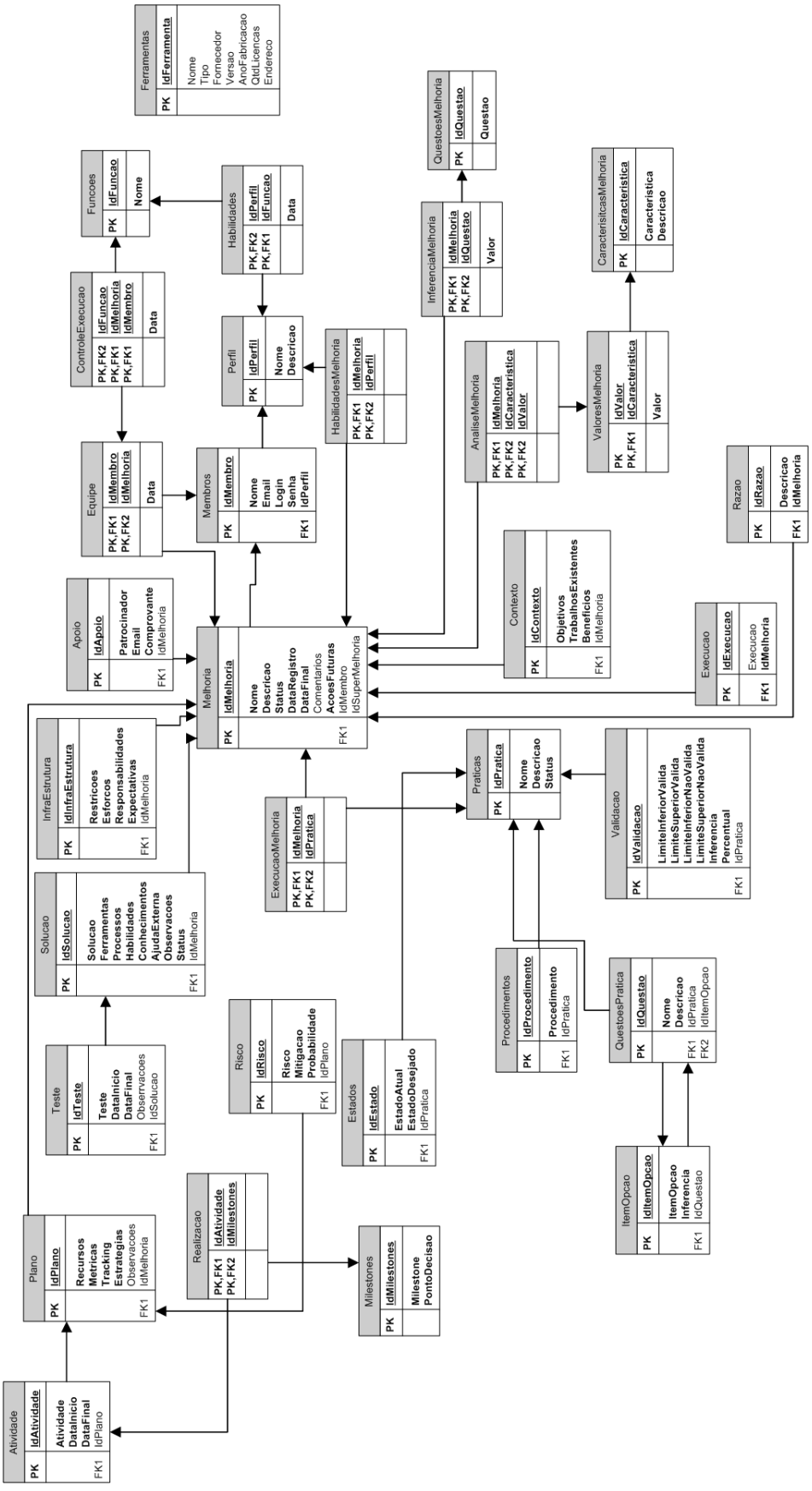
| Característica | Pergunta |
|---|---|
| Aprendizado | Alguma lição foi aprendida durante a execução da sub-melhoria? |

**Inferência:**

Se o valor *SIM* for atribuído à característica *Aprendizado*, então a inferência da questão será **SIM**; caso contrário, a inferência da questão será **NÃO**.

APÊNDICE B

# MODELO DE DADOS DO PROIMPROVE


Ferramentas
IdFerramenta
Nome
Tipo
Fornecedor
Versao
AnoFabricacao
QtdLicencas
Endereco
Funcoes
IdFuncao
Nome
Habilidades
IdPerfil
IdFuncao
Data
ControleExecucao
IdFuncao
IdMelhoria
IdMembro
Data
Perfil
IdPerfil
Nome
Descricao
HabilidadesMelhoria
IdMelhoria
IdPerfil
QuestoesMelhoria
IdQuestao
Questao
InferenciaMelhoria
IdMelhoria
IdQuestao
Valor
CaracteristicasMelhoria
IdCaracteristica
Caracteristica
Descricao
Equipe
IdMembro
IdMelhoria
Data
Membros
IdMembro
Nome
Email
Login
Senha
IdPerfil
AnaliseMelhoria
IdMelhoria
IdCaracteristica
IdValor
ValoresMelhoria
IdValor
IdCaracteristica
Valor
Apoio
IdApoio
Patrocinador
Email
Comprovante
IdMelhoria
Melhoria
IdMelhoria
Nome
Descricao
Status
DataRegistro
DataFinal
Comentarios
AcoesFuturas
IdMembro
IdSuperMelhoria
Contexto
IdContexto
Objetivos
TrabalhosExistentes
Beneficios
IdMelhoria
Razao
IdRazao
Descricao
IdMelhoria
Execucao
IdExecucao
Execucao
IdMelhoria
InfraEstrutura
IdInfraEstrutura
Restricoes
Esforcos
Responsabilidades
Expectativas
IdMelhoria
Praticas
IdPratica
Nome
Descricao
Status
ExecucaoMelhoria
IdMelhoria
IdPratica
Validacao
IdValidacao
LimiteInferiorValida
LimiteSuperiorValida
LimiteInferiorNaoValida
LimiteSuperiorNaoValida
Inferencia
Percentual
IdPratica
Solucao
IdSolucao
Solucao
Ferramentas
Processos
Habilidades
Conhecimentos
AjudaExterna
Observacoes
Status
IdMelhoria
Procedimentos
IdProcedimento
Procedimento
IdPratica
QuestoesPratica
IdQuestao
Nome
Descricao
IdPratica
IdItemOpcao
Teste
IdTeste
Teste
DataInicio
DataFinal
Observacoes
IdSolucao
Risco
IdRisco
Risco
Mitigacao
Probabilidade
IdPlano
Estados
IdEstado
EstadoAtual
EstadoDesejado
IdPratica
ItemOpcao
IdItemOpcao
ItemOpcao
Inferencia
IdQuestao
Plano
IdPlano
Recursos
Metricas
Tracking
Estrategias
Observacoes
IdMelhoria
Realizacao
IdAtividade
IdMilestones
Milestones
IdMilestones
Milestone
PontoDecisao
Atividade
IdAtividade
Atividade
DataInicio
DataFinal
IdPlano