UMA ABORDAGEM PARA ACOMPANHAMENTO DE PROJETOS DE SOFTWARE

Fabio Perez Marzullo

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

Aprovada por:

Prof. Geraldo Bonorino Xexéo, D. Sc.

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

Profa. Cláudia Maria Lima Werner D.sc.

Prof. Arnaldo Dias Belchior D.sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2006

MARZULLO, FABIO PEREZ

Uma Abordagem para Acompanhamento de Projetos de Software [Rio de Janeiro] 2006

IX, 135 p. 29,7 cm (COPPE/UFRJ M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Métricas de Software

I. COPPE/UFRJ II. Título (série)

Este trabalho é dedicado aos meus pais:

– Minha Mãe Maria Lúcia Perez Marzullo e Meu Pai Antônio João Marzullo Filho

Agradecimentos

Gostaria de agradecer, acima de tudo, aos meus pais Maria Lúcia Perez Marzullo e Antonio João Marzullo Filho, por sempre estarem ao meu lado, amparando, apoiando, ajudando e amando. Sem tamanho apoio certamente não teria conseguido finalizar este trabalho. Tudo que sou hoje devo a vocês, incondicionalmente! Agradeço aos meus irmãos Thatiana Perez Marzullo e Marcio Perez Marzullo, por terem paciência e sabedoria nos momentos mais difíceis e turbulentos, e da mesma forma, por me apoiarem e amarem.

Agradeço a minha namorada Renata Braga Dias por aceitar e entender as inúmeras horas, dias, meses e anos de estudos. Sei que não foram fáceis e por isso acredito que seu amor foi fundamental para que eu pudesse enfrentar e vencer todos os obstáculos.

Também gostaria de agradecer ao meu amigo Marco Antônio Aniceto Vaz, por sua paciência e sabedoria. Muito do que representa este trabalho se deve a sua compreensão. Aos amigos e companheiros de trabalho, por me ensinarem espírito de equipe e companheirismo.

Agradecimentos especiais a todos aqueles que fizeram parte deste trabalho direta ou indiretamente.

Ao professor Jano Moreira de Souza por confiar em meu trabalho nos projetos desenvolvidos pelo laboratório de Banco de Dados. Aos professores Cláudia Werner e Arnaldo Dias Belchior, por aceitarem fazer parte da banca de avaliação.

Aos meus orientadores Ana Regina Rocha e Geraldo Bonorino Xexéo, por terem acreditado em mim e confiado na minha capacidade.

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

UMA ABORDAGEM PARA ACOMPANHAMENTO DE PROJETOS DE SOFTWARE

Fabio Perez Marzullo

Março/2006

Orientadores: Geraldo Bonorino Xexéo,
Ana Regina Cavalcanti Rocha

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma abordagem para o acompanhamento de projetos de software baseada na extração, processamento e visualização de métricas de produto e qualidade retiradas dos artefatos gerados no processo de desenvolvimento de software.

Em especial, temos foco na visualização da evolução histórica e da visualização dos dados de produto e qualidade entre artefatos diferentes, de forma que um possa ser usado como valor preditivo do outro pelo gerente do processo.

Essa abordagem é complementada pela definição de uma plataforma de desenvolvimento, que inclui a escolha de ferramentas adequadas à extração das métricas e pela criação de uma ferramenta que centraliza essas métricas e permite visualizá-las em forma similar a um painel de controle do projeto.

Finalmente, apresentamos um protótipo funcional dessa plataforma, cujo uso é exemplificado com base em um projeto real.

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

AN APPROACH TO SOFTWARE PROJECTS ASSESSMENT

Fabio Perez Marzullo

March/2006

Advisors: Geraldo Bonorino Xexéo,
Ana Regina Cavalcanti Rocha

Department: System and Computing Engineering

This work presents an approach towards object oriented software project assessment, based on extraction, processing, and visualization of product and quality metrics acquired from artifacts developed along the project.

In special, we focus on historical evolution and in order to predict project information to aid project managers.

This approach is complemented by a definition of a development platform, which includes the proper selection of project tools, suited to provide necessary metrics to the created analysis tool, allowing analysis and process assessment.

Finally, we present a study scenario presenting the prototype working process, exemplified by a real project.

CONTEÚDO

Capítulo 1

Introdução

1.1 Motivação

Ao longo dos últimos anos, Indústria e Governos têm dedicado esforços na criação de práticas e padrões de desenvolvimento capazes de melhorar e facilitar o controle dos inúmeros projetos de software que são iniciados a cada ano [1]. Diferentes idéias vêm ganhando força há anos e, em sua maioria, envolvem a utilização de dados históricos em auxílio ao planejamento dos projetos. Em geral, todos têm tentado encontrar meios de reduzir custos e prazos associados ao processo de desenvolvimento [13, 14, 21].

Para atingirem ideais realistas de prazos e custos de projetos de software, diversas técnicas devem ser utilizadas objetivando uma formalização na definição de processos de desenvolvimento que se encaixem tanto com o perfil humano, como com o perfil técnico disponível na empresa. Uma vez identificado e definido esse processo, a empresa passa a conduzir seus projetos de maneira mais objetiva, conseguindo um controle maior das suas atividades, e adquirindo a capacidade de extrair informações de acordo com suas necessidades.

Uma das melhores maneiras de se controlar projetos de software é utilizar um programa de medição contínua do projeto e extrair informações relevantes às necessidades correntes do mesmo. Assim, a gerência por métricas torna-se uma das mais adequadas para controle geral do desenvolvimento. Entretanto, apesar de existirem esforços para criação de ferramentas de controle e apoio a medição, a gerência por métricas também traz uma complexidade muito elevada, o que, em muitos dos casos, acaba impedindo que seu uso beneficie a condução do projeto.

Amparado por estes aspectos, tentamos, com este trabalho, definir uma abordagem de acompanhamento capaz de apoiar atividades gerenciais necessárias em um projeto de software, utilizando, como estrutura básica, uma parte do estudo de métricas. Mais especificamente, criamos uma plataforma de desenvolvimento que permitisse aos gerentes e analistas inferir um determinado nível de conhecimento do real andamento de seus projetos. Projetado para ter o comportamento de um painel de controle, entendemos que o trabalho desenvolvido tenta contribuir, com uma abordagem voltada para utilização

de métricas de produto e qualidade, para um ganho mais significativo do produto em construção.

1.2 Definição do Problema

De acordo com pesquisas realizadas na área de desenvolvimento de projetos de software nos últimos anos, observa-se que, a cada ano, diferentes técnicas de medição aliadas a plataformas e ambientes de desenvolvimento integrados, despontam como solução para os problemas enfrentados por gerentes e analistas de sistemas. Exemplos como o Rational Unified Process – RUP [44] e o TABA [86], que estão há anos aprimorando seus recursos gerenciais, aparecem como excelentes soluções de apoio ao desenvolvimento de software.

Tendo como plano de fundo todo o estudo relacionado à área de métricas e as dificuldades enfrentadas atualmente, pelas equipes desenvolvedoras, no que diz respeito ao controle de seus projetos, destacamos nossa abordagem de visualização e acompanhamento de medidas relacionadas aos artefatos de software, aliada às diferentes metodologias de controle de projetos, métricas de software e padrões de desenvolvimento de sistemas, como uma proposta de avaliação e controle capaz de apoiar o entendimento do andamento de projetos.

Dentro dessa abordagem definimos uma plataforma de desenvolvimento que é apoiada por um conjunto bem definido de ferramentas de apoio, para a aquisição e tratamento das métricas dos artefatos construídos durante o projeto. Como ponto central da plataforma, criamos uma ferramenta de análise de métricas, que deve ser utilizada como um painel de controle capaz de fornecer informações básicas sobre o andamento do projeto.

1.3 Proposta do Trabalho

O trabalho foi desenvolvido em conjunto com os laboratórios de Engenharia de Software e Banco de Dados da Coordenação de Pós Graduação em Engenharia da Universidade Federal do Rio de Janeiro – COPPE/UFRJ.

O foco principal foi a concepção de uma abordagem de desenvolvimento orientado por métricas de produto e qualidade, capaz de auxiliar no controle de projetos, através da análise das métricas extraídas de artefatos de software produzidos ao longo do desenvolvimento, e identificar sua condição esperada no andamento do mesmo. Em conjunto, criamos uma ferramenta que ajudasse a controlar essas informações de forma simples e precisa.

A ferramenta foi denominada ***"PATT – Project Assessment and Tracking Tool".*** Tendo como plano de fundo atividades de medição, princípios de garantia de qualidade e padrões internacionais de desenvolvimento, seu funcionamento enquadra-se na manipulação de métricas referentes a projetos de desenvolvimento de software, onde, através dessa manipulação, a análise de artefatos, semanticamente iguais, capacita o gerente a observar comportamento e extrair informações relevantes sobre o andamento do projeto.

No contexto geral, a ferramenta faz parte de uma plataforma de desenvolvimento de baixo custo, composto por ferramentas "*Open Source*". Tais ferramentas desempenham um papel fundamental no processo de avaliação.

1.4 Estrutura da Dissertação

O **Capítulo II** apresenta um resumo do estudo de métricas. Este resumo servirá como cenário para a criação da abordagem. Relata seu histórico, evolução, seus tipos, objetivos e modelos e as principais métricas encontradas atualmente. Demonstra como estão associadas aos diversos paradigmas de programação, explica alguns dos principais métodos de utilização e relata o que há de mais significativo na área atualmente.

Ferramentas, metodologias e processos são apresentados para demonstrar algumas técnicas de utilização de métricas.

O **Capítulo III** fornece ao leitor embasamento teórico de como utilizar métricas na gerência de projetos. Novamente, este capítulo servirá como base para a construção da abordagem. Percorrendo os modelos de avaliação mais comumente utilizados em atividades de apoio a gerência, estudamos como é possível utilizar métricas na aquisição de informações relevantes ao andamento e controle do projeto.

O **Capítulo IV** descreve a abordagem definida para o processo de avaliação da ferramenta. Estabelece o contexto geral para utilização do protótipo desenvolvido, descreve como a plataforma foi concebida e demonstra como as atividades de avaliação devem ser realizadas para que o objetivo final seja alcançado.

O **Capítulo V** apresenta o protótipo da plataforma de avaliação criada para apoiar a abordagem de avaliação planejada no capítulo anterior. Cada característica desenhada pela abordagem de avaliação aparece como uma atividade dentro da plataforma proposta.

O **Capítulo VI** relata um estudo de caso de um projeto de desenvolvimento de software real, apresentando todas as atividades realizadas na sua avaliação. Avaliamos sucesso e falhas do modelo proposto e identificamos fatores indicativos de melhorias dentro do processo modelado.

Finalizando o trabalho, no **Capítulo VII** apresentamos alguns resultados relacionados com o uso da ferramenta e apontamos algumas dificuldades enfrentadas ao longo do desenvolvimento do trabalho. Além disso, descrevemos oportunidades de melhorias e acréscimos ao processo de avaliação e à ferramenta, oriundas de fatores que surgiram durante a realização da pesquisa.

Alguns apêndices foram incorporados ao texto à título de clareza de alguns temas que são abordados ao longo do trabalho. O **Apêndice A** apresenta um resumo da teoria de banco de dados XML, explicando, de maneira sucinta, o que são e como são utilizados. No **Apêndice B**, um resumo da teoria de XML, padrão amplamente utilizado para armazenamento e troca de informações, é apresentado com intuito de formalizar conceitos utilizados durante o decorrer do trabalho. O **Apêndice C** descreve as ferramentas de apoio utilizadas para desenvolvimento da plataforma e o **Apêndice D**

apresenta um resumo da linguagem XQuery, uma linguagem de consulta utilizada no armazenamento dos dados no banco de dados XML.

Capítulo 2

Métricas

"Quando é possível medir o que se fala e expressá-lo em números, você sabe algo sobre isso; mas quando não é possível medir e quando não se consegue expressar em números, seu conhecimento é insatisfatório; pode ser o início do aprendizado, mas você, em seus pensamentos, pouco avançou ao estágio da ciência".

Lorde Kelvin, 1891

2.1 Introdução

A Engenharia de Software ensina que para desenvolver software com qualidade é necessário obter entendimento quantitativo e qualitativo dos componentes de um processo de desenvolvimento. A avaliação destes componentes necessita de um processo formal de medição e análise, através do uso de técnicas específicas e bem definidas, fornecendo aos gerentes de projeto maior controle de suas atividades.

As dificuldades enfrentadas por gerentes de projetos são mais expostas, principalmente, quando são identificadas necessidades de informação no início de cada novo projeto. Por esse motivo, uma gerência eficiente deve possuir uma filosofia de trabalho onde as principais técnicas e atividades de medição sejam respeitadas e apoiadas por normas e padrões de qualidade para projetos de software, permitindo que os aspetos principais do processo sejam corretamente avaliados ao longo do projeto.

Com base nestes aspectos, o capítulo que apresentamos a seguir, objetiva relatar um resumo do estudo de alguns pontos de interesse na área de métricas. Esse resumo pretende criar um arcabouço teórico, no qual, identificamos a necessidade de especificação e desenvolvimento da abordagem de avaliação e construção da ferramenta.

2.2 Métricas de Software

De acordo com Norman Fenton em [7], uma métrica de software pode ser definida como um objetivo; uma medida matemática do software que é sensível a diferenças de suas características. Em seu primeiro trabalho sobre métricas [5], verifica-se que qualquer métrica de software é uma tentativa de medir ou predizer algum atributo (interno ou externo) de algum produto, processo ou recurso. Portanto, uma métrica nada mais é que uma representação formal do que deve ser medido efetivamente. É um dos principais métodos que o gerente possui para identificar se o projeto se encontra em um estado adequado e esperado de desenvolvimento, além de prever as chances de permanecer focado em seus objetivos. Métricas geram conhecimento e ajudam a

determinar quais ações devem ser realizadas para manter o projeto a salvo de riscos desnecessários.

A medição é importante, principalmente, no entendimento de determinados aspectos do desenvolvimento do projeto:

- Entender o que ocorre durante o desenvolvimento e a manutenção;
- Permitir avaliar as situações correntes, estabelecendo marcos que auxiliam na determinação de ações futuras;
- Manter o controle do projeto nas mãos dos gerentes, pois este pode utilizar os marcos e as avaliações para melhor entender os relacionamentos entre características diferentes;
- Predizer ações futuras;
- E, por fim, dependendo do nível de controle que o projetista ou gerente possua do projeto, as medições podem ajudar a aperfeiçoar pontos específicos levando a um aumento da qualidade, diminuição de custos e cumprimento dos prazos.

A tabela abaixo dá uma visão geral dos atributos que devem ser avaliados de acordo com os artefatos desenvolvidos durante um projeto de software:

Tabela 2.1 Relacionamento entre produtos de software e seus atributos.

Entidades	**Atributos**	
	Interno	**Externo**
Produtos		
Especificações	Tamanho, reuso, modularidade e redundância, funcionalidade, correção sintática etc.	Compreensibilidade, manutenibilidade etc.
Projetos	Tamanho, reuso, modularidade, acoplamento, herança e funcionalidade etc.	Qualidade, complexidade e manutenibilidade etc.
Código	Tamanho, reuso, modularidade, acoplamento, funcionalidade, complexidade algorítmica, estruturas de controle de fluxo etc.	Confiabilidade, usabilidade, manutenibilidade, reusabilidade etc.
Dados de Teste	Tamanho, nível de cobertura etc.	Qualidade, reusabilidade etc.

Tabela 2.2 Relacionamento entre processos e seus atributos.

Processos	Internos	Externos
Construindo Especificações	Tempo, esforço, número de mudanças em requisitos etc.	Qualidade, custo e estabilidade etc.
Projeto Detalhado	Tempo, esforço, número de defeitos encontrados na especificação etc.	Custo etc.
Teste	Tempo, esforço, número de defeitos encontrados no código etc.	Custo, estabilidade etc.

Tabela 2.3 Relacionamento entre recursos do projeto e seus atributos.

Recursos	Internos	Externos
Pessoal	Idade, Salários etc.	Produtividade, experiência, inteligência etc.
Equipes	Tamanho, níveis de comunicação. Estrutura etc.	Produtividade, qualidade etc.
Organizações	Tamanho, Certificações ISO, nível CMM etc.	Maturidade, lucro etc.
Software	Preço, tamanho etc.	Usabilidade, confiabilidade etc.
Hardware	Preço, velocidade, tamanho de memória etc.	Confiabilidade etc.
Escritórios	Tamanho, temperatura, iluminação etc.	Conforto, qualidade etc.

"Métricas de Software" é um termo que engloba diversas idéias, as quais envolvem atividades de medição diferentes [4, 7, 10 e 12]:

- Estimativas de Custo e Esforço;
- Aquisição de Medidas e modelos de produtividade;
- Aquisição de Medidas e modelos de qualidade;
- Coleta de Dados;
- Aquisição de Modelos de Confiabilidade;
- Aquisição de Modelos e Avaliações de Desempenho;
- Métricas Estruturais e Complexas;
- Avaliações de Maturidade e Capacidade;
- Gerência por Métricas;
- Criação de Metodologias e Ferramentas.

Cada uma das atividades citadas impõe o uso de métricas específicas. Em qualquer atividade, conseqüentemente, deve-se estar ciente de que regras devem ser respeitadas para se obter o controle adequado. Tais regras facilitam na consistência das medições, bem como promovem uma base sólida para interpretação dos dados coletados.

O estudo de métricas, conseqüentemente, estabelece estas regras ao criar modelos e padrões para o desenvolvimento de software.

Kafura e Fenton [6 e 7] convergem na idéia de que regras derivadas do estudo de métricas ajudam a estabelecer objetivos fundamentais para as ações de medição. Regras como *Relacionamento Empírico* e *Mapeamento* permitem criar modelos representacionais dos atributos avaliados, permitindo, através dessas relações, representarem em números características das entidades, traduzindo-as do mundo real para um sistema numérico. Também fornecem ferramentas suficientes para a criação sistemática de modelos capazes de visualizar as entidades de maneira simplificada e ajustada às necessidades do projeto, simplificando o controle, e promovendo previsões mais apuradas.

2.2.1 Modelos de Medição

McGarry, em [74], define que um modelo é uma abstração da realidade capaz de remover detalhes desnecessários das entidades avaliadas e dar enfoque somente a conceitos considerados importantes na perspectiva do projeto. Diversos modelos foram criados com o objetivo de ajudar a entender e predizer aspectos do desenvolvimento e, em sua maioria, o chamados *Modelos Estatísticos* dominavam.

Já em [5], verificamos que o estudo de métricas encontra-se dominado por modelos estatísticos denominados *Modelos de Regressão.* Esses modelos promovem um melhor entendimento das métricas, uma vez que mapeiam entidades do mundo real em elementos de um sistema numérico. Por exemplo, a maior parte do estudo de métricas esteve voltado para a criação de modelos capazes de ajudar a definir cursos de ação nas atividades de gerência. Normalmente, este tipo de trabalho envolvia a criação de modelos de regressão conforme os que estão definidos em [7].

Modelos de produtividade discutem métodos para quantificar a produtividade durante um projeto de software. Para a análise e modelagem da produtividade de um indivíduo existe a necessidade de se medir atributos bastante subjetivos, conforme o objetivo da avaliação. Seu modelo mais comumente usado é baseado em uma razão onde o total de artefatos produzidos pelos desenvolvedores é dividido pelo esforço ou custo

gasto para criá-los. A produtividade é vista como uma *medida indireta (apresentada no próximo capítulo)*, onde se calcula seu valor a partir do resultado da razão entre uma medida de um atributo de produto por outra medida de um atributo de processo. Outra forma de avaliação da produtividade está relacionada com a ação de calcular custos ao evitar defeitos antes da entrega com os custos associados à remoção desses defeitos após a entrega do produto. O esforço gasto com testes chega a 30-50% do esforço total de um projeto, conseqüentemente menos defeitos são encontrados após a fase de entrega do produto [53].

Modelos de qualidade, por sua vez, são baseados em medidas relacionadas com o produto e buscam, em sua maioria, inferir a qualidade do processo. Modelos de qualidade e produtividade compartilham certas características uma vez que a idéia de mapear produtividade através da razão entre *defeitos*[1] descobertos antes da entrega do produto e encontrados e corrigidos após sua entrega, já determina as bases para avaliação da qualidade do processo. Os fatores de qualidade normalmente utilizados correspondem a atributos externos do produto; contudo, o critério em que estes são segmentados geralmente corresponde a atributos internos de produto ou processo [13 e 14].

Por conseguinte, modelos de confiabilidade devem ser empregados para se determinar o tempo em que um produto leva para apresentar defeitos (MTTF – *"Mean Time To Fail"* e MTBF – *"Mean Time Between Failures"*).

Dentro dessa perspectiva, diversos tipos de modelos podem ser gerados e utilizados pelo estudo de métricas, e apesar de *Modelos de Regressão,* serem altamente informativos [7], a tendência atual é buscar a criação de *Modelos Causais*, pois estes demonstram como interpretar o comportamento associado aos elementos numéricos quando retornamos nossa visão para o mundo real.

---

1 - Como [12], um defeito ocorre quando as atividades de garantia da qualidade falham em descobrir um erro em um produto produzido durante o processo de desenvolvimento.

Modelos Causais conseguem promover uma estrutura explicativa dos eventos que precisam ser tratados. Conforme Fenton [5], pesquisadores na área de métricas buscam, constantemente, criar modelos que consigam manusear as características abaixo, sem introduzir sobrecargas às métricas utilizadas:

1. Trabalhar com variáveis de produto e processo diversificados;
2. Trabalhar com evidência empírica;
3. Trabalhar com relacionamentos coerentes entre causa e efeito;
4. Incerteza;
5. Informação incompleta.

Um exemplo bastante abrangente e conhecido é o estudo relacionado à *Rede Bayesianas*, que utiliza a representação de grafos, onde os nós representam variáveis e os arcos representam as relações causais associadas a probabilidades de ocorrências. Uma explicação mais aprofundada pode ser encontrada em [5].

Dessa forma perguntas do tipo

"Quanto pode ser diminuído dos recursos disponíveis, se estamos preparados para suportar um produto com qualidade inferior?"

ou

"Para uma especificação de tamanha complexidade, e dado que os recursos são limitados, quais são as chances de produzir um produto com qualidade aceitável?"

são mais facilmente respondidas utilizando-se modelos causais.

Assim, dos benefícios que o estudo de métricas pode trazer, o principal reside em prover informações que possam aumentar a capacidade gerencial de forma qualitativa durante o ciclo de vida do projeto.

2.2.2 Tipos de Métricas

A importância em determinar modelos que contenham informações relevantes para o processo de medição do software demonstra que, uma vez que o modelo esteja disponível, podemos definir métricas em termos das suas informações [8]. A maioria dos exemplos de utilização de métricas para software, emprega um mapeamento direto dos atributos das entidades avaliadas para os sistemas numéricos criados. Em contrapartida, quando as relações entre atributos se apresentam complexas, devido a combinação de várias métricas, torna-se necessário utilizar outros modelos para representar com coerência esses resultados. Assim a estrutura de medidas **Diretas** e **Indiretas** foi criada para distinguir os dois tipos de métricas [12].

Medidas Diretas de atributos de uma entidade envolvem um mapeamento simples dos atributos para o sistema de medição utilizado. Como exemplo a medição do tamanho de uma classe pode ser feito independente de outras classes ou propriedades.

Medidas Indiretas, por sua vez, dependem de outras medidas, em sua maioria medidas diretas, para serem calculadas. Normalmente é uma relação, um cálculo onde as variáveis utilizadas são medidas diretas de atributos da entidade. Como exemplo o cálculo da produtividade de um programador pode ser feito através da razão entre o número de Linhas de Código que foram criadas pelo tempo que este levou para escrevê-las.

Uma outra forma de entender o conjunto de métricas de software apresenta-se na idéia de subdividi-las em **Objetivas** e **Subjetivas**. Ao se executar as medições do software, um grande esforço é feito para manter essas medições na esfera objetiva. Manter a medição objetiva permite que diferentes grupos de pessoas possam repetir as atividades de maneira idêntica, sem que haja resultados inconsistentes.

Eventualmente, existe a necessidade de inserir algum grau de subjetividade na medição. Métricas subjetivas são suscetíveis às pessoas que as julgam; são dependentes da organização em que a pessoa trabalha do seu nível de instrução, inteligência e do seu humor no momento da aferição. Medidas subjetivas possuem grande valor, uma vez que quem esteja medindo entenda suas limitações [18].

O importante ao se analisar medidas subjetivas é perceber o panorama geral que fornecem. A aferição da qualidade, por exemplo, trás uma idéia de como sentir ou perceber o software sendo construído. Dessa forma ambos os tipos de medições devem ser considerados quando se deseja avaliar o projeto.

Uma outra concepção para tipos de métricas estaria relacionada com características apresentadas pelo software. Métricas orientadas ao tamanho do software, orientadas a funcionalidades e a qualidade coexistem e são utilizadas desde o início do estudo do estudo de métricas.

2.2.2.1 Métricas Orientadas ao Tamanho

Ao citar métricas orientadas ao tamanho, idealiza-se de imediato, a utilização de *Linhas de Código*. Independentemente da forma em que esteja inserido no contexto de avaliação é fato que, no final, *Linhas de Código Fonte* sempre são utilizadas como parâmetro de medição de tamanho.

Conforme Kusumoto em [52], métricas Orientadas ao Tamanho são derivadas através da normalização de fatores como qualidade e produtividade. De maneira a determinar quais métricas são representativas, a normalização ocorre a partir da utilização da métrica de *Linhas de Código*. Conforme o exemplo apresentado por [12], suponha que um gerente possua um conjunto de registros conforme a tabela abaixo, medidas indiretas de qualidade e produtividade poderiam ser derivadas para cada projeto:

Tabela 2.4 Dados de projetos e respectivas medidas.

Projetos	LOC	Esforço	Erros	Defeitos	Pessoal
Projeto 1	12.100	24	134	29	3
Projeto 2	27.200	62	321	86	5
Projeto 3	20.200	43	256	64	6

Medições sugeridas:

- Erros por LOC ou KLOC;

- Defeitos por LOC ou KLOC;
- Custo por LOC ou KLOC;
- LOC ou KLOC por pessoa-mês.

Assim o Projeto 1, por exemplo, teria os seguintes resultados:

- Erros por LOC ou KLOC → 134/12.1 = 11.07 Erros/KLOC
- Defeitos por LOC ou KLOC → 29/12.1 = 2,39 Defeitos/KLOC
- LOC ou KLOC por pessoa-mês → 12.1/24 = 0.504 KLOC/Pessoa-Mês

Contudo, como vimos anteriormente, a métrica *Linhas de Código* não é completamente aceita como a melhor forma de medir o processo. Por isso, sua utilização fica a cargo do engenheiro de software responsável pelo projeto.

2.2.2.2 Métricas Orientadas a Funcionalidades

Quando são avaliados conjuntos de métricas orientadas a função, as métricas por *Ponto de Função* são as que melhor representam esse tipo de técnica [18, 52 e 53].

Uma vez calculados, os *Pontos de Função* são usados de forma análoga a *Linhas de Código,* portanto as normalizações que se seguem representam os mesmos objetivos quando *Linhas de Código* são utilizadas:

- Erros por PF;
- Defeitos por PF;
- Custo por PF;
- PF por pessoa-mês

2.2.2.3 Métricas Orientadas a Qualidade

A qualidade de produtos de software é extraída através da medição do nível de corretude e completudes do desenvolvimento das funcionalidades concebidas como "necessidades ao usuário". A qualidade, portanto, está enraizada no nível de satisfação do cliente. Para alcançar esse nível de satisfação, é necessário que um conjunto de características internas e externas ao produto esteja presente, e assim, permita que a avaliação ocorra [4].

Ao longo da história, diversos modelos de qualidade foram propostos com esse objetivo [15]. Uma das principais normas é a ISO 9126 [13] que surgiu como modelo para qualidade de produto. Complementada pela a ISO 14598-5 [59] que foi criada com o objetivo de estabelecer regras para garantia da aplicação das atividades definidas pela ISO 9126. Ambas são ferramentas poderosas no apoio a atividades de garantia da qualidade de produtos de software.

Avaliar características do produto para inferir qualidade surgiu conforme apresentado no artigo escrito por McCall em [11]. Neste, os principais fatores de qualidade são enumerados e relacionados com métricas de software. As características conforme McCall as definiu são apresentadas no modelo definido pela ISO 9126, demonstradas na figura 1.1.

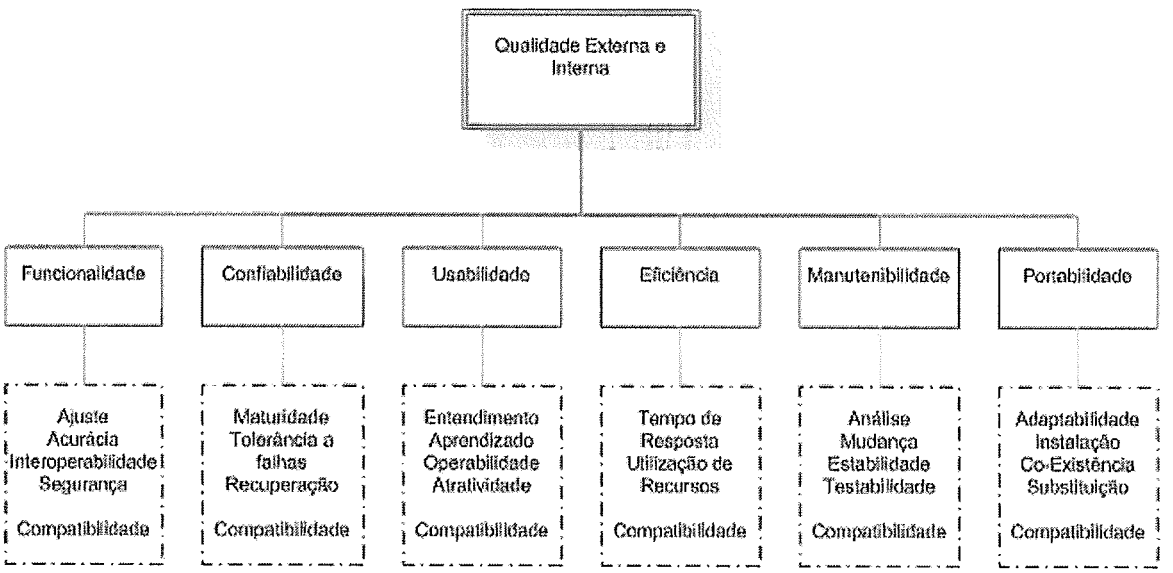

Figura 2.1 Características e sub-características de qualidade de produtos de software.

2.2.2.4 Métricas Orientadas a Objetos

No histórico apresentado no início do capítulo, alguns dos principais fatos que compuseram a evolução do estudo de métricas orientadas a objetos são apresentados. Lorenz e Kidd criaram um conjunto de métricas para projetos Orientados a Objetos, que trouxeram o primeiro avanço na avaliação e planejamento para desenvolvimento deste tipo de software [16]. Em seguida, vemos Chidamber e Kemerer [17], formalizarem um conjunto detalhado de seis métricas que propunham benefícios bastante consideráveis para a área gerencial.

O problema proposto possuía o objetivo de criar métricas Orientadas a Objetos que realmente pudessem ser utilizadas em auxílio às atividades gerenciais. Este problema é resolvido através do mapeamento inicial de um conjunto candidato básico de métricas, e por fim, através de avaliações de suas propriedades, surge a definição do conjunto definitivo que contém as seguintes métricas: (1) *Weighted methods per class (WMC)*, (2) *Depth of Inheritance Tree (DIT)*; (3) *Number of Children (NOC)*, (4) *Coupling Between Objects (CBO)*; (5) *Response for a Class (RFC)*, (6) *Lack of Cohesion in Methods (LCOM)*.

O conjunto possui grande influência em técnicas de gerência para projetos Orientados a Objetos atualmente Em particular, as métricas apresentadas deixaram de ser uma tentativa de formalização do estudo de métricas para projetos Orientados a Objetos, e passaram a ser uma das principais ferramentas existentes para o engenheiro de software no desenvolvimento de sistemas.

2.2.2.5 Complexidade Ciclomática

A métrica de *Complexidade Ciclomática* faz parte de um conjunto mais abrangente de métricas de software. Estas métricas são projetadas para calcular a complexidade dos fluxos de controle de programas. Em sua maioria são construídas com apoio do estudo de grafos [5].

Originalmente desenvolvida por McCabe [56 e 57], a métrica de complexidade ciclomática possui a capacidade de medir quantitativamente níveis de confiabilidade e

facilidade de teste. Também demonstram que indicadores de tamanho podem ser calculados através da sua utilização.

McCabe, em [55], e Pressman, em [12], identificam uma série de fatores para o uso de métricas de complexidade. A lista a seguir identifica tais fatores:

1. Métricas de Complexidade podem ser usadas para produzir informações críticas sobre confiabilidade e manutenibilidade;
2. Provêem informações ao longo do projeto que ajudam no controle de suas atividades;
3. Nas fases de teste e manutenção, ajudam a identificar pontos potenciais de instabilidade.

2.3 Métricas em Trabalhos Recentes

Nas últimas décadas o tamanho e a complexidade dos programas têm aumentado. Tornou-se importante garantir que o produto gerado possua a qualidade esperada a um custo adequado, sem comprometer seu prazo de entrega.

Na contínua busca pela qualidade, esforços na área demonstram uma evolução satisfatória; o advento de padrões, normas e técnicas de desenvolvimento contribuíram para o aumento de casos de sucesso [52]. Métricas de software, portanto, são utilizadas para garantir e especificar os seguintes atributos de projeto: (1) Qualidade; (2) Custo; (3) Esforço; (4) Tempo; (5) Tamanho; (6) Tecnologia.

Cada item representa elementos essenciais de conhecimento para gerentes que devem ser investigados com o objetivo de identificar ações dentro do projeto.

2.3.1 Qualidade

Até bem pouco tempo, a qualidade de produtos de software tinha uma associação com as atividades de testes desenvolvidas durante o projeto. De acordo com modelos de confiabilidade definidos ateriormente, medir a qualidade através de defeitos encontrados no software era objeto principal dos testes [12].

Claramente, durante a fase de testes, métricas de software são particularmente úteis. Com dados extraídos das medições, diversos atributos de qualidade são revelados e avaliados, conseqüentemente atributos como tempo e pessoal podem ser redefinidos diminuindo o esforço necessário para ajustar deficiências dos projetos [20]. Como Shooman apresenta em [53], esforços de teste contam como 30-50% de todo o desenvolvimento, aumentar a eficiência na aplicação dos testes diminui o esforço total de desenvolvimento.

Contudo, a busca pela qualidade não se resume a atividades de teste. Rocha *et. al.* discute em [15] que para garantir a qualidade do produto é necessário garantir a qualidade do processo de desenvolvimento. Kusumoto [52] também discute que o planejamento do projeto deve ser feito de forma a avaliar fenômenos indesejáveis que podem ocorrer durante o processo.

Esforços para garantia da qualidade culminaram com a criação de normas para avaliação de processos e produtos de software. O conjunto de ISO/IEC 9000 aliadas às ISO/IEC 12207 [14], 9126 [13] e 14598-5 [59], entre tantas outras, criam recursos teóricos que permitem gerentes e desenvolvedores estimar, avaliar e garantir que os artefatos de software desenvolvidos sob processos baseados nessas normas contivessem a qualidade esperada por clientes e usuários. Aliado a padronização apresentada, observa-se o surgimento de diversos trabalhos contendo visões e metodologias inovadoras na utilização de métricas conhecidas, bem como apresentam novas métricas para nichos específicos de desenvolvimento.

Patenaude em [61] verifica que, técnicas para avaliação da qualidade utilizando métricas conhecidas como *Linhas de Código* (LOC) e *Pontos de Função* (PF) em conjunto com normas de garantia de qualidade como a ISO 9126 [13], são avaliadas, obtendo resultados satisfatórios. Em [60] uma proposta de mudança de paradigma é detalhada quando métricas recebem uma nova categorização observando uma maior facilidade de manipulação.

O estudo de caso apresentado por Ping Yu em [20] demonstra como métricas são utilizadas para identificar probabilidades de falhas em projetos orientados a objetos. No processo, um conjunto de dez métricas relacionadas com: tamanho, acoplamento, coesão, herança e reuso; é utilizado e verifica-se que ao determinar as probabilidades de falhas

dos projetos, consegue-se prever ações corretivas e, conseqüentemente, aumenta-se a qualidade do produto.

Tantos outros trabalhos desenvolvidos recentemente expressam o interesse e a necessidade de extrair informações que se provem úteis aos gerentes quando do planejamento do projeto. Técnicas para melhorar os procedimentos de planejamento de projetos de software são apresentadas por Alkadi, Systä, Chapeaux, Scotto e Dufour em [63, 64, 65, 66 e 67], entre outros.

Um ponto de vista interessante para verificação de qualidade é visto em [67]. Ao determinar estabilidade de modelos de projeto em classes Java, busca-se garantir que o produto final gerado possua qualidade.

A busca da qualidade, portanto, apresenta-se como uma das principais atividades necessárias no ciclo de vida de um projeto. Através de testes ou medições, deve-se ter em foco a construção de bons produtos e, sobretudo, a satisfação do cliente.

2.3.2 Custo, Tempo e Esforço

Voltados mais para atividades referentes ao processo de desenvolvimento do que propriamente do produto, cálculos de custo, esforços e tempo são, sabidamente, discutidos desde os primórdios do desenvolvimento de software.

Exemplo conhecido é a ferramenta COCOMO [40]. Neste, modelos baseados em dados históricos tentam determinar valores adequados para as variáveis acima, de acordo com o tipo de software construído.

Conforme a engenharia de software se desenvolve, melhores se tornam os modelos criados para avaliação de custo, tempo e esforço, aumentando a capacidade de gerentes e desenvolvedores de controlarem suas atividades. Conseqüentemente essa evolução influencia na descoberta de parâmetros e informações essenciais para os cálculos.

2.3.3 Tamanho

Ao se pensar em tamanho de programas, rapidamente duas métricas despontam como peças chave: *Linhas de Código* (LOC) e *Pontos de Função* (PF). Apesar de toda controvérsia que cerca as duas métricas, ainda constituem o que há de mais comum no cálculo de tamanho.

Exemplo claro é descrito em Kusumoto em [52]. O trabalho apresenta uma proposta de medição automática de Pontos de Função para programas desenvolvidos com linguagem de programação Java. Entende-se que, além de estimar tamanho, também se consegue calcular esforço necessário para desenvolvimento do software. O tamanho medido, portanto, mantém-se constante uma vez que os cálculos são baseados em funcionalidades do sistema, as quais pouco se alteram ao longo do projeto.

Dufour em [66] apresenta um conjunto de métricas que beneficiam o entendimento do comportamento do programa de forma concisa e precisa. Este conjunto torna essa visão concreta e sujeita às validações experimentais. Ao longo da discussão, demonstra-se que LOC são utilizadas como métricas para cálculo do tamanho do programa.

Diversos outros trabalhos recentes apresentam Linhas de Código Fonte e Pontos de Função como métricas necessárias para o cálculo de tamanho. Nestes, diferentes formas de utilização das métricas são descritas para apoiar os cálculos necessários para o controle de projetos [62, 64 e 69].

2.3.4 Tecnologia

Atualmente computadores desempenham papel crucial na vida de todos os seres humanos. Tamanha dependência impõe a necessidade de se produzir software com o máximo de qualidade. Apesar da grande variedade de métricas existentes na área de desenvolvimento de software e das diferentes formas com que são aplicadas, uma tendência se mantém constante: métricas produzem melhores resultados em conjunto.

Este raciocínio é exposto por Mens em [69]. Atualmente um número significativo de pesquisadores investiga como métricas podem ser aplicadas em artefatos de software

em constante evolução. Também demonstra como métricas devem ser utilizadas em conjunto para que se consiga efetuar avaliações adequadas do software, produzindo resultados confiáveis.

Conforme explicado por Scotto em [65], o estabelecimento de programas de medição de software, utilizando conjuntos conhecidos de métricas, são organizados de forma a criar técnicas inovadoras que permitam alcançar melhores resultados ao longo do desenvolvimento.

Notadamente, todas as atividades envolvidas utilizam métricas que refletem três aspectos primários:

- Entender e modelar processos e produtos de software;
- Ajudar a gerência dos projetos;
- Guiar e alcançar melhorias no desenvolvimento.

Assim, independente do tipo de tecnologia utilizada, os objetivos a serem alcançados estão sempre alinhados.

É fato que questões como tecnologias devem ser levadas em consideração, pois projeto onde o tipo de software influencia o desenvolvimento, claramente influencia o modo como devem ser medidos e avaliados.

Assim sendo Dhyani *et. al.* em [70] demonstrasse essa face do estudo de métricas. A importância de medir atributos de objetos conhecidos há tempos é reconhecida como fator crucial ao entendimento do projeto. Para tanto a necessidade de medir aspectos específicos do projeto varia de acordo com a tecnologia. O trabalho expõe diferentes métricas utilizadas em projetos onde o foco de desenvolvimento eram softwares criados para a internet.

Kusumoto e Dufour em [53 e 67] demonstram como métricas propostas e conhecidas se mesclam e são capazes de fornecer informações cruciais em projetos onde a tecnologia utilizada baseia-se em linguagem de programação Java.

Apesar do crescimento e da evolução constantes das técnicas de medição e análise, observa-se que quase todo esforço empregado gira em torno do planejamento e do controle do projeto. Sem estes dados em mãos, gerentes e analistas enfrentam grandes riscos ao longo do projeto.

2.4 Conclusão

O capítulo apresentou algumas das principais atividades que compõem o estudo de métricas. Métricas permitem gerentes promoverem melhorias no processo e no produto gerado em suas empresas. Permitem avaliar a qualidade de um produto através de indicadores extraídos das medições, e definir cursos de ações necessárias, baseados nos cálculos extraídos de atributos de artefatos produzidos ao longo do desenvolvimento.

Os diversos tipos de métricas existentes, desde métricas simples como *Linhas de Código - LOC* até métricas para avaliação de qualidade foram apresentados e discutidos, fornecendo uma visão objetiva do que existe atualmente na área de métricas.

Medidas são relevantes, sobretudo quando vistas através de modelos. Os Modelos apresentados demonstram como métricas podem ser utilizadas de maneira a traduzirem resultados capazes de auxiliar ações corretivas.

O capítulo também apresentou o um pouco do que vem sendo estudado atualmente na área de métricas. As principais tendências apontam para a criação de conjuntos capazes de fornecerem dados mais precisos e sólidos.

Pode-se concluir que para buscar um desenvolvimento satisfatoriamente controlável existe a necessidade de adquirir e gerenciar o máximo de informação sobre o projeto. Assunto do próximo capítulo, a correta gerência das métricas e os resultados obtidos de suas medições podem fazer a diferença entre um projeto de sucesso ou fracassado. No próximo capítulo apresentamos como as atividades gerenciais podem ser conduzidas ao utilizarem algumas das técnicas descritas neste capítulo. A gerência por métricas é uma atividade que envolve disciplina e conhecimento. Deve-se ter em mente que gerenciar sem medir transforma atividades simples em potenciais fontes de risco.

Capítulo 3

Gerência por Métricas

Um dos maiores problemas enfrentados por gerentes de projetos, em atividades de desenvolvimento de software, reside na deficiência de enxergar que a natureza deste tipo de produto é altamente abstrata, não palpável.

Para lidar com esse tipo de problema, o capítulo a seguir descreve como projetos de desenvolvimento de software podem, e devem ser conduzidos através de atividades bem definidas de gerência.

Gerenciar, com o uso de métricas, fornece aos gerentes informações privilegiadas, de maneira a permitir um bom planejamento dos aspectos envolvidos no processo de desenvolvimento.

3.1 Introdução

Gerenciar significa controlar um conjunto de atividades. Conforme definido pelo Instituto de Gerenciamento de Projetos (PMI) [26], o gerenciamento de projetos envolve a aplicação de conhecimento, capacidade, ferramentas e técnicas em atividades do projeto, buscando atender seus requisitos. Gerenciar projetos significa conduzir um conjunto de atividades visando a realização de um bem maior.

Em processos de gerenciamento de software, algumas atividades aparecem como essenciais para o controle efetivo do projeto: (1) Iniciação (2) Planejamento; (3) Execução; (4) Controle; (5) Conclusão.

Essas atividades são necessárias quando se objetiva alcançar resultados de sucesso, pois abrangem atributos obrigatórios ao desenvolvimento do projeto.

Dentro desse aspecto, nos últimos vinte anos, a atividade de medição se desenvolveu e alcançou uma condição de *"boa prática"* de gerenciamento. Sua capacidade em obter informações relevantes ao projeto tem-se apresentado bons resultados para garantia da qualidade dos produtos desenvolvidos, pois ajudam e apóiam gerentes de projeto em decisões relevantes dentro da empresa [44].

Apesar de ser reconhecida como boa prática de gerenciamento, encontramos, atualmente, resistências por parte das empresas em utilizar técnicas de medição. Essa resistência ocorre devido ao fato de que a implantação de programas de medição e análise necessita de pessoal altamente qualificado, bem como de metodologias de desenvolvimento consistentes e padronizadas. Ou seja, implantar programas de gerenciamento adequados e bem sucedidos ainda é uma tarefa desafiadora, porque a maioria dos envolvidos no processo não possui formação adequada necessária para a correta condução de processos de medição. Além disso, a existência massiva de métricas e técnicas de medição contribui para sobrecarregar de informações os responsáveis pelos projetos.

Apesar do cenário caótico, atualmente existem diversas metodologias de desenvolvimento que ajudam na implantação de programas de medição. Esses programas possuem o objetivo de prover, aos gerentes, informações confiáveis sobre as atividades executadas pelo processo de desenvolvimento. Produtos de software requerem atividades

complexas que, sem o devido controle do desenvolvimento, correm o risco de não serem produzidos adequadamente.

Ao longo do capítulo, são apresentados metodologias e modelos de gerenciamento, os quais utilizam técnicas de medição e análise para controlar o processo de desenvolvimento. O objetivo é entender as bases do estudo de gerenciamento, apresentar algumas técnicas utilizando métricas e complementar o arcabouço teórico do capítulo anterior que inicia a criação do plano de fundo na qual a abordagem de avaliação se encaixa.

3.2 Gerência por Métricas

O cenário atual do estudo de processos de desenvolvimento de software demonstra que iniciativas de gerenciamento através do uso de métricas de software, tendem a ser cada vez mais utilizadas. O advento de técnicas de identificação de necessidades e de criação de métricas, atualmente, permite ao gerente do projeto definir, baseado no tipo de software envolvido, atividades de medição que o ajudarão a definir programas de medição que atendam aos requisitos do seu projeto [76, 77]. Dependendo do paradigma de desenvolvimento utilizado, processos de software costumam possuir caráter evolutivo, conseqüentemente, executam suas atividades em iterações pré-determinadas e bem planejadas. Através de métricas corretamente organizadas e bem definidas, alguns procedimentos, como identificação da produtividade da equipe e a qualidade do produto, podem ser avaliados sem impor ao desenvolvimento uma complexidade excessiva em sua gestão [71].

Dentro deste cenário, observam-se diversos movimentos direcionados a criação de modelos de avaliações capazes de prestar auxílios às empresas na formulação de processos de desenvolvimento bem definidos e maduros. Modelos como o CMMI (*Capability Maturity Model Integration*) [21], PSM (*Practical Software Measurement*) [44], responsáveis em grande parte pelo surgimento da ISO/IEC 15939 para processos de medição de software, as normas ISO/IEC 9126 [13] e 14598 [59] que definem atributos de qualidade de produtos de software e modelos de avaliação de produtos de software respectivamente, a ISO/IEC 12207 [14] que define atividades relevantes em processos de

desenvolvimento de software e, mais recentemente, o guia MPS-BR que define um modelo de processo de desenvolvimento de software brasileiro fornecem técnicas de gerenciamento baseados nas melhores práticas existentes atualmente.

Atividades de medição, avaliação e análise estão definidas de acordo com o perfil de cada modelo. A escolha de qual modelo seguir e quais métricas devem ser utilizadas depende do produto a ser desenvolvido.

Conforme sugerido por McGarry em [74], para se obter o conhecimento real e preciso sobre um projeto de software, é importante utilizar técnicas de medição de acordo com a natureza do projeto. A maioria dos projetos é desenvolvida visando o cumprimento de metas em termos de custo, prazos, qualidade e funcionalidades. Conseqüentemente, a correta utilização das atividades de medição, visando a aquisição de informações relevantes ao controle do projeto, depende do modelo de processo de desenvolvimento utilizado.

3.2.1 Capability Maturity Model Integration (CMMI)

O modelo CMMI (*Capability Maturity Model Integration*) [21] apresenta um conjunto de áreas e subáreas que contém atividades que devem ser realizadas dentro de um processo de desenvolvimento e que servem para definir diferentes níveis de maturidade de processos de desenvolvimento de software.

O CMMI é dividido em cinco níveis de maturidade cada um determinando aspectos de desenvolvimento necessários à empresa desenvolvedora que almeja possuir um processo de desenvolvimento com qualidade.

O nível 1, definido como **Inicial**, determina que processos nesse nível são basicamente *ad-hoc* e caóticos. A organização tem como prática um desenvolvimento dependente de desenvolvedores altamente qualificados, e é desprovida de um padrão de desenvolvimento. Costumam estourar seus orçamentos e atrasar a entrega de seus produtos. Empresas em nível 2 possuem um processo de desenvolvimento chamado de **Gerenciado.** Neste nível a empresa já é capaz de gerenciar seus requisitos, planejar e executar seu desenvolvimento de forma padronizada. Em conjunto, algumas técnicas de medição e análise são utilizadas para avaliação do processo e do produto gerado.

Em um nível de maturidade 3, chamado de **Definido,** a empresa já possui um processo de desenvolvimento bem arrumado e padronizado. Já consegue controlar adequadamente todos os aspectos dos níveis anteriores e consegue gerenciar seus objetivos de maneira repetível e unificada. Já no nível 4, tido como **Quantitativamente Gerenciável,** a empresa deve ser capaz de gerenciar todos os aspectos dos níveis anteriores e buscar aspectos quantitativos de qualidade para aprimorar suas atividades em busca de melhores resultados no desenvolvimento. Técnicas de medição e análise são largamente empregadas em modelos estatísticos para identificar objetos de melhorias e oportunidades de inovação.

Em seu último nível, nível de maturidade 5, **Otimizado,** a organização ou empresa foca na contínua melhoria de seus processos buscando aumentos de desempenho tanto no processo quanto no produto.

Conforme os aspectos apresentados, uma organização deve promover a contínua melhoria das suas atividades de gerência para conseguir alcançar seus objetivos de qualidade. A gerência de projetos de software, através de métricas, é reconhecida como uma prática essencial para o sucesso do desenvolvimento de sistemas. Essa notoriedade tornou-se mais forte a partir da inclusão da Área de Processo de Medição e Análise no nível dois de maturidade do CMMI.

O nível dois de maturidade determina que toda empresa que planeje certificar seus processos de desenvolvimento deve antes implantar e sustentar uma capacidade de medição, capaz de suprir suas necessidades de informação. Para tanto, a área de processo de medição e análise define as seguintes atividades:

1. Especificar os objetivos de medição e análise de forma a se alinharem com necessidades de informação e objetivos.
2. Especificar medidas, coleta de dados e mecanismos de armazenamento, técnicas de análises, e mecanismos de propagação da informação.
3. Executar atividades de coleta de dados, armazenamento, análise e propagação da informação.
4. Extrair resultados objetivos que possam ser usados para apoiar decisões e ações corretivas ao longo do projeto.

A integração dessas atividades se dá através das atividades normais de gerenciamento de projetos:

1. Planejamento e estimativa de objetivos.
2. Comparar desempenho real com estimativas iniciais.
3. Identificar e resolver contratempos relacionados aos projetos.
4. Criar meios de incorporar as medições em processos futuros.

Conforme estabelecido pela área de processo de medição e análise, o foco inicial para a utilização de métricas, normalmente, gira em torno do projeto, contudo o CMMI ainda discute que tais práticas podem ser avaliadas e executadas em níveis organizacionais. Para tanto, é necessário estabelecer o escopo relevante à medição, e assim aplicar as metas e práticas adequadas para cada necessidade.

Em termos de projetos de software, as metas especificamente estabelecidas para a implantação e execução de atividades de medição são definidas abaixo:

1. Alinhar medidas e atividades de análise.
2. Prover resultados de medições.
3. Institucionalizar um processo gerenciado de medição.
4. Estabelecer um processo definido (Não obrigatória no nível dois).
5. Coletar informações que promovam melhorias (Não obrigatória no nível dois).

As metas descritas fornecem um indicativo básico das práticas necessárias a uma gerência, através da utilização de métricas. A obtenção de informação relevante ao controle do projeto depende da realização das metas estabelecidas. Cada meta, por sua vez, possui seu próprio conjunto de práticas, que, quando executadas corretamente, permitem avaliar o andamento do projeto em desenvolvimento.

A utilização de métricas na gerência de projetos de software é definida pelo CMMI por diferentes conjuntos de atividades, que seguem uma metodologia de gestão rígida e ordenada. A primeira atividade a ser realizada deve definir o conjunto de métricas que deve ser usado para que a informação seja produzida de maneira coerente e confiável. Objetivos e necessidades de informação devem documentar motivos para a existência do processo de medição e análise, bem como definir o nível de controle

desejado sobre o projeto. Este conjunto de objetivos é refinado e quantificado de maneira a permitir uma escolha acurada das métricas utilizadas na avaliação. O conjunto de métricas estabelecido deve permitir que as necessidades de informação sejam supridas corretamente.

Após a execução da medição, análises são realizadas visando avaliar eventuais distorções no processo. Análises adicionais são conduzidas de acordo com as necessidades do projeto e, seus resultados, são revisados pelos responsáveis pelo projeto. Os resultados são armazenados em históricos através de mecanismos de armazenamento estabelecidos durante o planejamento do processo de medição e consultas são realizadas de acordo com as políticas de controle de acesso as informações guardadas.

Gerenciar projetos com o auxílio de métricas conforme definido pelo CMMI adequa-se ao planejamento e execução de um conjunto de atividades e práticas de desenvolvimento, necessárias, porém não exaustivas, para controle do processo. Através dessas atividades, o gerenciamento de projetos torna-se uma atividade repetível e padronizada, que permite avaliar as necessidades de melhorias e o estado atual do projeto.

3.2.2 A Norma ISO/IEC 12207

Desenvolvida com o principal objetivo de padronizar processos de desenvolvimento de software, a norma ISO/IEC 12207 [14], criada pela Organização Internacional para Padronizações (ISO) e pela Comissão Eletrotécnica Internacional (IEC), define como processos de desenvolvimento de software devem ser conduzidos visando um desenvolvimento padronizado, controlado, e gerenciado. Esta estabelece uma arquitetura comum para ciclos de vida de processos de software amparada por atividades, tarefas e processos organizacionais, que, em conjunto com as emendas 1 e 2, proporcionam um entendimento qualitativo das atividades que devem ser executadas pelas empresas ao longo do desenvolvimento de um projeto.

Dividida em três classes de processos, a norma define um conjunto de atividades que devem ser consideradas quando se deseja definir um processo de desenvolvimento adequado para o produto e ou projeto que se deseja desenvolver.

- Processos Fundamentais de Ciclo de Vida – Tratam do processo de aquisição, de fornecimento, desenvolvimento, operação e manutenção.

- Processo de Apoio de Ciclo de Vida – Trata dos processos de documentação, gerência de configuração, garantia da qualidade, verificação, validação, revisão conjunta, auditoria e resolução de problemas.
- Processos Organizacionais de Ciclo de Vida – Tratam do processo de gerência, da infra-estrutura, do processo de melhoria e do treinamento.

Dentro da estrutura da norma, dois processos são alvos de interesse deste trabalho: o Processo de Garantia da Qualidade e o Processo de Melhoria. Ambos destacam-se por suas características comuns na intenção de uso de métricas e técnicas de medição para a correta condução de suas atividades.

O processo de garantia da qualidade determina que todo o processo de desenvolvimento esteja em perfeita conformidade com requisitos especificados e planos estabelecidos. Deve garantir a correta condução das avaliações do produto de software e garantir que a satisfação do cliente seja alcançada em níveis desejáveis. Também impõe que atividades de medições do produto e do processo sejam realizadas e documentadas de acordo com padrões e procedimentos estabelecidos.

Em se tratando do processo de melhoria, as atividades de medição e análise se apresentam de forma mais contundente, onde se estabelecem processos de avaliação, medição, controle e melhoria dentro do ciclo de vida do projeto. A norma é flexível ao exercer abordagens específicas de engenharia de software para condução das atividades e processos especificados. Entretanto, a correta escolha do método de avaliação depende do nível de capacitação do indivíduo e na tecnologia utilizada.

A emenda I da ISO/IEC 12207 (2002) trata as atividades de medição como atividades permanentes dentro do processo de gerenciamento da ISO/IEC 12207. Esta define que o propósito das atividades de medição deve estar alinhado com técnicas de coleta e análise de dados referentes aos produtos desenvolvidos ao longo do projeto. Esta aderência deve garantir um suporte mais efetivo ao gerenciamento dos processos de desenvolvimento e apontar objetivamente a qualidade dos produtos produzidos.

3.2.3 As Normas ISO/IEC 9126 e ISO/IEC 14598

Produtos de software devem ser avaliados de forma a embasar a criação de modelos de controle de desenvolvimento adequados às necessidade da empresa. Ao avaliar o processo e o produto, insere-se um modelo de gestão altamente capacitado a produzir software com qualidade. Tanto o controle do processo como o desenvolvimento do produto, contribuem para a implantação de modelos de gerência alinhados com os aspectos de garantia de qualidade, conseqüentemente, contribui-se para o aumento da satisfação do cliente.

Alinhadas com esses objetivos, a norma ISO/IEC 9126, descreve características e sub-características de qualidade e modelos de avaliação com o objetivo de padronizar os critérios de avaliação relacionados ao desenvolvimento do produto de software.

Conforme a figura 3.2, a ISO/IEC 9126 define seis características de qualidade, divididas em outras sub-características, que representam aspectos de qualidade de um produto de software. Ambas são avaliadas através de atributos internos e externos do software e fornecem um modelo de avaliação altamente qualificado para apoiar decisões gerenciais.

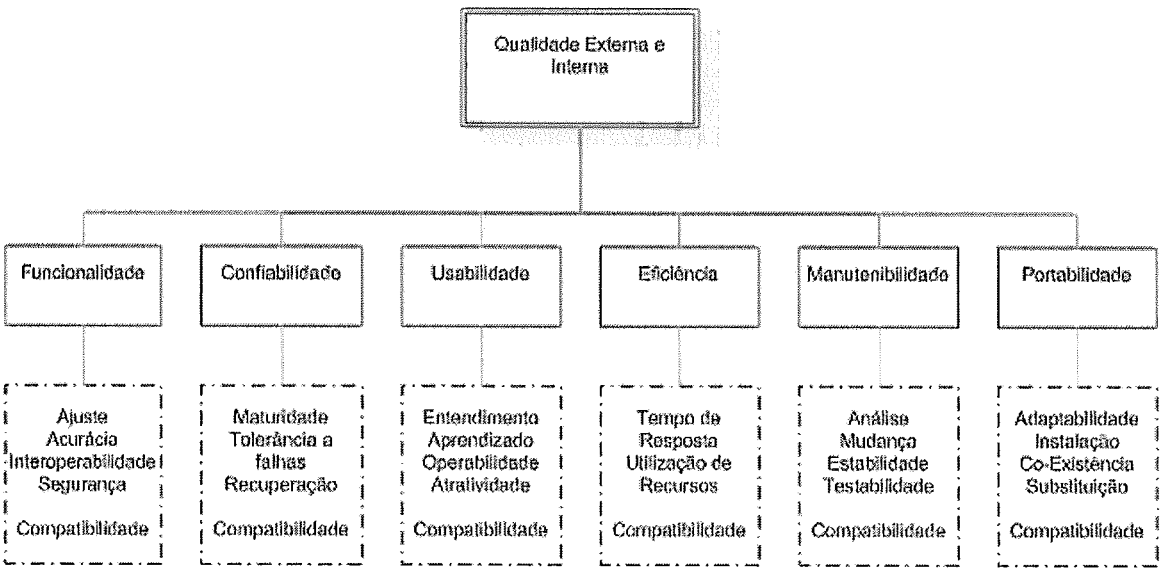

Figura 3.1. Características e Sub-Características de produtos de software.

A ISO/IEC 9126 também descreve um modelo de qualidade que utiliza técnicas de avaliação através do uso de métricas, conforme apresentadas nas subdivisões da norma (ISO/IEC 9126-2, 9126-3 e 9126-4). A figura 3.3 apresenta como esse modelo pode ser utilizado dentro do processo de avaliação da organização.

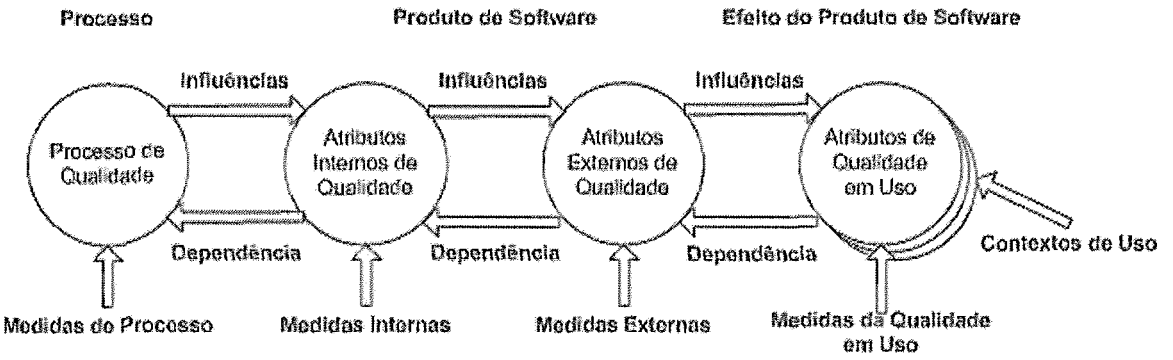

Figura 3.2. Modelo de Qualidade da ISO 9126.

Definindo grupos distintos de atributos de qualidade é possível gerenciar os diferentes aspectos relacionados à construção do produto. A qualidade do produto, portanto, pode ser avaliada através da medição desses atributos, internos e externos, de acordo com o contexto em que o desenvolvimento está inserido. Toda avaliação deve ser feita mediante a predefinição de objetivos claros de medição, pois falhas na correta identificação desses objetivos podem produzir resultados destoantes ou sem significado, o que oferece um risco à gerência, pois podem se basear em informações inválidas no momento de alguma tomada de decisão crítica para a empresa.

Em conjunto com a ISO/IEC 9126, a ISO/IEC 14598 apresenta um modelo de avaliação que utiliza todos os aspectos internos e externos, definidos na ISO/IEC 9126, e descreve um procedimento que permite determinar a qualidade do produto em desenvolvimento. Essa avaliação leva em consideração os inúmeros artefatos representativos do projeto em análise e os avalia conforme as características definidas pela ISO/IEC 9126. A avaliação da qualidade do produto se torna factível quando atributos de baixa abstração do projeto são medidos. Somente neste contexto o avaliador, no caso o gerente, conseguirá obter informações relevantes ao processo de desenvolvimento e ao produto.

O processo de avaliação, portanto, consistem em um conjunto de atividades conduzidas com o objetivo de entender o processo de desenvolvimento. Tais atividades possuem objetivos diferenciados de acordo com a avaliação em andamento e, ainda conforme a natureza do produto em questão. Portanto, o principal objetivo do processo de avaliação da ISO/IEC 14598 é garantir que as características definidas a seguir sejam atendidas ao longo da execução do processo:

- Repetibilidade: avaliações distintas de um mesmo produto, que sigam as mesmas especificações e busquem os mesmos resultados e que são executadas pelos mesmos avaliadores, devem produzir resultados idênticos.
- Reprodutividade: avaliações distintas de um mesmo produto, que sigam as mesmas especificações e busquem os mesmos resultados e que são executadas por avaliadores diferentes, devem produzir resultados idênticos.
- Imparcialidade: qualquer avaliação submetida as regras definidas pela norma não podem conter resultados tendenciosos.
- Objetividade: todo e qualquer resultado extraído das avaliações devem ser obtidos de maneira imparcial, longe de opiniões e sugestões de seus avaliadores.

Neste processo, a avaliação é composta de cinco atividades:

- Definição dos requisitos de avaliação;
- Especificação da avaliação de acordo com os requisitos levantados;
- Modelagem do plano de avaliação, conforme a especificação definida;
- Execução do plano de avaliação dentro de um ambiente imparcial e orientado a resultado;
- Conclusão da avaliação através de um relatório do trabalho executado.

Estas atividades definem como o processo de avaliação deve ser executado e quais os resultados que devem ser alcançados no final da execução.

A utilização das normas em conjunto determina um *"framework"* de avaliação poderoso que estabelece atributos de medição e normativas de procedimentos de análises

que devem ser realizados dentro de um processo de desenvolvimento que almeja a criação de produtos com qualidade e de baixo custo.

3.2.4 A Norma ISO/IEC 15939 para Processos de Medição de Software

O principal objetivo do padrão ISO/IEC 15939 [75] é definir atividades e tarefas necessárias para identificar, definir, aplicar e melhorar processos de medição de software dentro de estruturas organizacionais de medição.

O documento descreve que o processo de medição de uma organização deve executar um conjunto mínimo de ações, com o intuito de obter o máximo de conformidade e controle na gerência do projeto. Embora cada ação defina um escopo aceitável de execução, a ISO/IEC 15939 agrupa todas estas em quatro macro-atividades, as quais estabelecem a capacidade de medição, planejam as medidas, executam o processo de medição, e avaliam seus resultados. Cada macro-atividade é composta por diferentes tarefas, cada uma espelhando uma ou mais ações identificadas anteriormente.

Portanto, o processo de medição é definido em duas partes primárias: um modelo de processo e um modelo de informação. O modelo de processo define o relacionamento entre as atividades e tarefas necessárias ao processo de medição. O modelo de informação descreve o relacionamento entre as entidades que compõem o processo, através de um vocabulário conciso e particular.

O modelo de informação é um dos principais elementos que levam a um programa de medição bem sucedido. Esse modelo serve como mecanismo para descobrir necessidades de informação em um processo de desenvolvimento de software. Estabelece uma estrutura para relacionar conceitos diversos de medição, bem como provê uma base precisa de comunicação dentro da organização. Em contra partida, o modelo de processo fornece meios de relacionar entre si as necessidades de informação, medidas e resultados de análises. Ambos os modelos trabalham em conjunto para capacitar a organização na construção de programas de medição apropriados para cada projeto.

O modelo de processo contém quatro atividades essenciais para a implantação de processos de medição bem sucedidos: (1) Planejar a Medição; (2) Executar a Medição; (3) Avaliar a Medição; (4) Estabelecer e sustentar o processo de Medição. Por ser

iterativo por natureza, o modelo de processo foi definido para se adequar às características e restrições de um projeto em particular, sendo, ao mesmo tempo, adaptável às constantes mudanças ocorridas ao longo do projeto.

Conseqüentemente, a gerência de projetos auxiliados por processos de medição, definidos pela ISO/IEC 15939 [75], tendem a ser mais rigorosas e controláveis, uma vez que os modelos definidos estabelecem um método de medição que captura experiência e princípios aprendidos ao longo do desenvolvimento, e juntos, provêem uma base para um processo de medição eficiente.

3.2.5 O Método GQM (Goal/Question/Metric)

Em sua maioria, os padrões de qualidade e processos de software descrevem regras e metas de como devem ser realizadas atividades de medição e avaliação de software. Define-se, somente, o que deve estar presente no processo de gerenciamento para que se alcancem os resultados necessários. Em suma, têm-se diversas definições, porém pouco é dito no sentido de determinar como devem ser executadas tais atividades.

Conforme vemos em [76], o método GQM (Goal/Question/Metrics) define uma metodologia de execução de processos de medição baseados em metas, perguntas e métricas.

O GQM surgiu a partir de problemas comuns nos anos 70, onde se tentava definir como medir processos de desenvolvimento, de forma a se obter resultados adequados a metas pré-estabelecidas. Desenvolvido por V. Basili e D. Weiss, o GQM é um resultado de anos de práticas empíricas e estudos científicos.

O GQM representa uma metodologia para adequar e integrar metas a modelos de processos de software, produtos e qualidade. Seus resultados representam especificações de processos de medição que apontam conjuntos particulares de problemas, bem como interpretar seus resultados. Este define uma determinada meta, refina essa meta em perguntas e define que métricas podem responder essas perguntas de maneira objetiva e clara. Cabe ao gerente entender como utilizar as métricas e extrair o resultado necessário para execução de ações corretivas. Ao responder as perguntas, é possível identificar se a

meta em questão foi alcançada ou não. A figura abaixo melhor exemplifica como o método funciona:

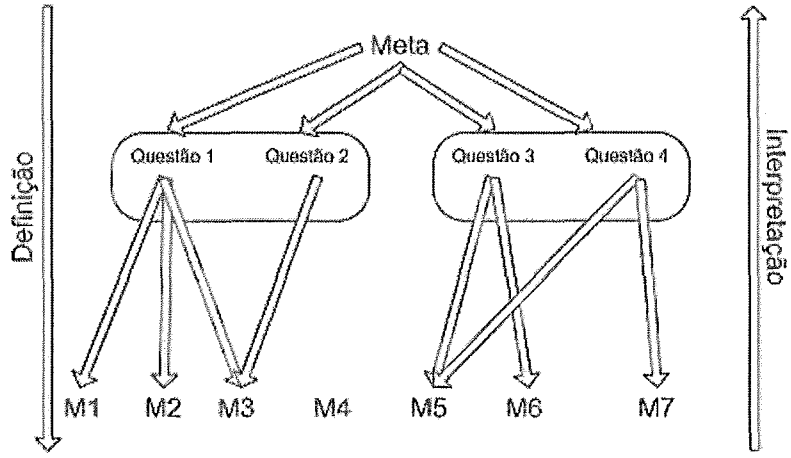

Figura 3.3. Modelo G/Q/M desenvolvido por V. Basili e D. Weiss

De acordo com a figura, o modelo GQM deve ser utilizado de cima para baixo. Ao definirem-se metas específicas, estas são refinadas em diferentes questões, as quais permitem dividir o conceito geral da meta em idéias menos complexas. Cada questão é, então, mapeada em métricas capazes de responder e, conseqüentemente, produzir informações adequadas às questões formuladas. Em contrapartida, a interpretação dos dados é realizada de baixo para cima. Responder as questões previamente definidas possibilita o entendimento da meta em foco. Conseqüentemente permite entender que o resultado obtido pela medição deve sempre estar alinhado com seu objetivo no momento, ou seja, a meta responsável pela formulação da questão. Esse alinhamento é importante, pois o compartilhamento de métricas torna-se bastante comum no processo, ou seja, dependendo do tipo de questão e/ou meta a ser trabalhada, a mesma métrica pode ser reutilizada para fornecer medições semanticamente independentes.

Dividido em quatro fases distintas, o GQM estabelece os conceitos de sua metodologia de maneira explícita, promovendo melhorias nos processos de medição chamados como "Goal-Oriented Measurement". Através de quatro fases, todo o processo de gerência e medição é desenvolvido e aplicado eficientemente.

1. Fase de planejamento: seleciona, define, caracteriza e planeja o projeto que será medido;

2. Fase de Definição: nesta fase o processo de medição é definido e documentado;
3. Fase de coleta de dados: a aquisição das informações geradas pelas medições é realizada nesta fase;
4. Fase de interpretação: após a coleta dos dados, análises são realizadas e resultados são avaliados de acordo com as metas definidas.

Conforme visto em [78], avaliações empíricas do GQM mostram que este é um dos métodos de gerência mais utilizados atualmente. De acordo com que é apresentado, o GQM demonstra uma facilidade na avaliação de modelos de desenvolvimento e fornece aos responsáveis pelos projetos bons níveis de segurança e confiabilidade nos resultados finais.

3.2.6 O Guia MPS-BR e a Estação TABA

O panorama atual das empresas desenvolvedoras de software brasileiras apresenta mudanças organizacionais em relação à garantia da qualidade de seus produtos. Visando alcançar mais espaço dentro dos mercados brasileiro e mundial, estas empresas buscam melhorar suas atividades de desenvolvimento em prol de alcançarem excelência reconhecida de seus processos produtivos. Alcançar a competitividade necessária no mercado atual, utilizando corretas técnicas e metodologias de garantia de qualidade impõem um fator determinante para o sucesso.

De acordo com dados da Secretaria de Política de Informática e Tecnologia do Ministério da Ciência e Tecnologia (MCT/SEITEC), poucas empresas no mercado brasileiro desfrutam dos benefícios decorrentes de certificações e avaliações de capacidade de processos, como o CMMI [21]. Em geral, empresas que possuem tais benefícios são empresas exportadoras de software, em sua maioria de grande porte, capazes de financiar programas de avaliação caríssimos sem ameaçar seus orçamentos internos.

Com foco na grande massa de empresas de micro, pequeno e médio portes, o MPS.BR busca democratizar modelos de avaliação sem que isso incorra em uma ruptura econômica para a empresa avaliada. Busca-se que seja adequado ao perfil desse tipo de

empresa e consiga atender as necessidades de implantar princípios de Engenharia de Software de forma adequada ao contexto em que vivem atualmente.

O MPS.BR tem como objetivo definir um modelo de melhoria e avaliação de processo de software para empresas brasileiras. Aderente a padrões e normas internacionais, o MPS.BR possui um base técnica composta pelas normas ISO/IEC 12207 – Processo de Ciclo de Vida de Software e suas emendas 1 e 2 e a ISO/IEC 15504 – Avaliação de Processo (conhecida como SPICE – Software Process Improvement and Capability determination e o seu modelo de Avaliação de Processo de Software ISO/IEC 15504-5). Sua estrutura organizacional está dividida em três componentes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MAPS) e o Modelo de Negócios (MN-MPS), onde cada modelo é descrito através de guias que formalizam seus conceitos.

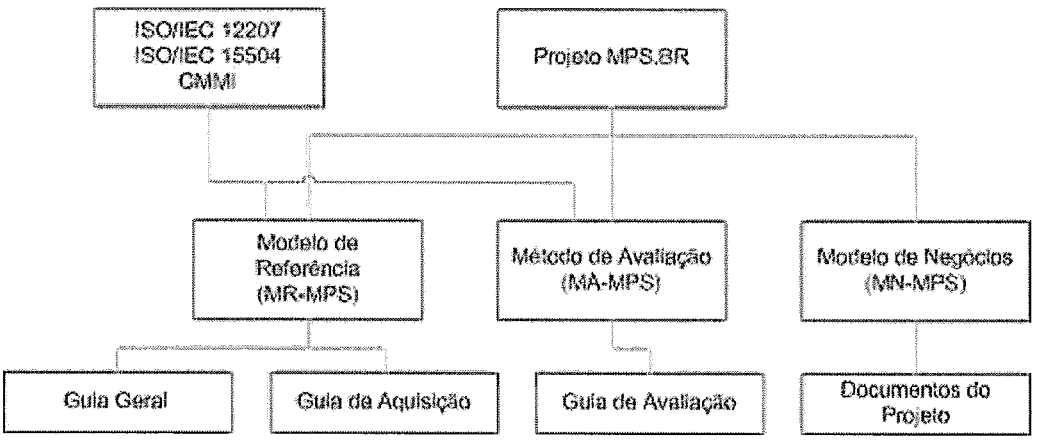

Figura 3.4. Estrutura do MPS.BR.

O MPS.BR define níveis de maturidade que combinam processos e capacidade de processos [90]. A definição de processos dentro do guia foi estruturada de acordo com a emenda 1 da ISO/IEC 12207, onde propósito e resultados de sua execução são declaradas para cada processo. O guia também entende que a correta avaliação da capacidade de um processo depende do entendimento dos atributos do processo associados ao processo propriamente dito, de acordo com o nível de maturidade em questão.

Dentro deste contexto, os níveis de maturidade estabelecem patamares de evolução para os processos de desenvolvimento das organizações. O guia define sete níveis de maturidade, cada um avaliando conjuntos de características e atributos de

processos dentro da organização. Iniciando no nível G e terminando no nível A, a escala de maturidade progride conforme os atributos de processos que compõem cada nível são atendidos. Cada nível possui seu perfil de processo e capacidade de processo, que fornecem indicadores de melhoria dentro do processo organizacional da empresa.

O **nível G, Parcialmente Gerenciado**, primeiro na escala, é composto pelos processos de Gerência de Projeto e Gerência de Requisitos, deve satisfazer os atributos de processo AP1.1 e AP2.1. Os atributos de processo AP1.1 e AP2.1 descrevem como o processo deve ser executado e gerenciado. O segundo nível, o **nível F, Gerenciado**, é composto pelo nível de maturidade anterior (G) e acrescido dos processos de Gerência de Configuração, Garantia da Qualidade, Medição e Aquisição, deve satisfazer os atributos de processo AP 1.1, AP 2.1 e AP 2.2. O atributo de processo AP2.2 descreve como os produtos de trabalho do processo devem ser gerenciados. O próximo nível na escala é o **nível E, Parcialmente Definido**, que é composto pelo nível de maturidade anterior (F) e acrescido dos processos de Treinamento, Definição do Processo Organizacional, Avaliação e Melhoria do Processo Organizacional e Adaptação do Processo para Gerência, deve satisfazer os atributos de processo AP1.1, AP2.1, AP 2.2, AP 3.1 e AP3.2. Os atributos de processo AP 3.1 e AP3.2 descrevem como um processo deve ser definido e implementado, de acordo com as características da organização. O **nível D, Largamente Definido**, é composto pelo nível de maturidade anterior (E) e acrescido dos processos de Desenvolvimento de Requisitos, Solução Técnica, Validação, Verificação, Integração do Produto, Instalação do Produto e Liberação de Produto, deve satisfazer os atributos de processo AP1.1, AP2.1, AP2.2, AP3.1, AP3.2. Em se tratando do quinto nível, o **nível C, Definido**, é composto pelo nível de maturidade (D) e acrescido dos processos de Gerência de Riscos e Análise de Decisão de Resolução, deve atender aos atributos de processo AP1.1, AP2.1, AP2.2, AP3.1, AP3.2. O sexto nível, **nível B, Gerenciado Quantitativamente**, é composto pelo nível de maturidade anterior (C) e acrescido dos processos de Desempenho do Processo Organizacional e Gerência Quantitativa de Projeto, deve satisfazer aos atributos de processo AP1.1, AP2.1, AP2.2, AP3.1, AP3.2. E por último, o nível mais alto da escala, o **nível A, Em Otimização**, é composto pelo nível de maturidade anterior (A) e acrescido dos processos de Inovação e

Implantação na Organização e Análise e Resolução de Causas, deve atender aos atributos de processo AP1.1, AP2.1, AP2.2, AP3.1, AP3.2.

3.2.6.1 A Estação TABA

A estação TABA é um meta-ambiente de trabalho, capaz de gerar ambientes de desenvolvimento específicos para organizações, adequando-se as necessidades e particularidades de seus processos de desenvolvimento [92]. Definida como um ADS (Ambiente de Desenvolvimento de Software), é altamente centrada em processos de desenvolvimento de software, o que facilita na definição, implementação e execução de ADS adequados a contextos específicos. Dentro destas características quatro funções foram definidas para a estação TABA [93]:

- Auxiliar o engenheiro de software na correta definição do ambiente de trabalho mais adequado as suas necessidades, considerando o processo de desenvolvimento que esteja inserido;
- Auxiliar o engenheiro de software na escolha correta do conjunto de ferramentas necessárias para o ambiente definido;
- Permitir a correta utilização do ambiente e das ferramentas que o compõem;
- Garantir a execução do software no ambiente definido para o desenvolvimento.

Ao longo de sua evolução a Estação TABA tornou-se capaz de não somente permitir a definição de ambientes de desenvolvimento de software, mas, em conjunto, permitir a criação de ambientes mais focados ao negócio em que o desenvolvimento se inseria. Sendo assim, visando um maior apoio às organizações e suas atividades, a Estação TABA passou a dar suporte a gerência de conhecimento através da configuração específica de seus ambientes, fornecendo ADSOrg (Ambientes de Desenvolvimento de Software Orientados à Organizações).

Contendo uma estrutura aderente a nova visão de definição de ambientes de desenvolvimento, algumas funções representam como a Estação TABA deve atuar:

- Auxiliar na configuração adequada ao processo organizacional da empresa, adequando-se ao seu desenvolvimento e manutenção de software, considerando seu processo de conhecimento.
- Auxiliar analistas e gerentes de projetos de software na correta definição dos ambientes de desenvolvimento orientados aos seus projetos, utilizando o Ambiente Configurado.
- Apoiar a gerência, o desenvolvimento e a manutenção do software utilizando o ADSOrg.

Os ambientes que suportam as funções da Estação TABA e que são utilizados na definição de ambientes organizacionais são:

- O Meta-Ambiente: caracteriza um ambiente de apoio a configuração de ambientes organizacionais específicos;
- O Ambiente Configurado: caracteriza um ambiente de apoio a definição de ambientes ADSOrg para projetos específicos;
- O Ambiente Instanciado (ADSOrg): caracteriza um ambiente utilizado pelo responsável pelo projeto específico para as necessidade do desenvolvimento. Este ambiente é derivado do ambiente configurado e deve ser definido e utilizado para cada projeto da organização.

Por ser um ambiente altamente complexo, a Estação TABA define diferentes ferramentas, cada uma com características individuais para apoio a processos ou atividades específicas, associadas aos padrões e normas de desenvolvimento de software. É aderente ao MPS.BR no que diz respeito a necessidade de definição de como seus processos devem ser conduzidos, ampliando a capacidade da organização em atingir suas metas de qualidade e competitividade.

Dentro do ambiente construído para a Estação TABA, a abordagem de medição e análise é feita fortemente baseada no *Goal-Question-Metrics* (GQM), na definição de medição e análise do CMMI e está totalmente aderente aos processos de medição do MPS.BR. Todo o processo é apoiado por duas ferramentas o MedPlan e o Metrics [94].

A ferramenta MedPlan foi projetada com o objetivo de elaborar planos de medição para projetos de software. Utilizando o método GQM, ela é capaz de oferecer ao seu usuário um conhecimento sobre objetivos, questões, métricas e procedimentos de coleta, armazenamento e análise de dados que devem ser utilizados para apoiar decisões estratégicas da organização. A figura 3.6 define o processo de avaliação:

Figura 3.5. Processo de Avaliação da Estação TABA.

A ferramenta Metrics, por sua vez, foi projetada para apoiar e disseminar informações adquiridas a partir do processo de planejamento e execução das atividades de medição e análise executadas durante o projeto.

3.2.7 Rational Unified Process e o Practical Software Measurement

O Rational Unified Process (RUP) [44] é um padrão de desenvolvimento comercial, amplamente utilizado no mercado. Apesar de ser um processo de desenvolvimento de software, o RUP define um processo de medição fortemente embasado pelo Practical Software Measuremente (PSM). O PSM é uma iniciativa de um conjunto de organizações como governos, indústrias e entidades acadêmicas, que definem melhores práticas para apoiar a gerência de projetos de software através do uso de métricas. O PSM serviu como base para definição do processo de medição contido na ISO/IEC 15939.

A proposta contida no PSM, somado ao modelo de desenvolvimento descrito pelo RUP, junta dois *frameworks* para desenvolvimento de software em uma única visão de gerência, através do uso de métricas. Mais uma vez, as necessidades de informações que forneçam suporte adequado a decisões críticas em projetos, despontam como o objetivo primário na definição do processo de medição. Em sua estrutura, o novo *framework* ajuda a gerência da seguinte forma:

1. Aumenta a capacidade de comunicação: Como a medição possui como resultado dados previamente estabelecidos como relevantes, as informações extraídas tornam-se comuns a todos os envolvidos no projeto;
2. Identificar e corrigir deficiências o mais cedo possível: As medições favorecem ações pró-ativas, e consequentemente, facilitam as atividades da gerência;
3. Escolhas e mudanças ao longo do projeto são facilitadas: Decisões em uma determinada área atingem outras áreas. As medições ajudam a identificar o grau de impacto que tais decisões trazem ao projeto;
4. Acompanhar objetivos específicos de projeto: Questões do tipo custo e prazo são avaliadas de forma mais clara e aproximada;
5. Gerenciar riscos: Medidas permitem avaliar riscos mais eficientemente.
6. Defender e justificar decisões: Medidas possuem o poder de apoiar com maior eficácia decisões referentes ao projeto.

Apesar de o modelo RUP definir um *"workflow"* para gerenciamento de processos de software, este possui como parte integrante, um sub-processo de controle e monitoramento, o qual destaca a atividade de medição em projetos de software. Em conjunto com o PSM, estes formam um *"framework"* de desenvolvimento de software pautado em práticas de gerenciamento por métricas amplamente reconhecidas como eficientes. Ambos enfatizam a importância de processos de medição como atividade de gerência de projeto.

Como descrito em [44], a capacidade de levar projetos de software ao sucesso está enraizado na aplicação de processos de medição pela gerência. Somente após implantação de programas de medição para controle de projetos, uma organização possuirá um nível satisfatório de controle do seu negócio. Com isso, o seu desempenho comercial tende a aumentar, pois os principais atores do projeto poderão avaliar, com maior precisão e confiança, as atividades constituintes do processo, e possuirão informações adequadas para apoiar decisões cruciais.

3.3 Conclusão

Com o aumento exponencial da complexidade dos softwares atuais, a utilização de meios técnicos de avaliação é de suma importância. Gerenciar utilizando métricas é uma atividade que deve ser implantada gradativamente. Processos de medição devem ser capazes de estabelecer atividades, que se realizadas ao longo do projeto, podem ajudar a determinar informações necessárias ao projeto. Os resultados devem ser avaliados e reportados para a equipe envolvida no processo, para que as ações devidas sejam executadas.

O processo de medição deve satisfazer necessidades de informações identificadas com bastante critério, sempre respeitando as características do projeto. Informações obtidas através do processo de medição devem representar o estado atual do desenvolvimento e, assim, devem ser capazes de promover ações corretivas confiáveis.

Sem o auxílio dessas técnicas, o gerente põe seus projetos em risco e impede que sua organização consiga desenhar um processo de desenvolvimento repetível, satisfatório e alinhado aos seus objetivos.

Capítulo 4

A Ferramenta

A abordagem apresentada foi baseada nas principais práticas para garantia da qualidade de processos e produtos de software. Seu objetivo está associado às necessidades de informações dos diferentes artefatos de software, produzidos ao longo do desenvolvimento, capazes de melhorar o entendimento sobre os diferentes aspectos do projeto.

Tendo como plano de fundo as atividades de medição e análise definidos pelos padrões de desenvolvimento CMMI (Nível 2) [21] e ISO/IEC 15939 [75], a abordagem utiliza métricas para produto e qualidade, conforme definido em [17], de maneira a permitir um aumento satisfatório do conhecimento do andamento do desenvolvimento. Auxiliado por ferramentas de apoio, responsáveis por gerar e disponibilizar medidas relacionadas ao software, a ferramenta apresenta-se como elemento central na criação de uma plataforma de acompanhamento de projetos, capaz de analisar e produzir resultados referentes às principais características de projetos de desenvolvimento de software com ciclo de vida incremental. Projetada sob o nome "Project Assessment and Tracking Tool" – PATT, a ferramenta possui um conjunto de funcionalidades que executam avaliações buscando conhecimento sobre o software em construção.

4.1 Introdução

O presente capítulo apresenta a abordagem desenvolvida para o processo de avaliação de projetos de software. Esta abordagem pretende definir uma técnica de avaliação capaz de identificar aspectos relevantes sobre o andamento do projeto. Em conjunto com a definição da abordagem, criamos uma plataforma de desenvolvimento, que utiliza ferramentas de apoio e uma ferramenta de avaliação, criada como protótipo deste trabalho, capaz de usar os diferentes produtos produzidos ao longo do desenvolvimento, em prol de uma análise completa.

A ferramenta foi desenvolvida tendo como ideologia alguns dos principais aspectos de desenvolvimento destacados em padrões como CMMI (nível 2) [21], ISO/IEC 9126 [13] e ISO/IEC 15939 [75]. Sendo assim, as descrições que se seguem refletem a estrutura concebida para padronização do processo de medição e análise da ferramenta:

Atividades que compõem o processo e que são apoiadas pela plataforma:

1. *Planejar Medições:* Trata dos aspectos necessários para organização do processo de medição

   Sub-Atividades:

   i – Estabelecer Objetivos;
   ii – Especificar Medidas;
   iii – Especificar Coleta de dados;
   iv – Definir mecanismos de Armazenamento;
   v – Definir técnicas de análise.

2. *Executar Medições:* Garante a correta execução e utilização dos procedimentos definidos na atividade anterior,

   Sub-Atividades:

   i – Executar Coleta de Dados;

3. *Avaliar Medições:* Responsável por extrair informações capazes de suportar ações corretivas dentro do processo.

   Sub-Atividades:

   i. Executar atividades de Análise;
   ii. Promover a comunicação das informações adquiridas dentro do processo de análise;
   iii. Executar ações corretivas;
   iv. Avaliar, selecionar e priorizar as melhores soluções.

As atividades de análise descritas formam a espinha dorsal do processo de avaliação utilizado. Baseando-se nestas diretivas, a pesquisa visou criar uma infra-estrutura que conseguisse relacionar aspectos de programação e modelagem de projetos, com ações que permitissem aumentar a qualidade, contribuindo para uma melhoria no gerenciamento e acompanhamento destes projetos.

4.2 Fundamentação Técnica da Plataforma de Avaliação

4.2.1 A Ferramenta

A ferramenta surgiu a partir da necessidade de obtenção de informações confiáveis do andamento de um projeto de software. Ela é vista como peça chave da

plataforma de desenvolvimento abordada, e se dispõe a manipular diferentes métricas orientadas a objeto com o propósito de estabelecer o progresso atual do desenvolvimento.

Ao agregarmos as atividades de coleta de dados, efetuadas por ferramentas de apoio, com técnicas de avaliação e análise encontradas na literatura especializada, conseguimos definir uma plataforma capaz de fornecer dados, relacionados com o estado corrente do projeto.

A ferramenta não foi concebida visando a geração de medidas, mas para coletar e armazenar essas informações, previamente adquiridas, agindo como um "Gestor de Medições". A coleta é realizada com auxílio das ferramentas de apoio e, então, as informações são organizadas, produzindo análises e resultados pertinentes.

4.2.2 Concepção da Plataforma de Avaliação

A plataforma formaliza o conjunto de atividades necessárias para o processo de análise. Ao longo de sua modelagem, observou-se a necessidade de estabelecer um processo de medição, composto por atividades de medição, visando a criação de um programa repetível e confiável.

Utilizando os principais aspectos dos padrões CMMI [21] e ISO/IEC 15939 [75], o processo e suas atividades foram definidos conforme exemplificado adiante. Essencialmente, a solução encontrada apresentou-se como sendo de baixa complexidade e de simples implantação.

A plataforma, como foi definida, possui os seguintes elementos:

1. Modelo de Processo (4.2.3);
2. *Workflow* de Avaliação (4.2.4);
3. Conjunto Básico de Métricas (4.2.5);
4. Conjunto de Atividades de Análise (4.2.6);

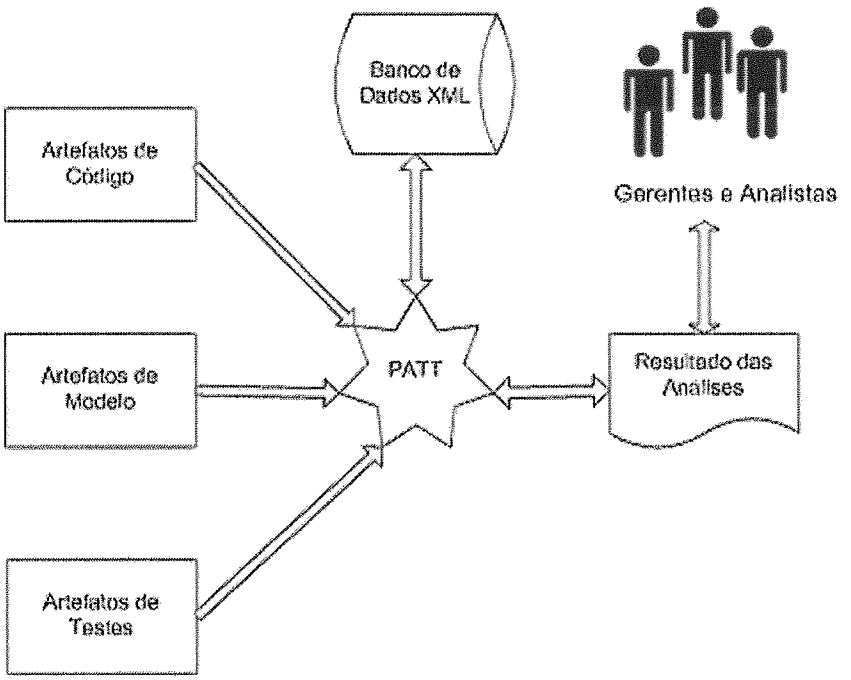

Figura 4.1. A figura descreve aspectos básicos da plataforma de avaliação da ferramenta.

4.2.3 O Modelo de Processo

Cada organização possui sua própria necessidade de informação. Ao longo do andamento de seus projetos, estas necessidades sofrem alterações constantes. Em cada fase do projeto, requisitos são modificados, recursos e tecnologia mudam, e os objetos do desenvolvimento (modelos, projetos, códigos, entre outros) sofrem alterações, independentemente da vontade dos desenvolvedores [44]. Ao lidarmos com processos de desenvolvimento evolutivos, o desenvolvimento é efetuado através da iteração de diferentes fases, cada fase é responsável por criar diferentes artefatos, e assim, em cada iteração, diferentes versões destes artefatos são criadas. Para manter um controle adequado das versões, mostrou-se necessária a criação de um protocolo interno para controle das versões, de forma a evitar que artefatos obsoletos pudessem ser usados no processo de análise.

Como o objetivo da ferramenta é avaliar projetos de software orientados a objetos, o modelo de avaliação criado manteve o foco em processos evolutivos de

desenvolvimento. Sob esse aspecto, e após avaliação de um conjunto de modelos de desenvolvimento, o modelo de processo escolhido foi o RUP (*Rational Unified Process*), de onde foram extraídas as seguintes atividades:

1. Modelagem de Negócios e Requisitos. Esta atividade é responsável por modelar processos de negócio. Produz documentos de requisitos, incluindo características e funcionalidades do software. É responsável por entender os motivos e as necessidades do software em desenvolvimento;

2. Análise e Projeto. Esta atividade é responsável por desenvolver modelos de negócio e aplicação. Tais modelos especificam características e experimentam modelos arquiteturais, criando uma visão lógica do software;

3. Codificação. Esta atividade é responsável por codificar o software. Produtos comuns nesta fase são pacotes e arquivos de código fonte;

4. Teste. Esta atividade foca na garantia da qualidade do software.

Cada fase é responsável pela criação de inúmeros artefatos essenciais para o projeto. Independente do artefato utilizado, a metodologia de avaliação entende que estes devem estar enquadrados em uma das categorias apresentadas na figura 4.2.

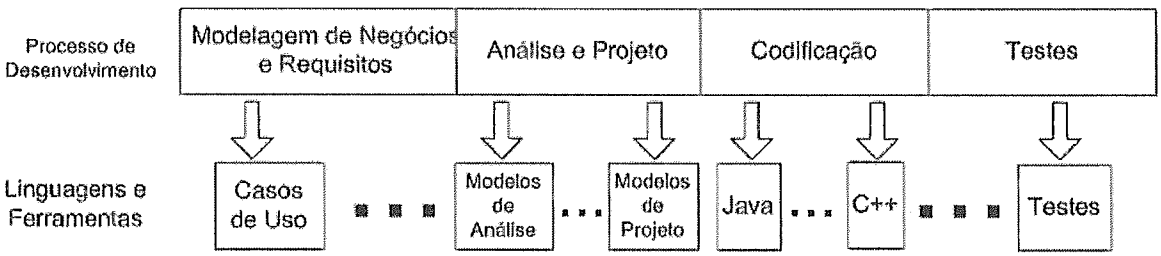

Figura 4.2. Processo de desenvolvimento de interesse da plataforma.

4.2.4 O Workflow da Ferramenta

O *"workflow"* da ferramenta foi modelado com intuito de permitir uma troca adequada de informações entre os diversos componentes que fazem parte da plataforma de avaliação. A figura 4.3 apresenta como esse *"workflow"* está organizado e demonstra a interação entre suas diferentes partes.

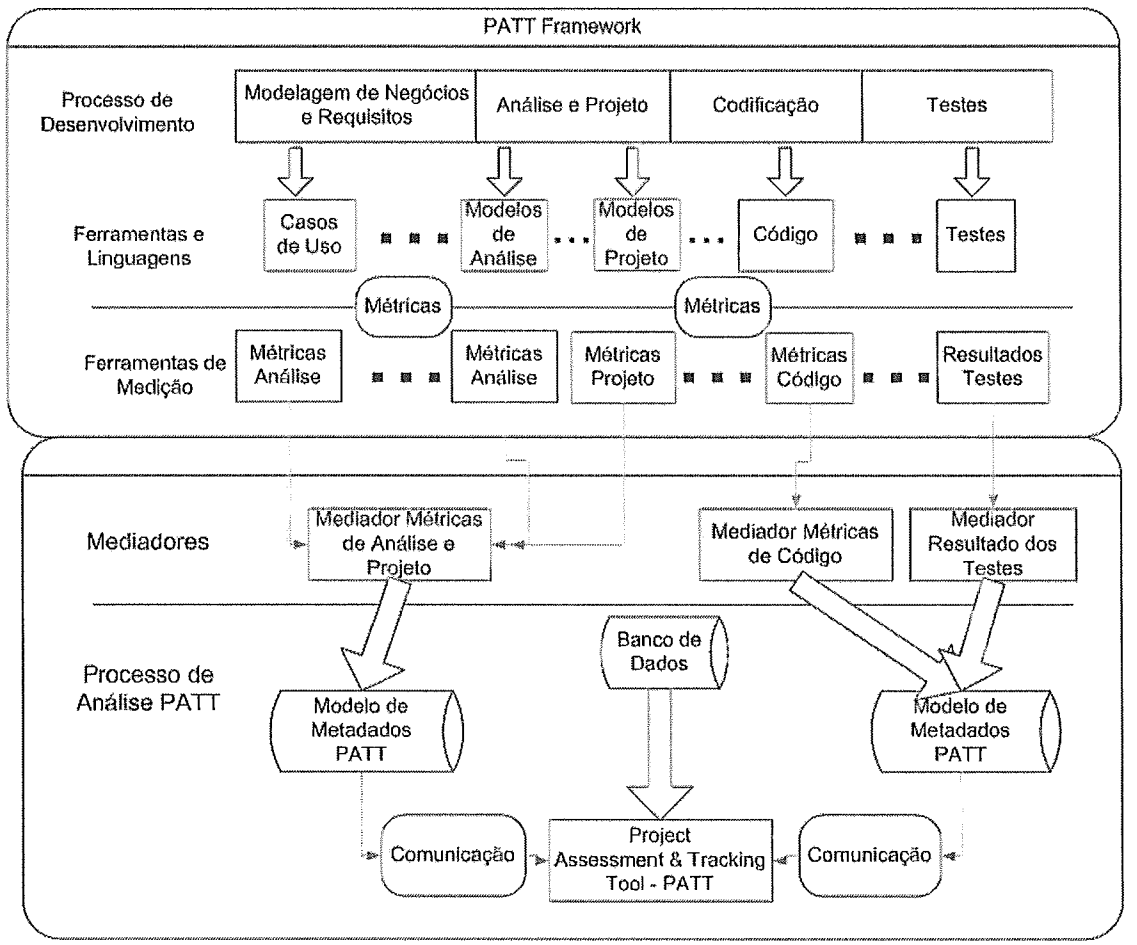

Figura 4.3. Plataforma de Avaliação: modelo de desenvolvimento de software e atividades de medição.

4.2.5 Métricas

A ferramenta analisa o andamento do projeto por meios de manipulação de métricas de produto e qualidade. As métricas usadas para a avaliação dependerão das

necessidades de informação do responsável pelo projeto. Assim, após definição e identificação dessas necessidades, o conjunto de métricas pode ser definido e configurado dentro da ferramenta e comparado conforme as atividades de confronto definidas mais adiante.

4.2.6 O Conjunto de Atividades de Análise

As atividades de avaliação que são realizadas dentro da plataforma servem para efetuar análises de projetos de software e foram sugeridas quando da concepção do "*workflow*" de execução. A seguir são apresentadas, detalhadamente, as atividades do processo.

4.2.6.1 Atividade 1: Produzir e adquirir artefatos relevantes ao projeto

Esta atividade é responsável por produzir todos os artefatos do projeto. De acordo com o cronograma de construção do software, os desenvolvedores criam diferentes artefatos utilizando diferentes ferramentas de desenvolvimento. Conseqüentemente, a criação depende do conjunto de ferramentas utilizado no projeto. Independente do tipo de tecnologia utilizada, linguagem de programação, ou paradigma de desenvolvimento, os artefatos que são produzidos representam uma visão lógica do software em construção.

Desde documentos de requisitos até pacotes de arquivos de código fonte, a proposta se mostrou favorável a utilização de diferentes tipos de artefatos, conforme apresentado pelo uso da ferramenta durante o cenário do projeto avaliado. Para tanto, foi necessário estabelecer mecanismos de representação destes artefatos que permitissem uma correta condução das atividades de avaliação. Após estudo que avaliou como os artefatos deveriam ser vistos pela ferramenta, optamos por representá-los através de documentos XML. Esta representação unificada e simples permitiu que qualquer tipo de artefato pudesse ser utilizado como objeto de avaliação sem que houvesse mudanças drásticas no processo.

O conjunto mínimo de artefatos relevantes ao processo de avaliação pode ser descrito como:

1. *Documentos de Requisitos;*
2. *Modelos de Análise e Projeto;*
3. *Pacotes de Arquivos Fonte;*
4. *Testes.*

4.2.6.2 Atividade 2: Selecionar medidas de acordo com os artefatos produzidos:

Esta atividade é responsável por identificar as medidas que podem ser extraídas de cada artefato representado em XML. Após esta identificação, um novo documento é criado e utilizado no processo de medição.

Todo o processo é realizado da seguinte forma: após a produção dos artefatos, as ferramentas são responsáveis por representá-los em documentos XML. É importante identificar quais ferramentas são capazes de exportar seus artefatos neste formato para que possam ser utilizadas durante o projeto. É fato que cada ferramenta possui meios próprios para criação destes artefatos, portanto, cada artefato criado deverá ser tratado antes de ser inserido na plataforma de avaliação. A técnica criada para o correto entendimento dos artefatos pela ferramenta foi representar as informações proprietárias em um modelo unificado chamado *Modelo de Metadados*.

Esta conversão é necessária do ponto de vista que a ferramenta manipula informações específicas e padronizadas, dentro de sua interface programada. Se esta atividade não estivesse disponível, a quantidade de informações geradas pelas ferramentas de desenvolvimento acabaria por prejudicar os mecanismos de extensão da ferramenta, e assim poucas tecnologias poderiam ser avaliadas, trazendo benefícios a um número pequeno de projetos.

4.2.6.3 Atividade 3: Transformar Documentos XML em Modelos de Metadados:

Conforme identificado anteriormente, devido a utilização de diferentes ferramentas de desenvolvimento, foi necessário representar os diversos documentos XML gerados em um modelo de metadados que, por sua vez, tem o papel de criar uma representação genérica e única das informações manipuladas pelas atividades de análise. Para que este processo pudesse ser realizado corretamente, definimos a criação de componentes chamados *Mediadores*. Mediadores são subsistemas agregados à ferramenta com o objetivo de transformar estas representações proprietárias em modelos de metadados. Independente de tecnologia, uma vez que cada documento XML possua seu próprio mediador, este pode ser avaliado pela ferramenta.

Mediadores transformam os XML com o objetivo de criar uma representação única, o que facilita o processo de crescimento da ferramenta, e padroniza a manipulação das métricas envolvidas no processo. Após converter os documentos, a ferramenta executa uma série de rotinas para carregar e analisar os dados adquiridos.

4.2.6.4 Atividade 4: Análise

A ferramenta facilita o apoio às necessidades de acompanhamento das atividades de medição existentes em um processo de desenvolvimento com ciclo de vida incremental. Em sua maioria, projetos da área possuem dificuldades em determinar o andamento do desenvolvimento no momento da avaliação dos artefatos do projeto. O protótipo apresentado no próximo capítulo demonstra como é possível avaliar programas desenvolvidos sob o paradigma de Orientação a Objetos.

Formalmente, a atividade de análise deve identificar os dados coletados, avaliá-los, planejar e efetuar análises adicionais, de acordo com a necessidade, estudar as informações adquiridas e programar revisões e ações referentes ao conhecimento obtido.

4.2.6.5 Atividade 5: Armazenamento e Comunicação

Após o processo de análise, documentos e gráficos são apresentados como veículos de divulgação e propagação dos resultados. Por sua vez, o mecanismo utilizado para armazenamento das informações envolve sistemas de banco de dados XML.

4.3 Acompanhamento

Informações sobre o acompanhamento do projeto são extraídas através das avaliações referentes às medições efetuadas de cada artefato. Os artefatos envolvidos no processo, em geral, encaixam-se nas seguintes categorias:

1. Documentos de Requisitos;
2. Modelos de Análise/Projeto;
3. Arquivos e pacotes de Código Fonte;
4. Testes.

O funcionamento da ferramenta determina que, ao menos, três artefatos devem estar presentes para que seja possível obter um conjunto de informações satisfatório. Um conjunto de requisitos, arquivos de código fonte e ao menos um tipo de modelo (análise e/ou projeto) são essenciais para a avaliação, pois contemplam, de uma forma minimalista, as informações produzidas dentro de um projeto. Essa regra impõe que a etapa mais substancial da avaliação possa ser efetuada. A utilização de documento de requisitos agrega grande valor ao resultado, sua presença ajuda na inferência de valores em termos de funcionalidades desenvolvidas e permite uma rastreabilidade ao nível de funcionalidades do software.

Em um segundo momento, definimos a estrutura organizacional dos artefatos dentro da ferramenta, ou seja, como deveríamos manipulá-los e que regras deveriam ser respeitadas para que o processo não fosse comprometido. Uma categorização dos artefatos, baseada em Versões e Tipos, foi estabelecida, visando facilitar a manipulação dos artefatos. A tabela a seguir melhor apresenta essa estrutura:

Tabela 4.1: Estrutura organizacional dos artefatos.

Artefatos	Tipos	Versões
Requisitos	Funcional	Data 1, Data 2, Data 3 etc.
Modelos	Análise e Projeto	Data 1, Data 2, Data 3 etc.
Código	Java, C++ etc.	Data 1, Data 2, Data 3 etc.

A organização dos artefatos permite, conforme a estrutura anterior, que cada um dos elementos possa ser analisado sempre a partir da sua última versão, mantendo-se um histórico das análises de versões anteriores.

4.3.1 Regras de Utilização dos Artefatos

Como o processo de avaliação pressupõe que, para se acompanhar o desenvolvimento de um requisito, deve-se confrontar todas as suas representações de artefatos geradas ao longo do desenvolvimento, ou seja, dado que um requisito R tenha gerado modelos de análise e projeto A e P, e arquivos de código fonte C, o confronto deve ser efetuado do artefato mais específico para o mais geral. Assim, para se avaliar R teríamos que confrontar C com P ou A, ou P com A. Dessa forma, fica clara a dependência de um desenvolvimento em um ciclo de vida incremental, onde cada fase do processo tem como objetivo, construir novas funcionalidades levantadas na fase anterior. Assim, como novos artefatos são criados e versões de artefatos antigos são atualizadas por iteração, deve-se reavaliar o processo para que não haja qualquer perda de informação dentro da ferramenta, tampouco, que sejam analisados artefatos inconsistentes produzindo resultados falsos.

Para impedir esse tipo de falha, estabelecemos regras para o correto andamento das análises. A primeira delas trata da categorização dos artefatos que, como visto anteriormente, devem ser mantidos em um processo de versionamento (tabela 4.1). A segunda regra determina que sempre que um artefato novo for introduzido na ferramenta, esta deve recalcular seus resultados efetuando novamente o processo de análise. Em terceiro, verificamos que para manter uma coerência dos resultados, todas as análises devem se basear nas últimas versões de cada artefato.

4.3.2 Confrontos

O resultado final da avaliação é o produto da manipulação dos artefatos, conforme definidos anteriormente. Utilizando uma combinação simples, verificamos que dez análises (confrontos) diferentes são possíveis. Estas, por sua vez, foram organizadas em quatro conjuntos:

Conjunto 1 – Confronto entre documentos de requisitos e outros artefatos:

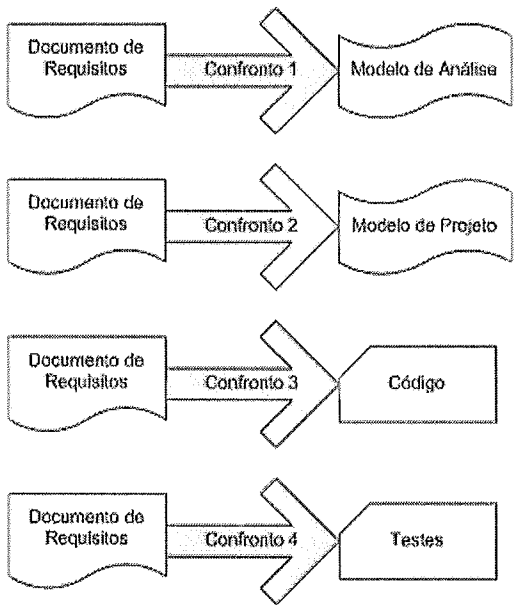

Figura 4.4-a. Confrontos com documentos de requisitos.

Conjunto 2 – Modelos de Análise e outros artefatos:

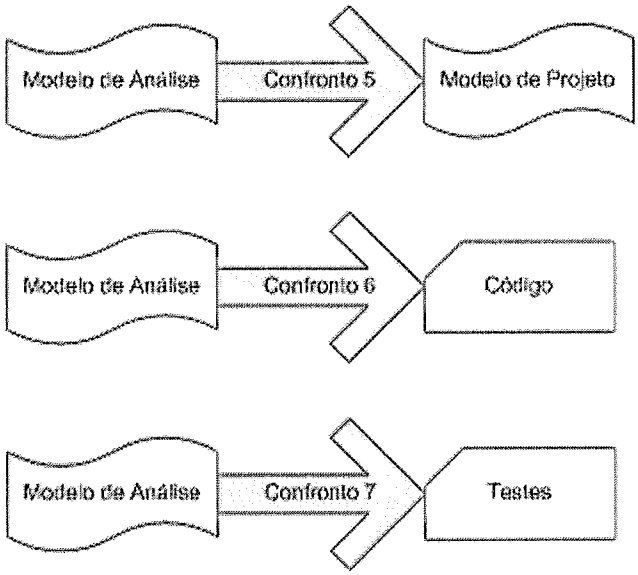

Figura 4.4-b. Confrontos com modelos de análise.

Conjunto 3 – Modelos de Projetos e outros artefatos:

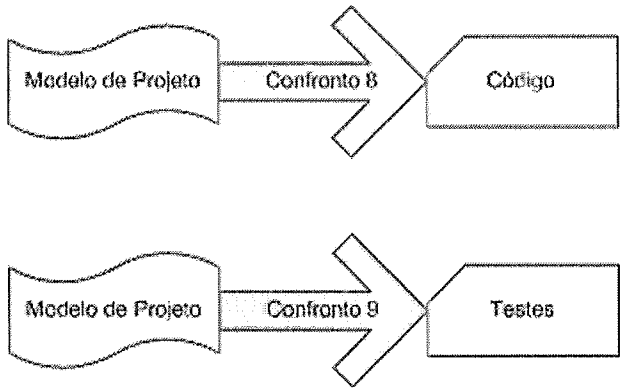

Fig. 4.4-c. Confrontos com modelos de projeto.

Conjunto 4 – Código e Testes:

Figura 4.4-d. Confrontos com código.

Do ponto de vista da ferramenta, quatro dos confrontos possíveis são essenciais para gerar informações sobre o desenvolvimento:

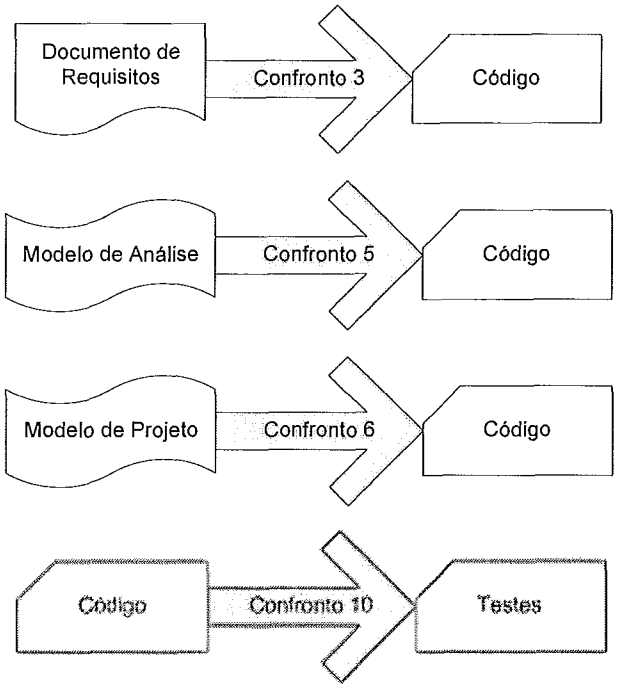

Figura 4.5. Confrontos de interesse.

I. O confronto **três** determina se uma funcionalidade Fi já foi desenvolvida em código.

II. O confronto **cinco** determina se um conjunto de classes de análise Ai={A1, A2, ..., Ax}, contendo um conjunto de métodos das classes de análise MAj={MA1, MA2, ..., MAi}, já foi desenvolvido em código;

III. O confronto **seis** determina se um conjunto de classes de projeto Pk={P1, P2, ..., Py}, contendo um conjunto de métodos das classes de projeto MPn={MP1, MP2, ..., MPj}, já foi desenvolvido em código.

IV. O confronto **dez** determina se os pacotes de classes estão funcionando corretamente de acordo com os resultados dos testes.

4.3.2.1 Documentos de Requisitos

Para identificar se um requisito já foi concluído, ou se ainda necessita de desenvolvimento, é necessário realizar ao menos um confronto entre modelo e código e

outro entre código e teste. Ou seja, inicialmente carregamos os artefatos na ferramenta, em seguida mapeamos, manualmente, os requisitos aos pacotes e classes de código fonte. Verificamos se características de modelos estão presentes nos arquivos de código, validamos se o código foi aprovado pelos testes e por fim identificamos o percentual de desenvolvimento do requisito.

4.3.2.2 Modelos de Análise

Para modelos de análise sabemos que uma funcionalidade Fi gera somente métodos de negócio essenciais ao funcionamento do sistema. Assim, um modelo de análise representa o seguinte conjunto de classes e métodos:

Ai={A1, A2, ..., Ai} de classes e MAj={MA1, MA2, ..., MAj} de métodos.

Seja, então, um código representado pelos seguintes conjuntos de classes e métodos:

Ck={C1, C2, ..., Ck} de classes e MCn={MC1, MC2, ..., MCn} de métodos.

De uma forma geral, o confronto se dá comparando os conjuntos e verificando que:

Ai está contido em Ck e MAj está contido em MCn

4.3.2.3 Modelos de Projeto

A mesma funcionalidade Fm, vista no modelo de análise anterior produz, em modelos de projeto, todos os métodos necessários para sua construção.

De acordo com o que foi visto, Fm gerou um modelo de análise com Ai={A1, A2, ..., Ai} classes e MAj={MA1, MA2, ..., MAj} métodos. Já um modelo de projeto teria Pk={P1, P2, ..., Pk} classes e MPn={MP1, MP2, ..., MPn} métodos, onde Ai contido em Pk e MAj contido em MPn. Para o mesmo código da análise anterior, Cx={C1, C2, ..., Cx} classes e MCy={MC1, MC2, ..., MCy} métodos seriam construídos, representando, portanto, boa parte da construção de F1.

Mesmo que um processo de desenvolvimento não seja formalmente definido, o nível de informação produzido pela abordagem é satisfatório quando pretendemos ter algum tipo de indicação do andamento do projeto.

Conseqüentemente o confronto entre o modelo de projeto e o código construído seria realizado obedecendo a seguinte regra:

Pk está contido ou é igual a Cx e MPn está contido ou é igual a MCy.

Nesta hipótese, o percentual de desenvolvimento estará associada ao percentual de confirmações entre número de classes e número de métodos. É fato que isso ainda não determina o real andamento do projeto, mas produz fortes indícios de como o desenvolvimento está sendo realizado. Assim, de acordo com o percentual dos métodos do modelo de projeto fosse encontrado em código, poderíamos sugerir que o desenvolvimento do projeto estaria próximo de concluir. Faltando apenas verificar os resultados dos testes criados para garantir a qualidade do desenvolvimento dos arquivos de código fonte.

4.3.2.4 Testes

Para garantir que os resultados não possuem qualquer tendência ou gerem resultados falsos, não basta indicar que, a partir de confrontos realizados com artefatos de código e modelos, uma funcionalidade esteja concluída sem antes se certificar que esta esteja funcionando corretamente. Após os respectivos confrontos e de posse dessas informações, só se pode sugerir, com certo grau de confiabilidade, que uma funcionalidade está finalizada após a execução do confronto entre os códigos e os resultados dos testes efetuados para cada conjunto de pacotes que representam um requisito. Ao se avaliar este confronto e identificar que todas as classes passam nos testes específicos, pode-se sugerir, então, que o desenvolvimento está próximo do fim.

4.3.2.5 Condições para Garantia de Confiabilidade

As regras apresentadas anteriormente para as análises (confrontos) dos artefatos, identificam quatro condições obrigatórias e uma suplementar para que os resultados sejam gerados corretamente.

I. Condição 1:

Para que seja possível efetuar a análise, deve-se satisfazer a condição de possuir um conjunto de requisitos, todos os pacotes de código do projeto inseridos na ferramenta, bem como ao menos um dos modelos de classes (Análise ou Projeto). A condição um é satisfeita se pelo menos estes três artefatos estão disponíveis: Requisito, Código e um Modelo.

II. Condição 2:

A execução da análise é realizada a partir das medidas contabilizadas pelas ferramentas de apoio. Portanto, para que a condição dois seja satisfeita, deve-se efetuar a análise de acordo com as regras de conjuntos, definidas anteriormente. A condição dois é satisfeita se os confrontos são efetuados conforme o tópico de Análises.

III. Condição 3:

A condição três parte do princípio que somente as medidas não são suficientes para determinar se uma determinada classe foi criada ou não. Assim, uma busca pelos nomes dos métodos tornou-se necessário. Portanto, a condição três é satisfeita se os nomes dos métodos escritos nos modelos são idênticos aos nomes dos métodos no código.

IV. Condição 4:

A condição quatro determina que os códigos avaliados devem estar funcionando corretamente. Assim esta condição é satisfeita se os resultados dos testes são satisfatórios.

V. Condição 5 (Suplementar):

Esta condição avalia os atributos das classes descritas em modelos e arquivos de código fonte. Uma comparação simples do número e nome de atributos é realizada para garantir um certo grau de conformidade de criação entre modelos e códigos.

4.3.3 Estimativas

Ao longo do processo de análise é possível importar valores estimados para as métricas envolvidas no processo de avaliação. De acordo com informações históricas, é possível identificar esses valores e utilizá-los na ferramenta. A ferramenta também permite que a exportação dessas métricas com intuito de fomentar esse armazenamento histórico e promover a criação de bases de métricas para criação e cálculo de fórmulas matemáticas capazes de determinar esses valores.

Para cada métrica utilizada, é possível inserir um valor estimado que, quando comparado com o valor real, consegue projetar indícios da quantidade de esforço necessário para o desenvolvimento do produto em questão.

O estudo da diferença de comportamento entre valores reais e estimados foi projetado para auxiliar no entendimento do processo de desenvolvimento e na busca ações de controle e planejamento que permitam identificar melhores práticas para o desenvolvimento.

4.3.4 Resultados

Os resultados, como apresentados pela ferramenta, possuem características informativas, e desempenham um papel de apoio às decisões efetuadas ao longo do

projeto. Tentamos oferecer material à gerência que pudesse ser utilizado em auxílio às ações executadas durante o desenvolvimento.

Na seção que apresenta a Arquitetura, definimos como estes mecanismos são executados e apresentados aos usuários.

4.3.5 Armazenamento

O CMMI (nível 2) [21] explica que ao armazenar informações obtidas em processos de análises, conseguimos utilizar os dados históricos a um baixo custo e em um tempo hábil, permitindo interpretá-los em benefício do projeto. Ao longo de todo o processo, observamos a produção de informações relevantes ao projeto que devem ser armazenadas para que se possa manter uma base histórica e obter uma visualização evolutiva do projeto.

Através da criação do modelo de metadados, concluímos que essas informações teriam um formato específico e próprio da ferramenta e, assim, pudessem ser armazenadas de maneira simples e eficiente.

A escolha mais lógica foi utilizar um banco de dados como mecanismo de armazenamento, permitindo uma melhor disposição das informações criadas pela ferramenta.

4.3.6 Comunicação

De uma maneira geral, a disseminação da informação é feita através da ferramenta. As informações estão disponíveis a partir do banco de dados, ou da própria ferramenta, o que facilita a geração de relatórios gerenciais. Uma segunda opção de distribuição dos resultados é apresentada pela ferramenta de análise. Dentro desta, diversos gráficos são produzidos e permitem uma avaliação "on-the-fly", ou seja, dinâmica dos resultados obtidos.

4.3.7 Modelo de Metadados

A criação do modelo de metadados mostrou-se necessária, a partir do momento que identificamos dificuldades em manter um determinado nível de isenção tecnológica. De início, pensamos em uma solução que estivesse desacoplada de qualquer linguagem ou paradigma de desenvolvimento, permitindo que a execução das análises, fosse realizada em cima de uma plataforma genérica, a qual suportasse diferentes metodologias de desenvolvimento.

Para tanto, construímos a plataforma de desenvolvimento buscando maneiras de manipular as diferentes representações XML dos artefatos do projeto, sem que afetassem eventuais necessidades de extensão. Sendo assim, ao identificarmos como as ferramentas de apoio geravam seus resultados parciais em XML, resolvemos padronizar o nosso modelo de armazenamento.

Ao generalizar os conceitos estudados, verificamos que as facilidades de manipulação de diferentes objetos foram razoavelmente boas, facilitando o processo de expansão da ferramenta. Benefícios para esta mudança de representação envolvem procedimentos de armazenamento mais simples, funções de análises menos complexas e facilidade na manipulação dos resultados. A gerência das medidas no formato de metadados acelerou o acesso aos dados na base XML, e conseqüentemente, melhoraram a organização das informações, derivadas dos resultados.

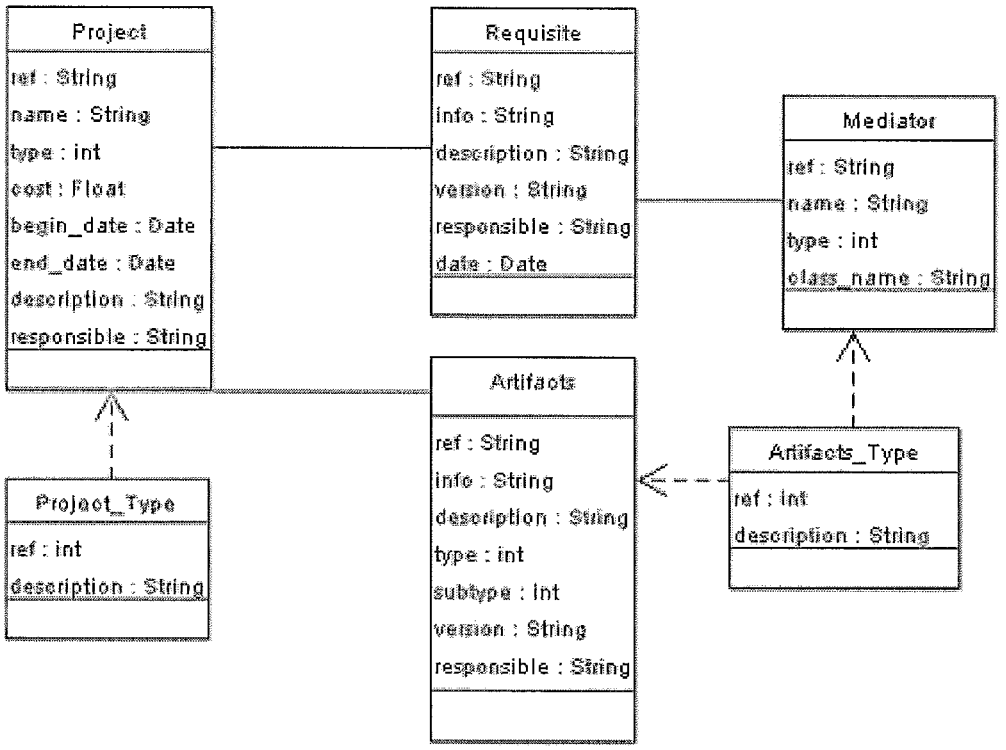

Figura 4.6. Modelo de metadados criado para a ferramenta.

4.3.8 Mediadores

Mediadores são os componentes da ferramenta que auxiliam na transformação dos resultados das medições de um formato específico e proprietário, criado pelas ferramentas de apoio, para o formato mais geral, representado através do modelo de metadados definido anteriormente.

Esses componentes possuem uma estrutura simples, que facilmente se integram à ferramenta. Seguindo diretivas de programação impostas pela interface de programação da aplicação (API), tornou-se fácil o mecanismo de incorporação de novos mediadores para diferentes artefatos. A idéia almejou criar componentes que conseguissem transformar os resultados das medições de um artefato e carregá-los na ferramenta, para que pudessem ser utilizados corretamente.

O processo de transformação dos documentos XML proprietários em documentos do modelo de metadados é realizado de maneira simples. As figuras 4.7 e 4.8 exemplificam como estes procedimentos ocorrem:

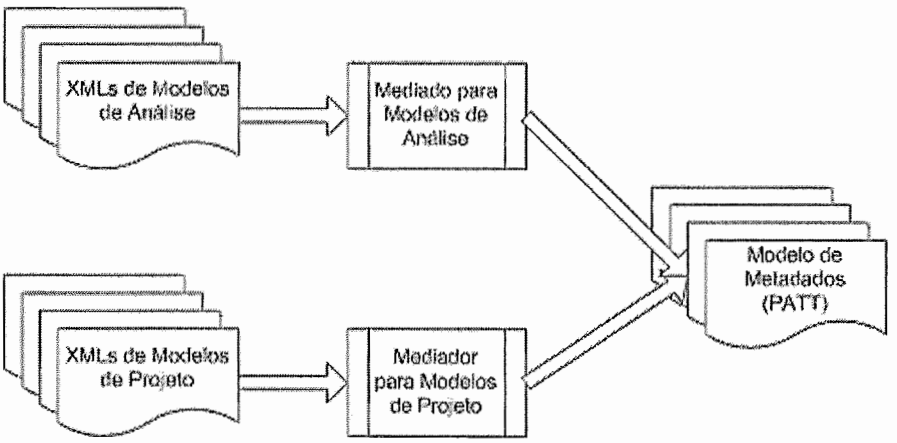

Figura 4.7. Modelos de análise e projeto possuem seus próprios mediadores para que não hajam conflitos entre documentos XML.

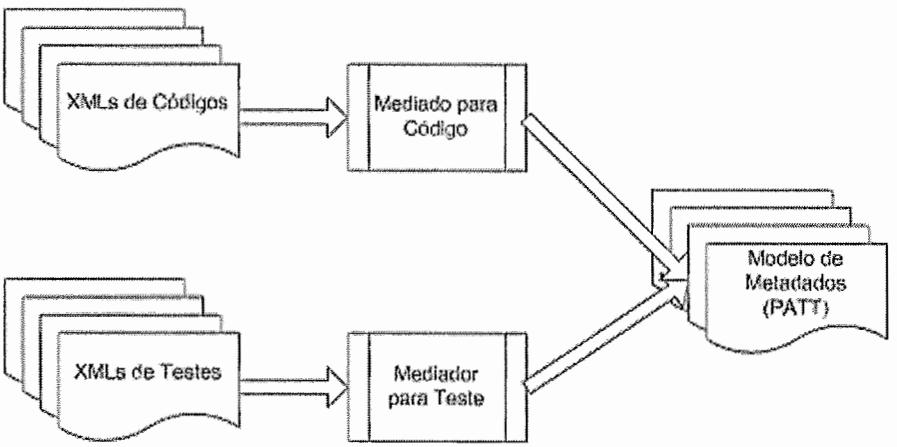

Figura 4.8. Arquivos de Código Fonte e Testes também possuem seus próprios mediadores.

4.4 Conclusão

Este capítulo apresentou o estudo realizado para criação da plataforma de avaliação. Em apoio a essa plataforma, definimos como devemos utilizar a ferramenta de avaliação na aquisição e manipulação das métricas de produto e qualidade para execução de análises aplicadas aos artefatos de software, bem como obtenção de resultados e propagação das informações adquiridas. Entendemos que é possível avaliar projetos de software extraindo medidas de todos os artefatos constituintes do desenvolvimento, e confrontando estas medidas em busca de informação confiável relativa a tempo de desenvolvimento e qualidade do produto.

Com o objetivo de identificar informações relativas ao andamento do projeto, criamos um conjunto de atividades capazes de produzir informações que pudessem auxiliar ações gerenciais. O estudo apresentado discorreu sobre como é possível comparar diferentes artefatos e usá-los como componentes de avaliação do projeto, apresentou, também, como devemos analisá-los, armazená-los e apresentá-los aos usuários, como um painel de controle.

Capítulo 5

O Protótipo da Ferramenta

5.1. Introdução

A abordagem apresentada no capítulo anterior para o processo de análise forneceu amparo técnico para a criação do protótipo da ferramenta. Sua criação serviu para materializar o processo de avaliação, uma vez que pôs em prática todas as atividades definidas para o processo. A seguir apresentamos os principais aspectos na concepção da plataforma, envolvendo a apresentação da arquitetura, e o desenvolvimento das atividades do processo.

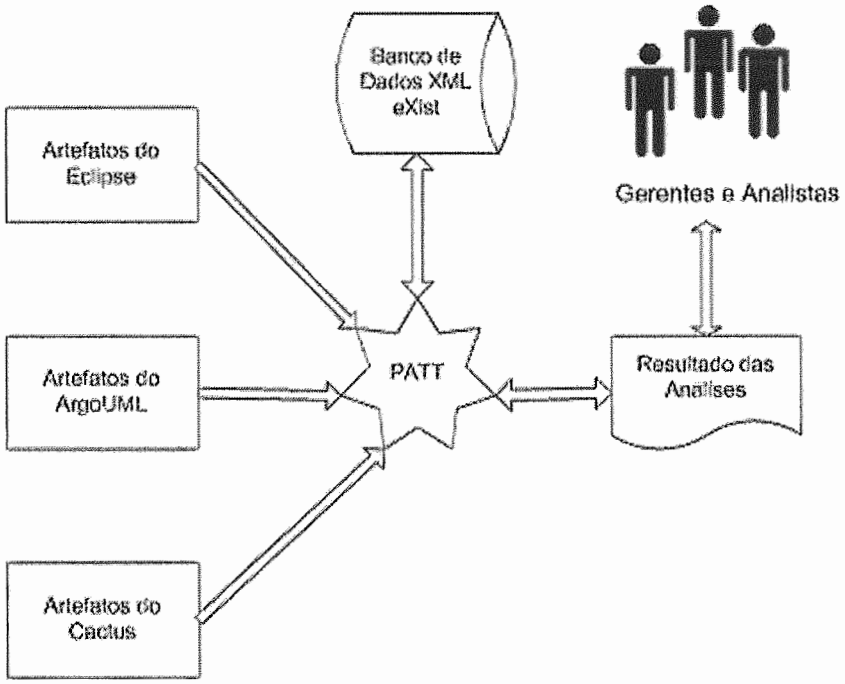

Figura 5.1 – Modelo da plataforma construída para o cenário de uso do protótipo.

5.1.1. Modelo de Processo

O modelo de processo apresentado no capítulo anterior determinou como um processo de desenvolvimento de software orientado a objetos pode ser mapeado e padronizado para facilitar a criação de modelos de negócio. De acordo com o que foi

apresentado, foram definidas as seguintes fases para apoiar as atividades de avaliação propostas para o protótipo:

1. Modelagem de Negócios e Requisitos. Esta fase é responsável por modelar processos de negócio. Produz documentos de requisitos, incluindo características e funcionalidades do software. No que diz respeito ao protótipo desenvolvido, o escopo definido para documentos de requisitos embasam sua construção a partir do modelo de criação de documentos de requisitos no padrão IEEE [32]. Estes documentos são tratados em formato PDF e transformados em documentos XML para melhor manipulação.

2. Análise e Projeto. O protótipo utilizou o programa de desenvolvimento ArgoUML [24] (Apêndice C) para criação destes artefatos. Esta ferramenta apresentou-se razoavelmente eficaz na tradução de suas informações em documentos XML. Contudo demonstrou fragilidade quando trabalhamos com um número elevado de classes.

3. Codificação. A etapa do desenvolvimento que melhor justifica a necessidade de modelos de análise de processos de desenvolvimento de software é a codificação. Representa a materialização dos conceitos abstratos definidos nos modelos de análise e projeto para o desenvolvimento. Nesta etapa, utilizamos a ferramenta Eclipse [23] (Apêndice C) como IDE de desenvolvimento. Sua escolha também foi influenciada pela sua capacidade em representar seus artefatos em documentos XML.

4. Teste. A ferramenta escolhida para o desenvolvimento deste tipo de artefato foi o *"framework"* de testes Cactus [87] (Apêndice C). Como seu conceito de implementação é todo baseado na geração de documentos XML, foi possível desenvolver, dentro da plataforma construída para o protótipo, um conjunto de classes de testes que gerassem seus resultados em documentos XML.

Estas atividades forneceram um amparo técnico para a organização do processo de avaliação. Notoriamente, atividades como estas são comuns em quase todo processo de desenvolvimento e, como tais, ajudam a aumentar o alcance da ferramenta em atividades de avaliação, independente do projeto de software desenvolvido.

5.1.2. *Workflow* de Avaliação

O "workflow" de avaliação definiu como o processo deve ser conduzido, de forma a produzir resultados relevantes ao desenvolvimento avaliado. Os artefatos manipulados dentro da plataforma advêm das ferramentas envolvidas no projeto. Em geral, as ferramentas, além de produzir os artefatos, também fornecem as informações relativas a cada um e, conseqüentemente as métricas manipuladas dependem da capacidade de medição de cada ferramenta de apoio. A execução das atividades de medição se torna possível após a aquisição destas informações, pois são armazenadas dentro da ferramenta de análise e avaliadas conforme especificado.

Toda a análise ocorre dentro da ferramenta, e seus resultados são armazenados dentro do banco de dados XML eXist [25] (Apêndice A). Essas informações ficam a disposição para consultas posteriores e permitem formar uma base histórica de consulta, importante para o desenvolvimento de projetos futuros.

5.1.3. Conjunto Básico de Métricas

O conjunto de métricas utilizado reflete, em geral, métricas para software orientado a objetos. As ferramentas escolhidas facilitaram a aquisição das métricas porque conseguiram aliar a facilidade do desenvolvimento com a funcionalidade de exportação das métricas em arquivos XML. Tanto o aplicativo de modelagem ArgoUML (Apêndice C) como as ferramentas de desenvolvimento Eclipse [23] (Apêndice C) e o plugin Metrics [88] (Apêndice C), agiram como facilitadores desse processo de aquisição. Em relação aos testes efetuados para o software, o *"framework"* Cactus [87] (Apêndice C) cumpriu papel relevante apresentando resultados satisfatórios na contabilização dos dados de testes.

A lista de métricas apresentada adiante compõe o conjunto de interesse da ferramenta em suas avaliações. Este conjunto não é exaustivo e pode sofrer alterações de acordo com o projeto em questão.

- O Conjunto CK de Métricas [17]:

1. Weighted methods per class (WMC);
2. Depth of Inheritance Tree (DIT);
3. Number of Children (NOC);
4. Response for a Class (RFC);
5. Coupling Between Objects (CBO);
6. Lack of Cohesion in Methods (LCOM);

- Métricas Propostas por Lorenz and Kidd [16]:
  1. Class Size (CS);
  2. Number of Operations Overridden by a Subclass (NOO);
  3. Number of Operations added by a Subclass;

- Métricas de Projeto [6]:
  1. Source Lines of Code (SLOC);
  2. Function Points (FP).

5.1.4. Exportação

O protótipo também dispõe de um mecanismo de exportação das informações adquiridas (métricas) e transformadas (análises). Este mecanismo foi pensado de forma a ajudar na criação de bases históricas de medidas, para auxiliar na eventual criação de mecanismos de identificação de valores estimativos, como fórmulas estatísticas, capazes de determinar o comportamento padrão de criação e evolução do software e projetar, no tempo, modelos do desenvolvimento dos sistemas avaliados.

Como todo o processo de manipulação de dados, as informações são exportadas em arquivos XML, conforme o próprio modelo de metadados, e armazenadas na base de dados XML.

5.2. Arquitetura da Ferramenta

De acordo com a definição da plataforma de avaliação, definimos uma arquitetura para construção da ferramenta apoiada conforme as atividades apresentadas na figura 5.2. A arquitetura planejada buscou manter a ferramenta de avaliação como ponto central da plataforma de avaliação utilizando as ferramentas de apoio como fontes de informação dos artefatos produzidos no projeto.

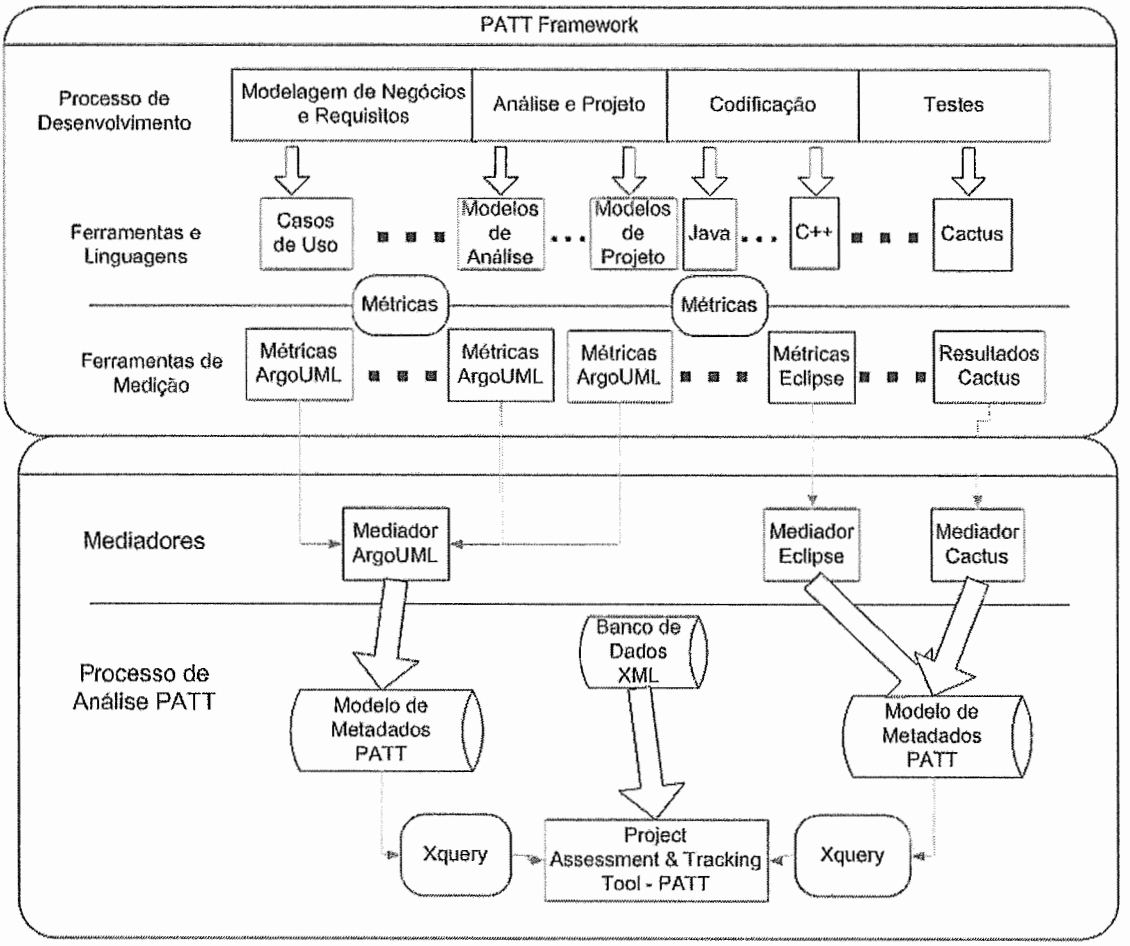

Figura 5.2 – Framework criado para avaliação do protótipo.

5.3. Modelo de Metadados

De acordo com o que foi apresentado no capítulo anterior, o modelo de metadados facilita a manipulação das informações referentes aos artefatos do projeto, com uma representação simples e unificada. Utilizado pelos mediadores (definidos a seguir), conseguem abstrair diferentes tipos de representação, próprias de ferramentas de desenvolvimento.

5.4. Mediadores

Através da interface chamada `"Mediator"`, diferentes aplicações podem ser integradas à ferramenta, e executadas independentemente do tipo de aplicação em avaliação. A interface `"Mediator"` define um único método chamado `"execute()"`, o qual é usado pela ferramenta, quando um mediador deve executar sua tarefa. Esta regra permite que diversos artefatos possam ser avaliados independentemente, pois envolvem as regras necessárias para a conversão dos dados nos XML proprietários em XML no modelo de metadados. A interface é necessária para que exista uma uniformidade na construção dos mediadores, tornando-os de fácil instalação e execução pela ferramenta.

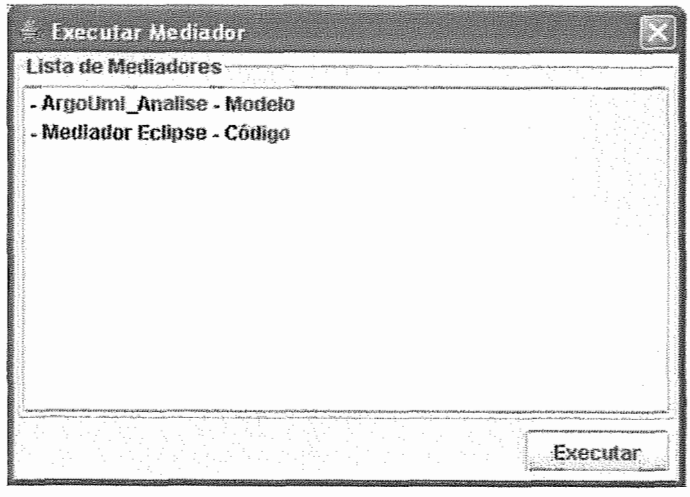

Figura 5.3. A Interfade de Mediadores: Executando um mediador da lista de mediadores disponíveis

O processo de transformação é realizado de maneira simples. De acordo com a figura 5.4, o protótipo consegue analisar o projeto, após a execução dos mediadores, criados para transformação dos resultados das medições, feitas pelas ferramentas ArgoUML, para os modelos de análise, Eclipse em códigos fonte Java e Cactus para testes criados para as classes Java.

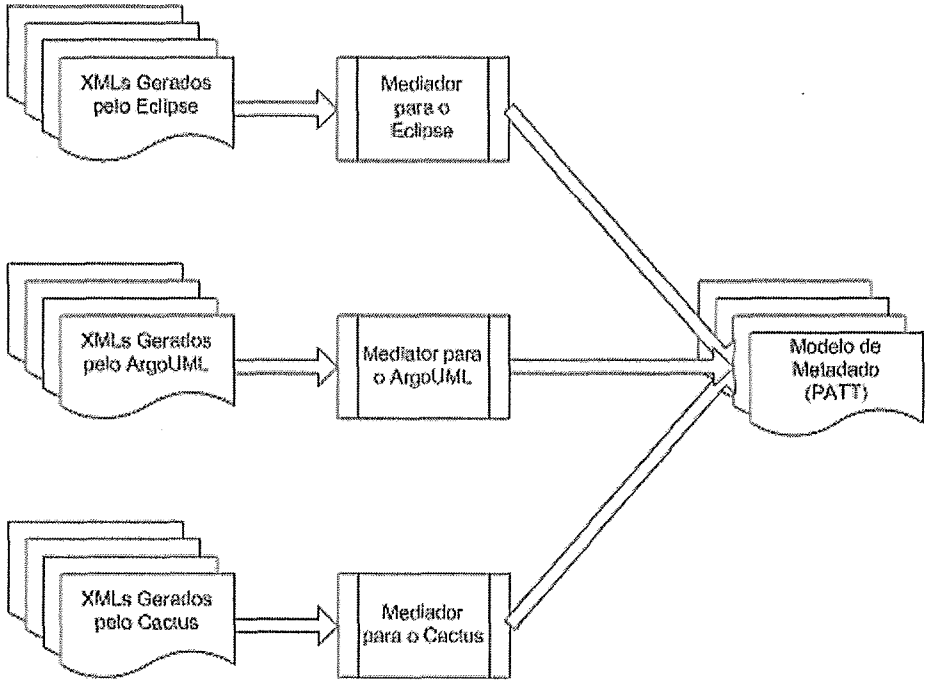

Figura 5.4-a. Mediadores criados especificamente para a geração dos arquivos em formato de metadados necessários para a manipulação das informações pela ferramenta.

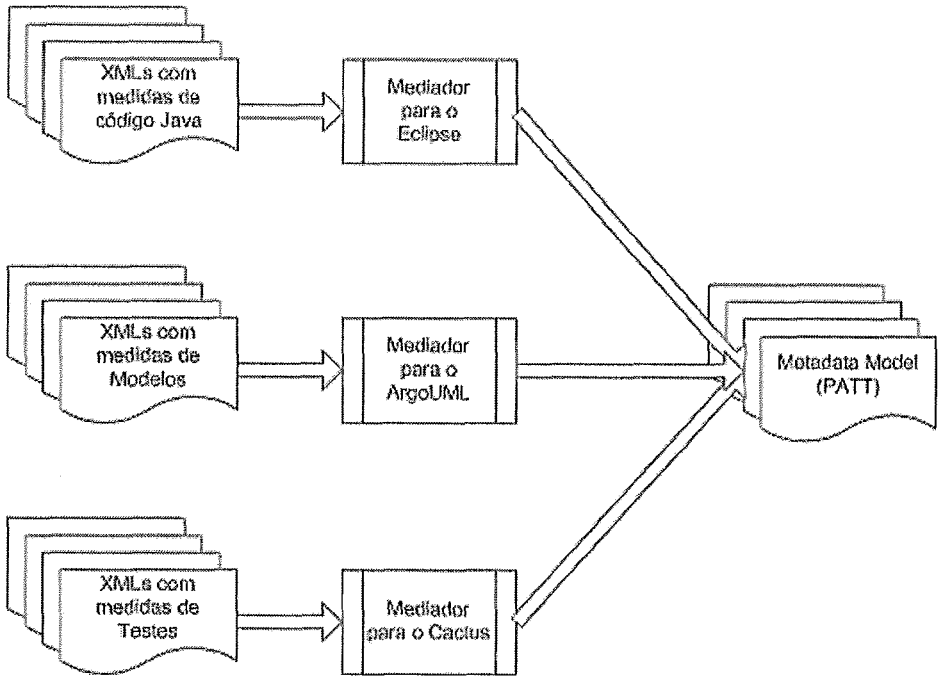

Figura 5.4-b. Exemplo para os artefatos utilizados.

5.4.1. Configurações

A utilização de mediadores pode ser refinada através de mecanismos de configuração. Tais mecanismos determinam como informações adquiridas pelas ferramentas de desenvolvimento devem ser carregadas no protótipo. Através de um arquivo XML de configuração, é possível informar ao mediador que informações serão importadas e, conseqüentemente, permitindo um nível maior de objetividade nas avaliações.

5.5. Armazenamento

O armazenamento dos documentos é efetuado utilizando um banco de dados central, disponível para a ferramenta, representado por um banco de dados XML Nativo. De acordo com Salminen e Megginson [49,51], Bancos de Dados XML Nativos são sistemas de gerenciamento adequados para processar documentos XML. Um XBMS

(XML Based Management System) para processamento de documentos XML em larga escala deve possuir alguns requisitos, incluindo:

1. Armazenar documentos eficientemente;
2. Suportar linguagens de query;
3. Suportar APIs como SAX [49] e DOM [50];
4. Controlar Concorrência.

O armazenamento dos documentos, em bancos de dados XML, estabelece e mantém uma estrutura mais simples, sem que haja perda na confiabilidade e consistência dos dados armazenados. A estrutura de armazenamento apresentou ganhos consideráveis por ser livre de plataforma. Isto contribuiu para o perfil de abstração estabelecido para a plataforma, e com isso, a integração das atividades de análise e distribuição dos resultados obteve ganhos em termos de manipulação e manutenção de históricos de análises passadas.

A idéia de armazenar os arquivos XML gerados pelos mediadores, bem como os diversos outros XML criados pela ferramenta, partiu da necessidade de mantermos um histórico de todas as atividades realizadas dentro da plataforma. Esse histórico é utilizado como uma fonte de informações para projetos futuros.

A partir de relatórios, previamente especificados, o gerente de um novo projeto poderá obter informações determinantes na escolha de ações críticas dentro do novo desenvolvimento.

O banco utilizado foi todo desenvolvido em Java e chama-se eXist™ (Apêndice A) . O eXist™ possui todas as características de uma banco XML Nativo, além de se encaixar na diretiva principal da pesquisa: é um software "open source", ou seja, está sob a licença GPL (General Public License). Para maiores detalhes consulte o apêndice A.

Para manipulação dos dados armazenados na base XML, utilizamos linguagem de consulta XQuery (Apêndice D). XQuery [42] é a linguagem padrão para armazenamento de informações da ferramenta, pois possui a característica de simplificar o modo como o manuseio de dados ocorre.

5.6. Ferramentas de Apoio

São denominadas ferramentas de apoio, todas aquelas que estão envolvidas na condução do processo e construção do software. Em sua maioria, ferramentas de apoio refletem ferramentas de desenvolvimento utilizadas durante o projeto. Em nossa pesquisa, buscamos definir uma plataforma de desenvolvimento e análise que contivesse somente ferramentas de código livre.

Ao trabalhar com documentos de requisitos, modelos de análise e projeto e arquivos de código fonte, três ferramentas foram escolhidas:

1. Editor de Texto: para a definição dos requisitos do software;
2. Ferramenta de modelagem: para a criação dos modelos de análise e projeto;
3. Ferramenta de desenvolvimento e construção do software: para a construção do software.

Mais especificamente, a preocupação estava em definir ferramentas que pudessem representar seus artefatos em formato XML. Como os documentos de requisitos podem ser tratados em formato PDF, não foi necessário definir um editor de texto capaz de representar suas informações em documentos XML, contudo duas ferramentas ainda se encaixavam no perfil:

I - Modelos de Análise e Projeto: a ferramenta escolhida foi o ArgoUML™ desenvolvida pela Universidade da Califórnia. Desenvolvida em Java, esta possui a característica de representação XML dos artefatos.

II – Código: como ferramenta de desenvolvimento, escolhemos a IDE de programação Eclipse™ da IBM. Em conjunto com o plugin Metrics, também possui característica de representação dos artefatos desenvolvidos em XML.

III – Teste: como mecanismo de apoio à avaliação dos testes, foi utilizado o *"framework"* Cactus. Este *"framework"* foi integrado na solução da ferramenta pois gera seus resultados em arquivos XML, necessários para utilização pela ferramenta.

Utilizando as três ferramentas de apoio, foi possível obter medições para modelos de análise e projeto e para códigos escritos em linguagem orientada a objetos, contendo as principais informações definidas por [16] e [17].

5.6.1. Coleta de Dados

A coleta dos dados é realizada em conjunto com as ferramentas de apoio. As medições são efetuadas e então repassadas para a ferramenta com o auxílio dos componentes Mediadores. A figura a seguir exemplifica como a plataforma foi construída.

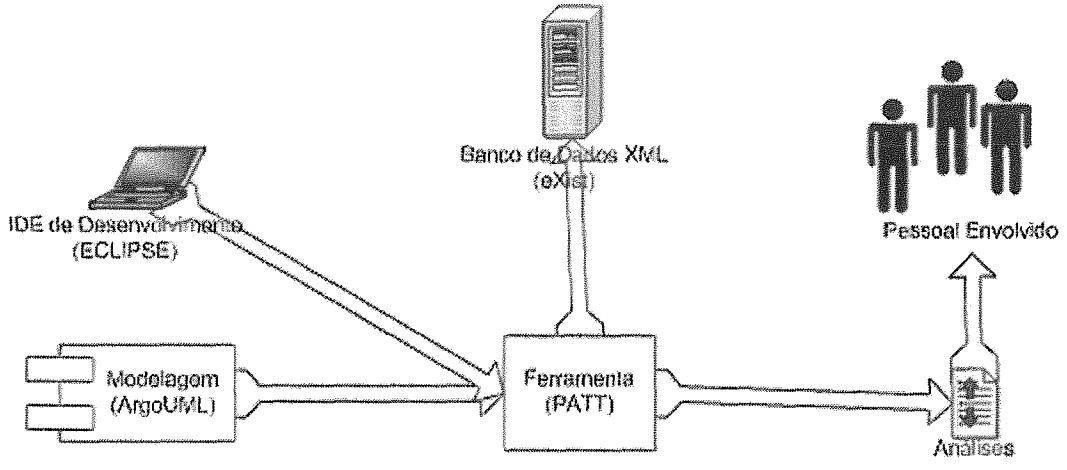

Figura 5.5. Plataforma definida para o protótipo da ferramenta.

5.7. O Protótipo em Execução

O apoio e amparo técnico extraídos das atividades de medição e análise dos padrões de desenvolvimento, CMMI [21] e ISO/IEC 15939 [75], serviu para ajudar a identificar algumas atividades relevantes ao nosso processo de avaliação:

1. **Estabelecer objetivos de medição e análise, verificando seu alinhamento com as necessidades de informações do projeto**; Esses objetivos são estabelecidos a partir dos princípios de interesse da ferramenta, ou seja, manipular métricas com o objetivo de avaliar o andamento do projeto.

2. **Especificar as medidas, Coleta de Dados e Mecanismos de Armazenamento, Técnicas de Análise e Mecanismos de Distribuição dos Resultados.** Essas ações dependem do conjunto de ferramentas de desenvolvimento escolhido. Os IDE's de desenvolvimento ArgoUML, Eclipse, o "plugin Metrics" e o *"framework"* de testes Cactus se apresentaram bastante eficientes na manipulação das métricas.

3. **Instituir Coleta, Armazenamento, Análise e Distribuição da Informação;** A coleta é feita quando as métricas são exportadas em documentos XML pelas ferramentas, o armazenamento ocorre quando as atividades de confronto são executadas e as informações são apresentadas conforme as interfaces gráficas da ferramenta.

4. **Fornecer resultados objetivos que possam ser utilizados no apoio a decisões e ações corretivas apropriadas**. Os resultados são apresentados nos diferentes gráficos de análise disponíveis na ferramenta.

Executar uma análise dentro da plataforma proposta e utilizando o protótipo da ferramenta é uma tarefa relativamente simples: Suponha que um número de artefatos (modelos de análise, projeto, e código fonte) representa diferentes visões do mesmo requisito e que todos os artefatos possuem informações necessárias para implantação destes requisitos. Ou seja, modelos contêm diagramas de classes, que representam todas as funcionalidades definidas pelos requisitos, bem como arquivos de código fonte também representam as mesmas funcionalidades.

A ferramenta foi projetada para entender se um requisito foi desenvolvido, quando um **confronto** entre modelo e código (ou uma série de comparações entre modelos e códigos) é realizado e avaliado. O confronto lança mão das métricas *Number*

of Packages (NOP), Number of Classes (NOC), Number of Methods (NOM), Number of Attributes (NOF). Durante a avaliação, a comparação dos valores calculados nos modelos e nos arquivos de código fonte e os valores estimados projetados para cada artefato, garantem uma estimativa aproximada do andamento geral do projeto.

O processo de avaliação é iniciado somente após todas as atividades de aquisição e transformação das métricas tenham ocorrido. Em sua tela principal, verificamos como a ferramenta apresenta resultados em relação aos requisitos do sistema. A cada nova versão estes requisitos são contabilizados e representados pelo gráfico apresentados pela figura a seguir:

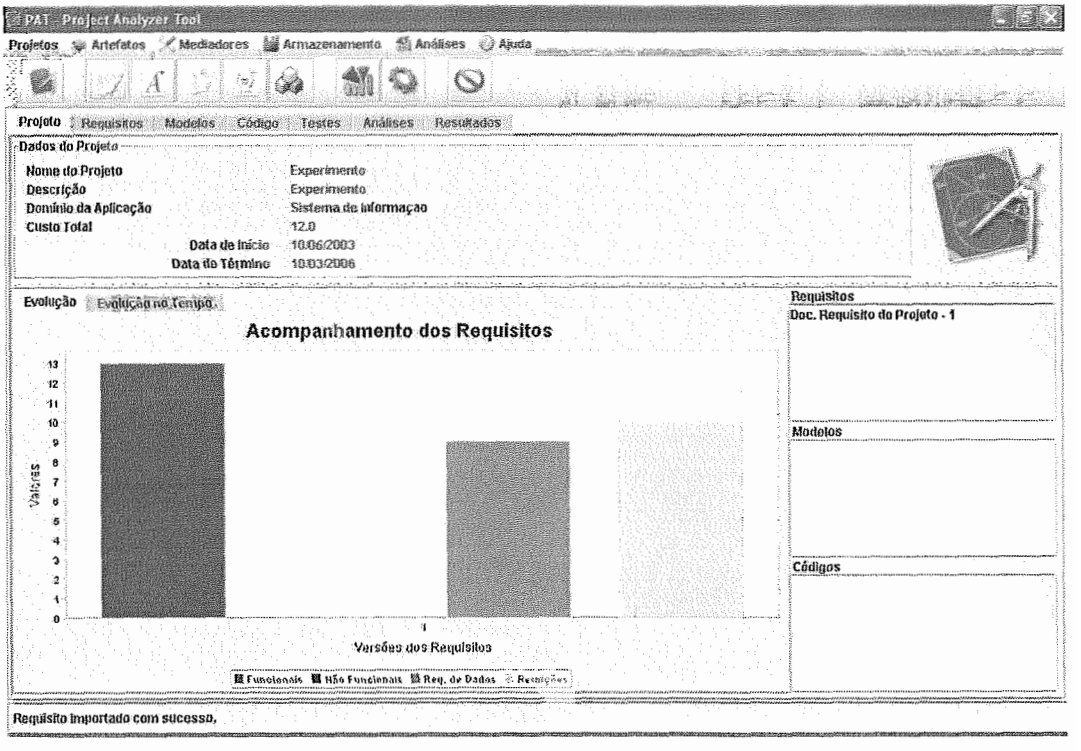

Figura 5.6. Tela do painel de controle da ferramenta.

A partir deste ponto, suponha que queiramos verificar se um requisito **R1** representado por um conjunto de classes Java **J** (incluído pacotes e classes) e por um modelo de análise **M**, já foi desenvolvido. Para responder essa pergunta, a ferramenta executa os seguintes passos:

1. Primeiro, converte-se o documento XML do modelo de análise na representação interna de metadados e também importa suas medidas para dentro da ferramenta. Em seguida, visualizamos as medidas contabilizadas para o modelo em questão:

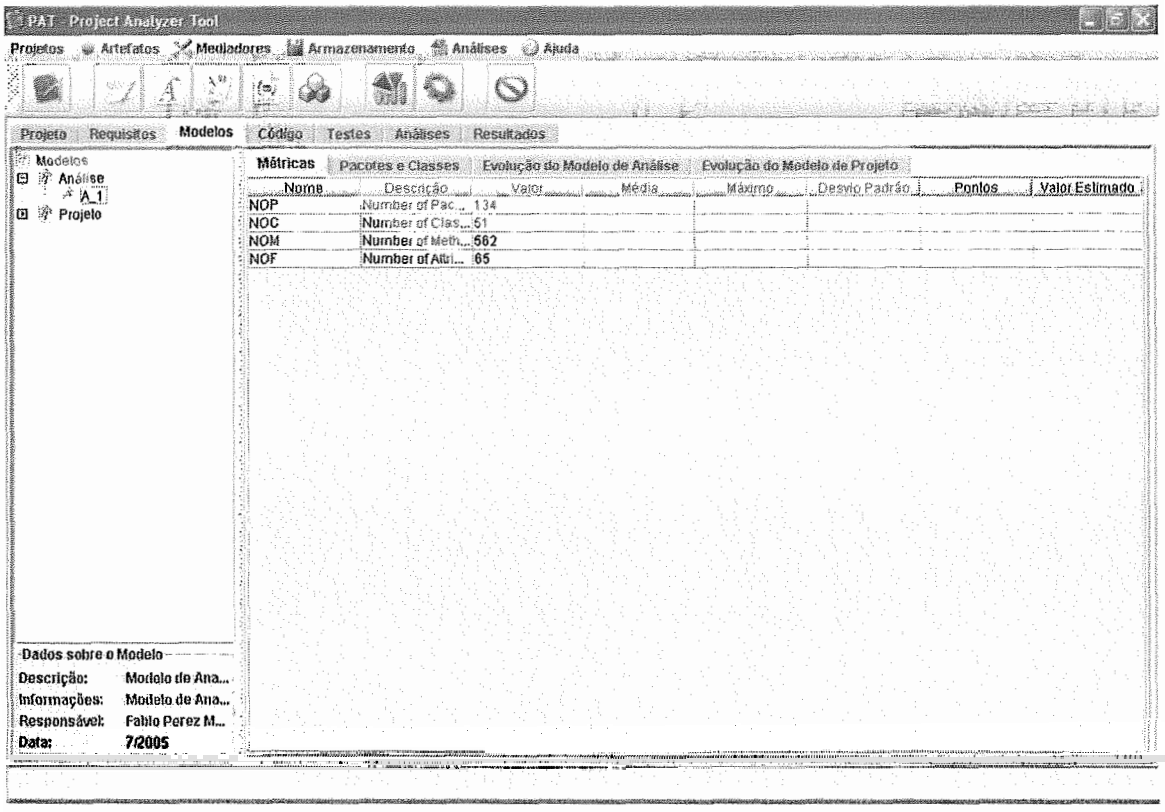

Figura 5.7. Funcionalidade do protótipo para aquisição das métricas.

Dentro desta funcionalidade da ferramenta, é possível visualizar detalhes das medições do artefato, bem como os gráficos de evolução das diferentes versões dos modelos de análise:

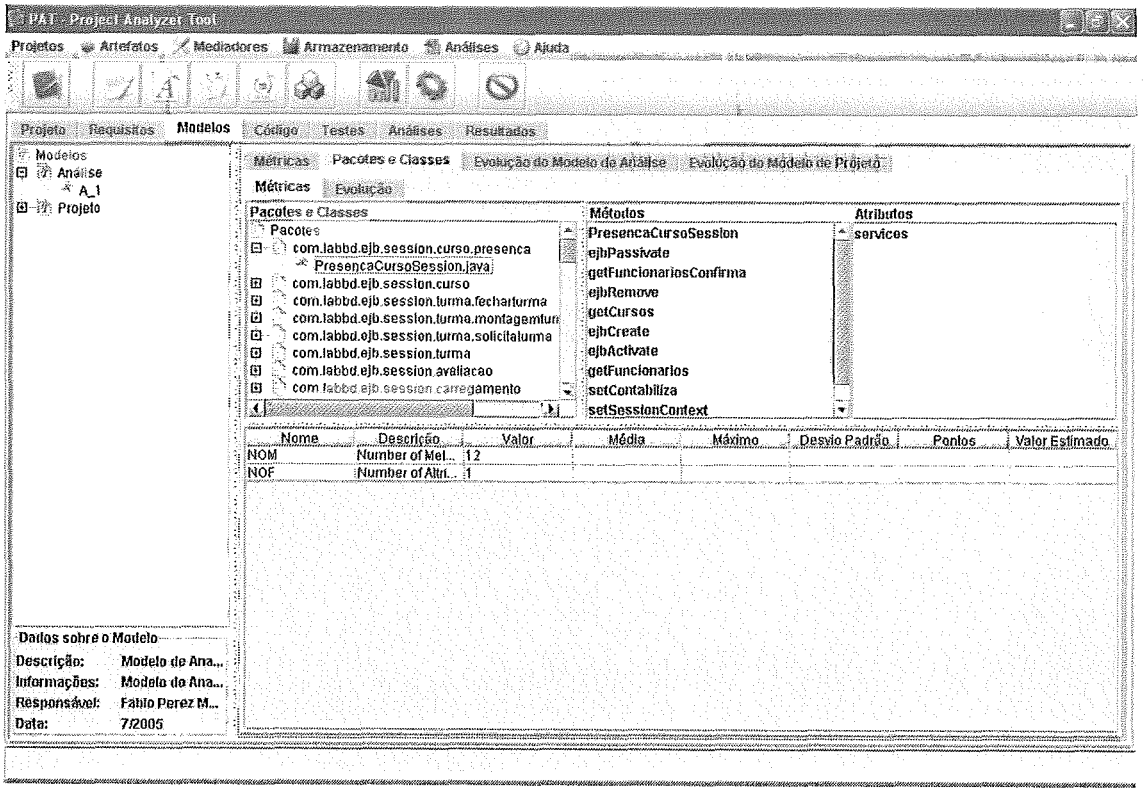

Figura 5.8. Detalhes das medições do modelo de análise.

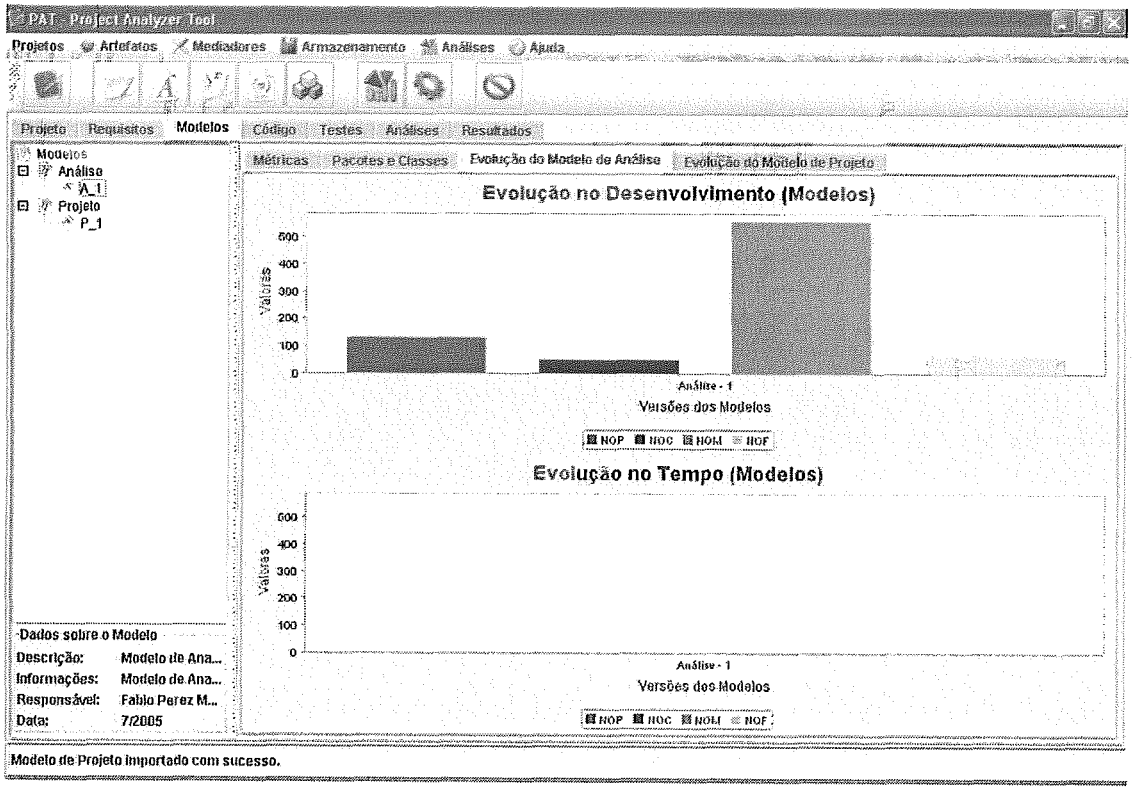

Figura 5.9. Gráficos de evolução dos modelos de análise.

2. Em seguida, convertem-se documentos XML que representam arquivos de código fonte em representação de metadados e importam-se todas as medidas contidas neste novo arquivo XML (metadados) para dentro da ferramenta. Os detalhes das medições e dos gráficos de evolução estão representados nas figuras a seguir:

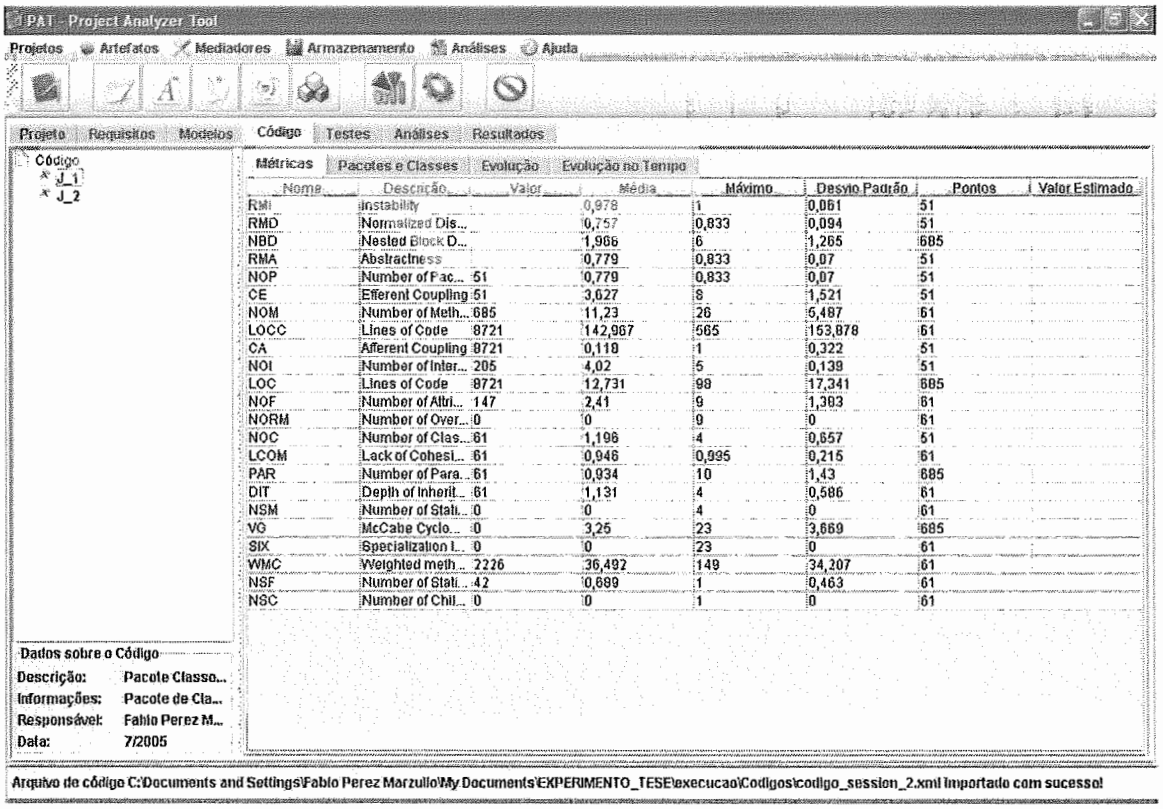

Figura 5.10. Medidas de Código Fonte.

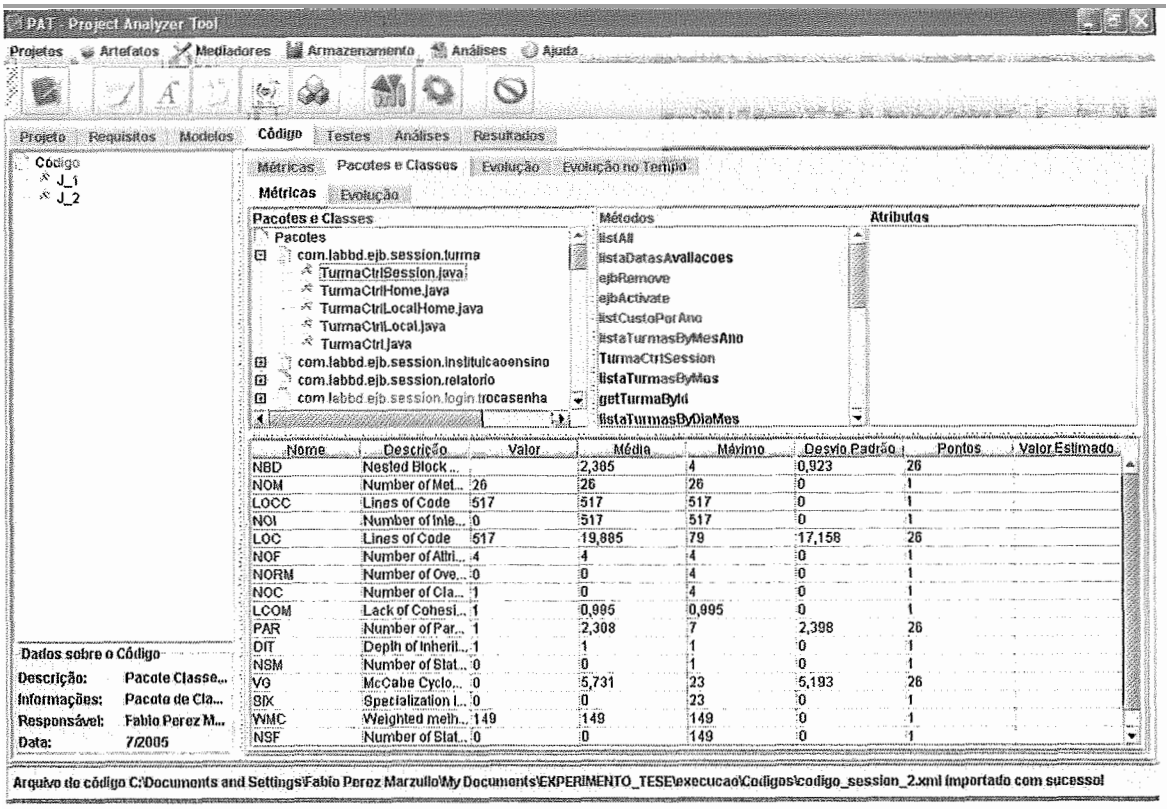

Figura 5.11. Detalhes das medidas dos arquivos fontes.

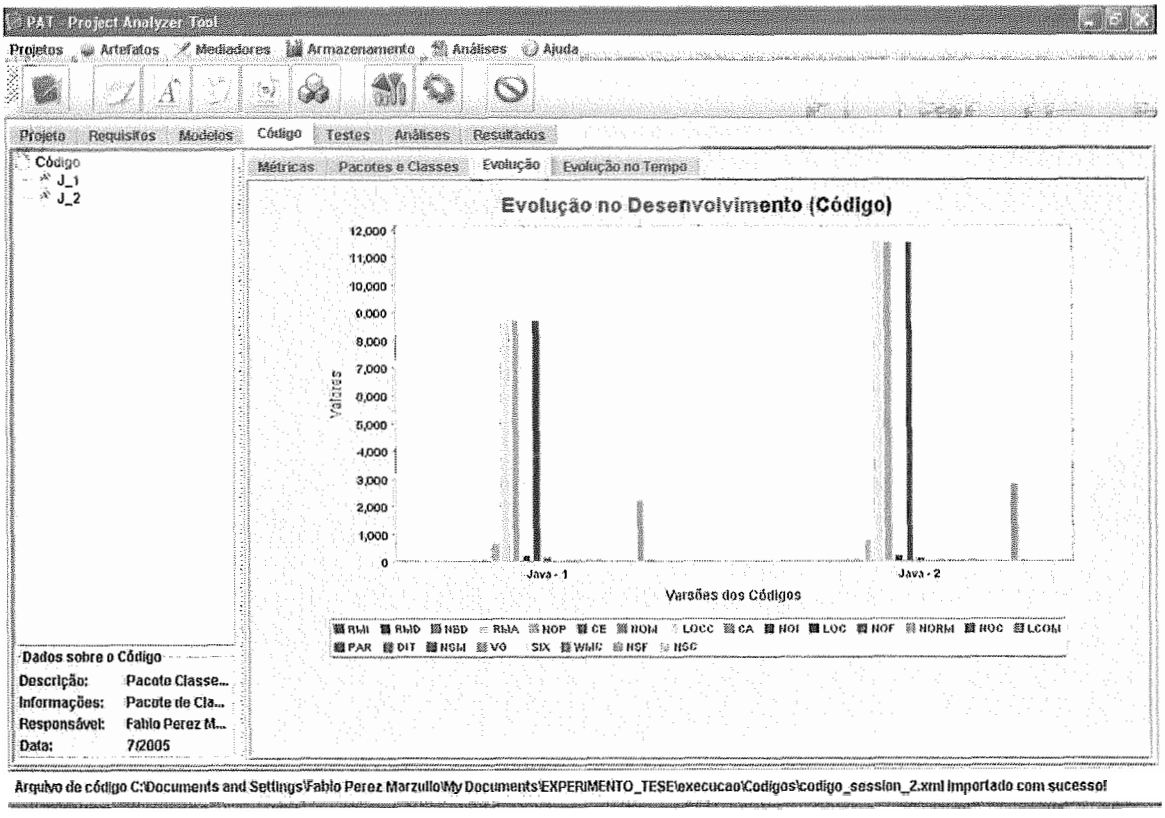

Figura 5.12 Gráfico das métricas dos pacotes de classes

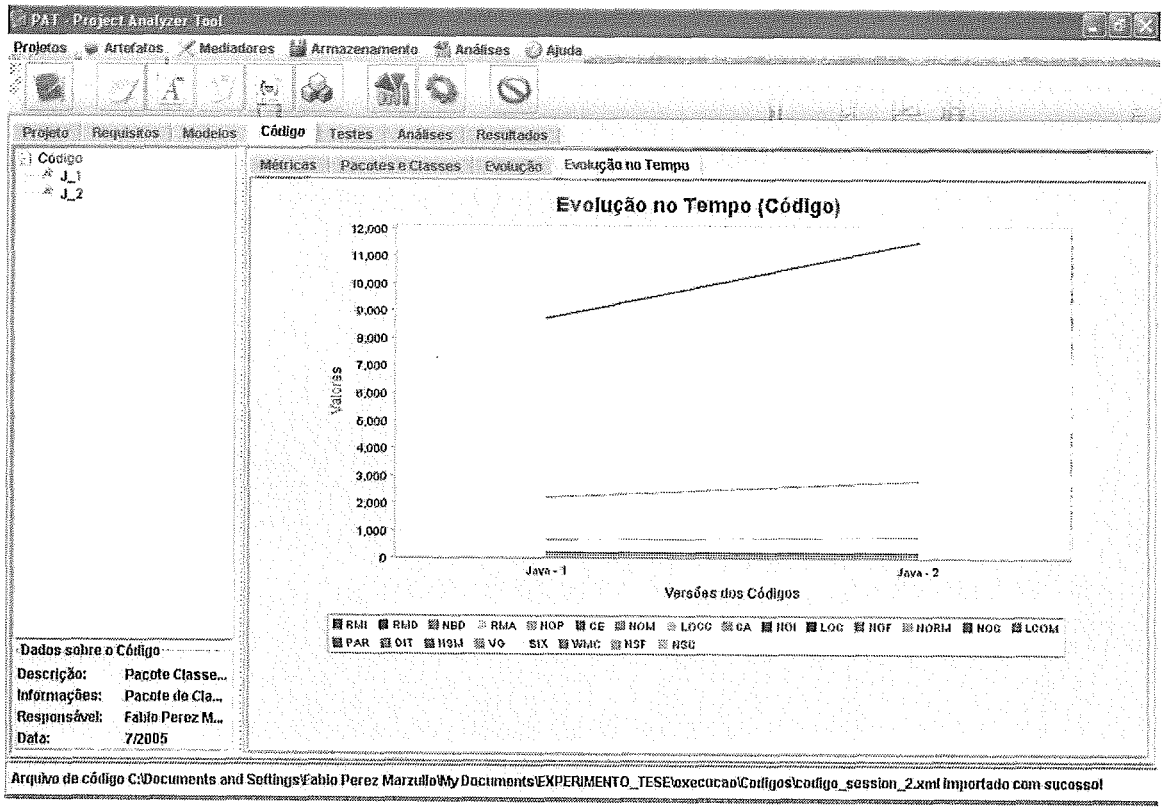

Figura 5.13. Gráfico de evolução dos arquivos fonte do projeto.

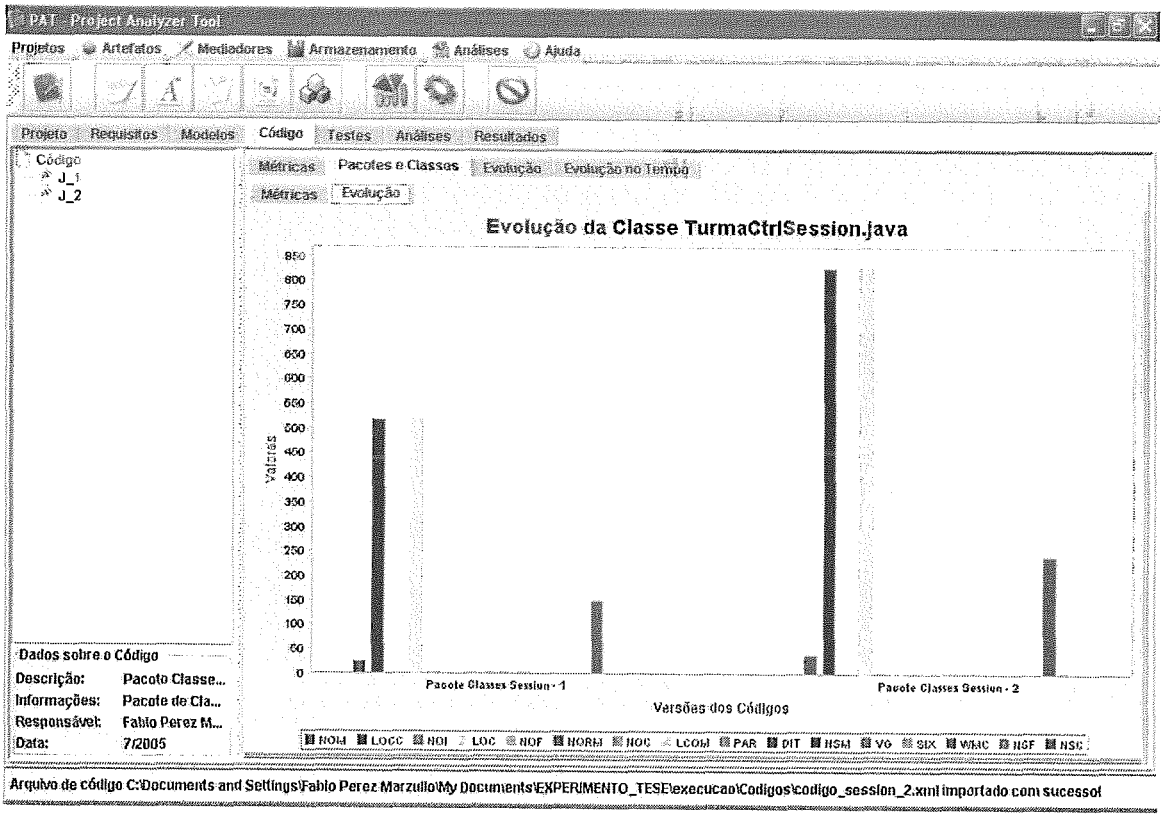

Figura 5.14. Detalhe da evolução de uma classe especificamente.

3. Por último, carregamos os dados dos testes na ferramenta para completar as informações necessárias para a execução das atividades de análise:

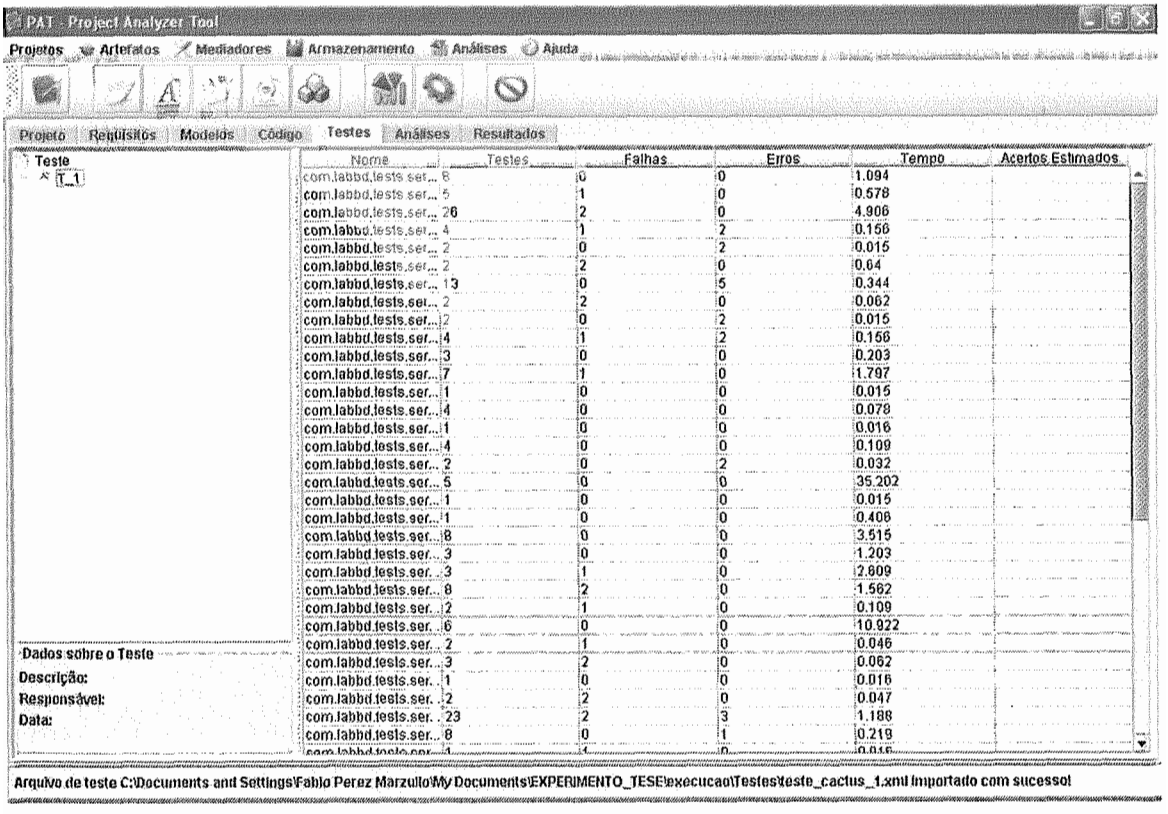

Figura 5.15. Tela com os dados carregados dos testes efetuados nas classes em avaliação.

4. A partir deste momento, a ferramenta é capaz de executar os procedimentos de análise e compara medidas de modelo e código, verificando se as classes definidas em **M** estão criadas em **J**.

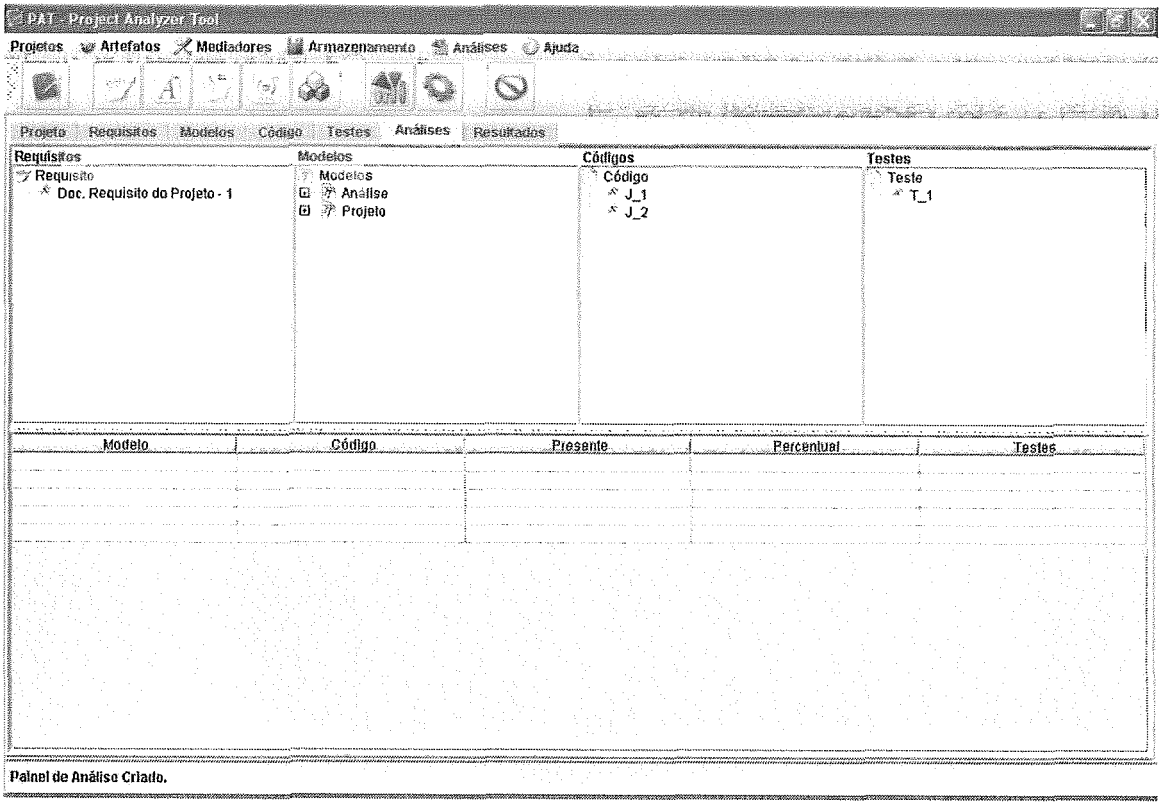

Figura 5.16. Tela de análise inicial.

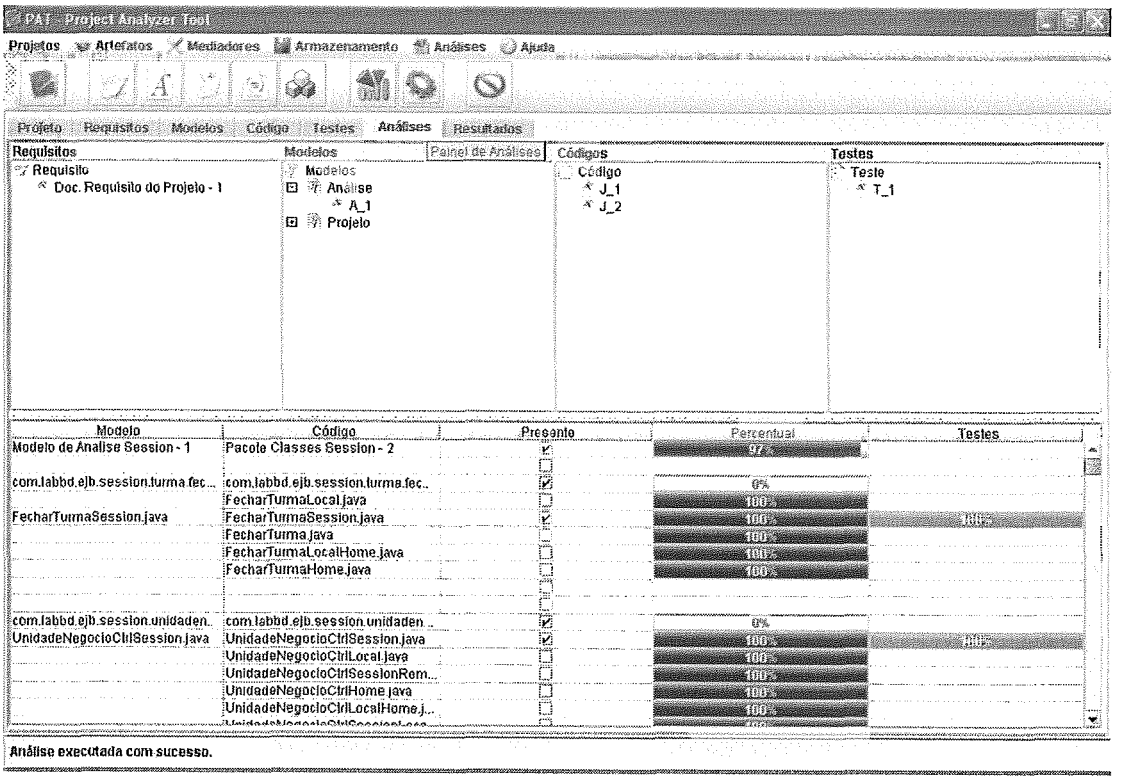

Figura 5.17. Tela de análise após a execução da análise.

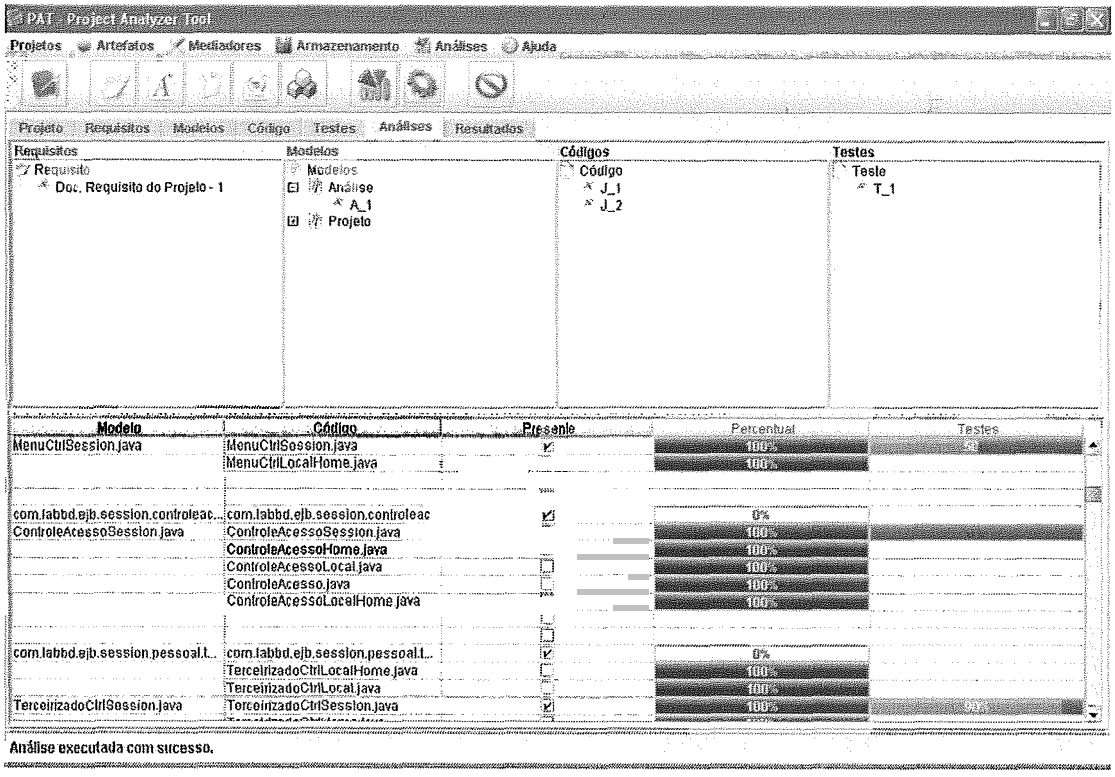

Figura 5.18. Continuação da tela anterior de análise, neste caso apresentado classes em que os testes falharam.

5. Finalmente, de acordo com os dados produzidos nas análises, relatórios são criados e apresentados para os usuários.

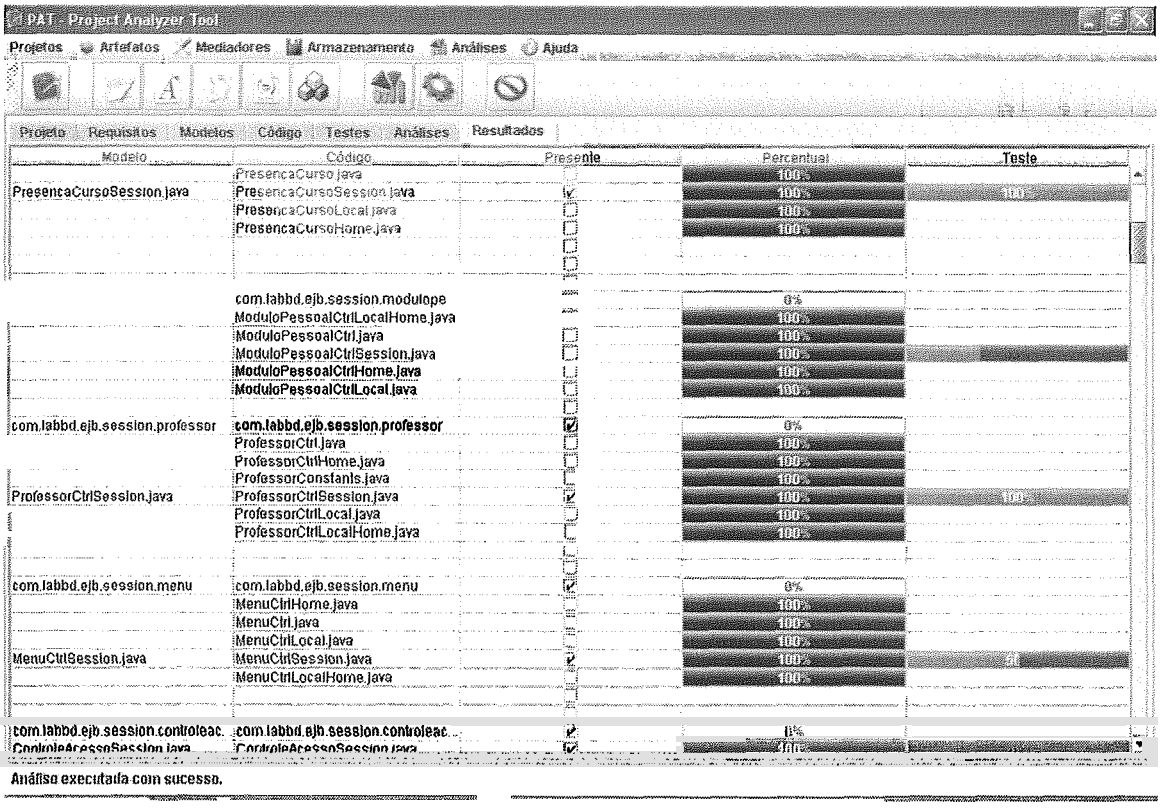

Figura 5.19. Resultados das Análises Realizadas.

Resultados obtidos do confronto apresentam o progresso de desenvolvimento em um nível de granularidade de classes e pacotes. Esta também apresenta resultados apontando o que foi feito e o que ainda deve ser desenvolvido no projeto. As informações disponíveis são satisfatórias do ponto de vista do andamento do projeto, pois utiliza as técnicas anteriormente explicadas para verificar, por exemplo, se um conjunto de classes apresenta problemas em relação aos dados coletados de testes, e dependendo do resultado, pode-se ou não assumir que está finalizada. A figura 5.20 destaca o mecanismo de análise desenvolvido pela ferramenta.

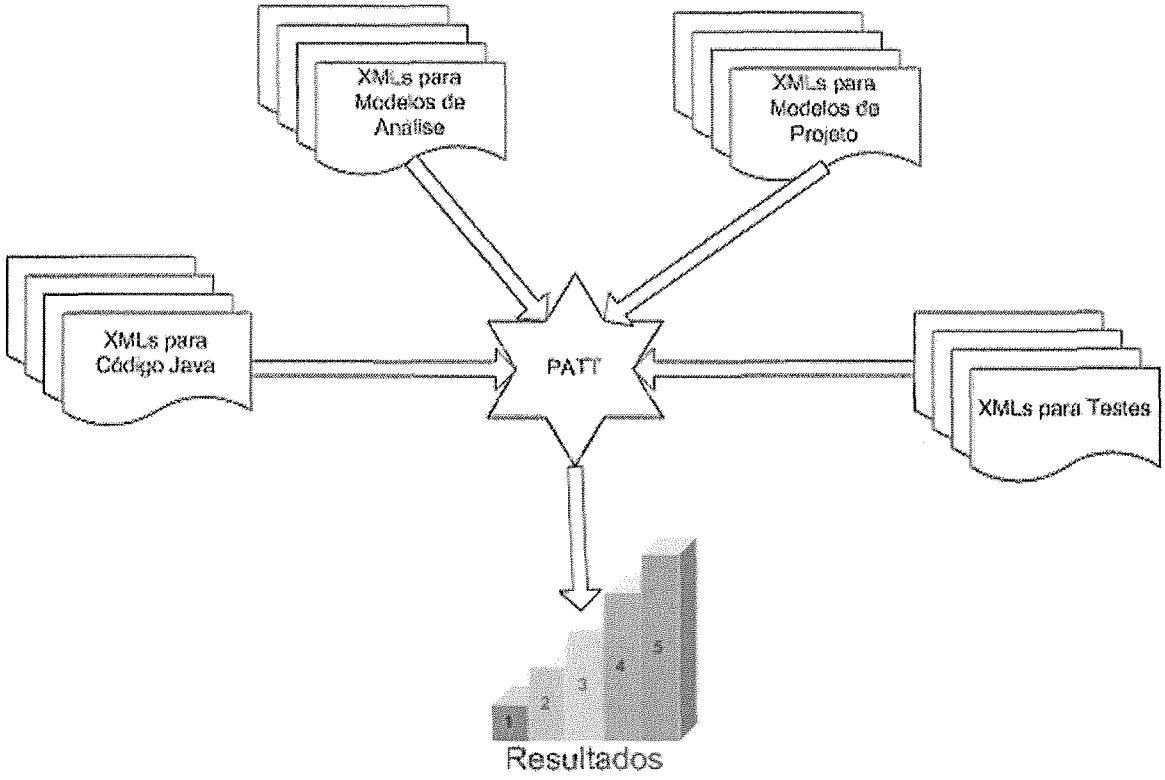

Figura 5.20. A comparação das medidas obtidas dos artefatos produz resultados relativos ao andamento do projeto.

5.8. Armazenamento

A atividade de armazenamento compõe uma etapa importante para o processo de avaliação. Tendo em vista a necessidade de criar uma base histórica com avaliações reais e válidas para suporte e apoio a tomada de decisão, armazenam-se todos os resultados inferidos do processo de análise dentro de um banco de dados XML. Como todas as informações trabalhadas dentro da plataforma são guardadas em arquivos XML, nada mais correto que utilizar um mecanismo transparente de armazenamento que se integrasse facilmente ao processo.

5.9. Conclusão

O capítulo apresentou a descrição do protótipo criado para dar apoio a abordagem descrita no capítulo anterior. Forneceu uma visão de como o processo de avaliação e medição de software foi implementado, utilizando uma solução de baixo custo e de fácil utilização. Unindo ferramentas de desenvolvimento e o protótipo criado em um único *framework* de avaliação, demonstrou-se como o processo de análise proposto pode ser posto em prática e como seus resultados podem ser utilizados para aquisição de informações relevantes ao desenvolvimento do projeto.

A seguir apresentamos um cenário de uso a abordagem de acompanhamento e do protótipo, através da análise em um projeto de software real. Este estudo identifica como a ferramenta foi utilizada e se os resultados gerados foram ao encontro das premissas discutidas até agora.

Capítulo 6

Um Cenário de Uso do Protótipo

6.1. Introdução

Com objetivo de avaliar o protótipo desenvolvido, o estudo apresentado neste capítulo fornece dados interessantes quanto ao objetivo principal deste trabalho: avaliar o andamento de projetos de software utilizando artefatos construídos durante o mesmo.

Partindo-se da premissa que o grupo de desenvolvedores envolvido no projeto deveria ser capaz de produzir os artefatos de interesse, e que tais artefatos possuíssem boa representatividade do projeto em questão, a observação dos dados gerados pelo protótipo permitiu entender o comportamento do desenvolvimento de um sistema de software orientado a objetos.

6.2. Premissas para Utilização do Protótipo

O contexto de utilização do protótipo se enquadra na necessidade de um processo de desenvolvimento adequado e formalmente baseado em normas e padrões de processo. Ou seja, para que uma empresa consiga utilizar toda infra-estrutura de avaliação é necessário que possua um processo de desenvolvimento formal e gerenciado. Essa regra determina que, se a empresa possui esse nível de evolução gerencial, consegue construir os artefatos necessários para as avaliações, e conseqüentemente, consegue manter um controle de versão adequado.

Dentro do quadro especificado pelos capítulos anteriores, os artefatos que possuem maior relevância são aqueles que podem fornecer mais informações, como documentos de requisitos, modelos de negócio e projeto, conjunto de pacotes de arquivos de código fonte, e o conjunto de testes realizados para garantia da qualidade do produto. Dessa forma a relação de artefatos abaixo se mostrou essencial para a correta condução do estudo e utilização:

a) Documento de Requisitos:

A utilização de documentos de requisitos é essencial para o processo de avaliação. Seu processo de análise é iniciado após seu carregamento na ferramenta. O mecanismo de importação foi criado de maneira a permitir que as informações pertencentes ao documento pudessem ser configuradas. Um arquivo XML é utilizado com este propósito e determina, através de marcas no documento, como este deve ser trabalhado. O documento abaixo representa um arquivo de configuração, seu conteúdo foi modelado a partir de especificações recomendadas pelo IEEE [32].

```
<requisite-marks>
     <header>Introdução</header>
     <objective>Objetivo</objective>
     <extras>Glossário</extras>
     <functional-req>Requisitos Funcionais
          <functional-req-submark>Requisito
     Funcional</functional-req-submark>
     </functional-req>
     <data-req>Requisitos de Dados
          <data-req-submark>
               Requisito de Dados
          </data-req-submark>
     </data-req>
     <nonfunctional-req>Requisitos Não Funcionais
          <nonfunctional-req-submark>
               Requisito Não Funcional
          </nonfunctional-req-submark>
     </nonfunctional-req>
     <constraints>Restrições
          <constraints-submark>
               Restrição
          </constraints-submark>
     </constraints>
</requisite-marks>
```

b) Modelo de Análise/Negócio:

Modelos de análise são utilizados como elo entre o documento de requisitos e os pacotes de código fonte. Através das informações extraídas nestes modelos, é possível

determinar se uma característica pensada a partir de um requisito, já foi implementada em código.

c) Modelo de Projeto

O mesmo pensamento pode ser utilizado para modelos de projeto. A partir de suas informações, é possível identificar características em código, que refletem a especificação do documento de requisito. Uma vantagem, contudo, está no nível de refinamento que um modelo de projeto possui sobre um modelo de análise. Como este tenta se aproximar ao máximo da implementação do código, conseqüentemente, acaba por possuir mais informações, o que aumenta a capacidade de avaliação da ferramenta.

d) Arquivos de Código Fonte:

Essenciais para a ferramenta, são estas informações que determinam o que foi desenvolvido no projeto. São essas características que servem de base para a avaliação, tanto para a comparação com os modelos como para a verificação com os resultados dos testes.

e) Resultados dos Testes:

São essas informações que dão o veredicto final na análise. Após as comparações e avaliação com os outros artefatos, a ferramenta verifica se o código está efetivamente funcionando. Somente aquelas classes que passam em todos os testes podem ser avaliadas como fechadas.

Todas as ferramentas utilizadas em apoio ao desenvolvimento são abertas, ou seja, de código fonte livre, o que facilita a adoção do processo definido dentro da empresa. Além disso, a ferramenta tem como público alvo gerentes de desenvolvimento, não se preocupando em categorizar perfis dentro da plataforma. Como a avaliação requer a utilização de diferentes ferramentas e o envolvimento de todos os responsáveis pela

construção do software, o trabalho de agregar e produzir os resultados fica a cargo do gerente.

Todas as informações relativas aos artefatos, bem como arquivos de configuração, foram armazenadas em arquivos XML. Com a necessidade de generalização e abrangência de diferentes técnicas de programação, a forma mais adequada de representação e manipulação das informações mostrou ser a utilização de arquivos XML. Somado a isso, um artefato só pode ser entendido pela ferramenta, quando é convertido para o nosso modelo de metadados, conforme apresentado em capítulos anteriores.

6.3. Planejamento

O estudo ocorreu em um projeto real de um sistema desenvolvido para uma grande empresa petrolífera brasileira. Este projeto tinha como objetivo construir um sistema de controle de certificação profissional através da adoção de técnicas de gestão do conhecimento e ensino à distância. O projeto foi desenvolvido em plataforma WEB, utilizando-se linguagem de programação Java. A infra-estrutura de desenvolvimento adotada por parte da equipe foi definida no capítulo de apresentação do protótipo.

Dentro do projeto utilizamos os pacotes que continham as regras de negócio, excluindo pacotes de ajuda ou de interface. Ao todo foram contemplados dezoito requisitos, conforme listados abaixo:

1. Controle de Acesso;
2. Controle de Provas;
3. Controle de Cursos;
4. Impressão de Diplomas;
5. Manutenção de Especialidade;
6. Controle de Formação;
7. Internacionalização;
8. Controle de Instituições de Ensino;
9. Controle de Kits Didáticos;
10. Manutenção de Funcionários;
11. Manutenção de Papel;

12. Manutenção de Professores;
13. Manutenção de Questões de Prova;
14. Relatórios Gerenciais;
15. Manutenção de Módulos;
16. Manutenção de Unidades Federativas;
17. Manutenção de Unidades de Negócio;
18. Controle de Documentos de Reação;

Atendendo a estes requisitos, um modelo de análise foi desenvolvido para entendimento das regras de negócio e um conjunto de quarenta pacotes com uma classe e quatro interfaces cada foi construído, e para cada classe de negócio foi criada uma classe de teste que tinha a responsabilidade de validar seus métodos. O modelo foi desenvolvido utilizando a ferramenta ArgoUML [24] e os pacotes de classes e os testes foram construídos pelo IDE Eclipse [23], utilizando o *"framework"* de testes Cactus [87].

6.4. Participantes

O estudo foi conduzido pelo gerente de desenvolvimento responsável pelo projeto a partir dos artefatos produzidos pelos seus desenvolvedores. Dois grupos foram definidos:

1. Analistas: responsáveis pela aquisição das regras de negócio e do entendimento do projeto;
2. Desenvolvedores: responsáveis pela construção do software em questão.

6.5. Procedimentos

As atividades que foram executadas durante os procedimentos de utilização e avaliação da ferramenta estão descritas conforme os itens listados abaixo:

1. Construção do software utilizando um processo de desenvolvimento que atendesse aos requisitos do processo de avaliação do protótipo;

2. Aquisição dos artefatos construídos (dentro de um controle de versão rígido);
3. Exportação dos dados dos artefatos para documentos XML próprios de cada ferramenta;
4. Transformação dos XML próprios em XML com o padrão dos metadados definidos para o protótipo;
5. Carregamento dos dados para dentro do protótipo;
6. Execução da avaliação;
7. Obtenção de Resultados.

Os sete itens descrevem como a ferramenta foi executada. Dentro do projeto avaliado, obtivemos duas versões de documentos de requisitos, seis versões de modelos de análise, oito versões dos pacotes de código fonte e uma versão dos resultados dos testes que foram executados para avaliação do produto final. Neste contexto os resultados que se seguem fornecem informações interessantes de como um projeto desse porte se comporta durante seu desenvolvimento.

6.6. Resultados

Duas visões foram consideradas durante a utilização da ferramenta para buscar os resultados corretos. A primeira visão tinha o objetivo de projetar o resultado esperado em cada etapa da avaliação. Neste caso, o gerente foi responsável por fazer uma projeção dos resultados esperados do desenvolvimento. Ou seja, o gerente tinha que indicar, por exemplo, o número de pacotes que a versão deveria ter; o número de classes; métodos etc. A segunda visão representava a leitura real dos dados extraídos dos artefatos. Basicamente, o número real de pacote, classes, métodos etc. Vale ressaltar que a ferramenta possui características de aquisição de dados estimados manualmente ou automáticos (gerados a partir de dados históricos), conforme explicado no capítulo da abordagem da ferramenta.

De posse desses dados, o processo de acompanhamento apresentou-se de maneira simples e de acordo com o planejamento esperado. Em média, mesmo sob os auspícios de um processo formal, o desenvolvimento real mantinha um atraso de 14,63% no número

de classes desenvolvidas. A tabela 6.1-A apresenta como o andamento do desenvolvimento ocorreu de acordo com a versão do modelo de análise.

Tabela 6.1 – A - Valores para Modelos de Análise.

Versões	Valor Estimado	Valor Real	Percentual de Atraso
1	50	44	12,00 %
2	55	47	14,54 %
3	55	47	14,54 %
4	60	50	16,66 %
5	60	51	15,00 %
6	60	51	15,00 %

Em relação aos pacotes de código fonte o atraso médio foi de 5,35 %, conforme apresentado na tabela 6.1-B.

Tabela 6.1 – B - Valores para Arquivos de Código Fonte.

Versões	Valor Estimado	Valor Real	Percentual de Atraso
1	50	45	10,00 %
2	60	56	6,66 %
3	60	56	6,66 %
4	65	59	9,23 %
5	65	61	6,15 %
6	65	62	4,61 %
7	70	67	4,28 %
8	80	79	1,25 %

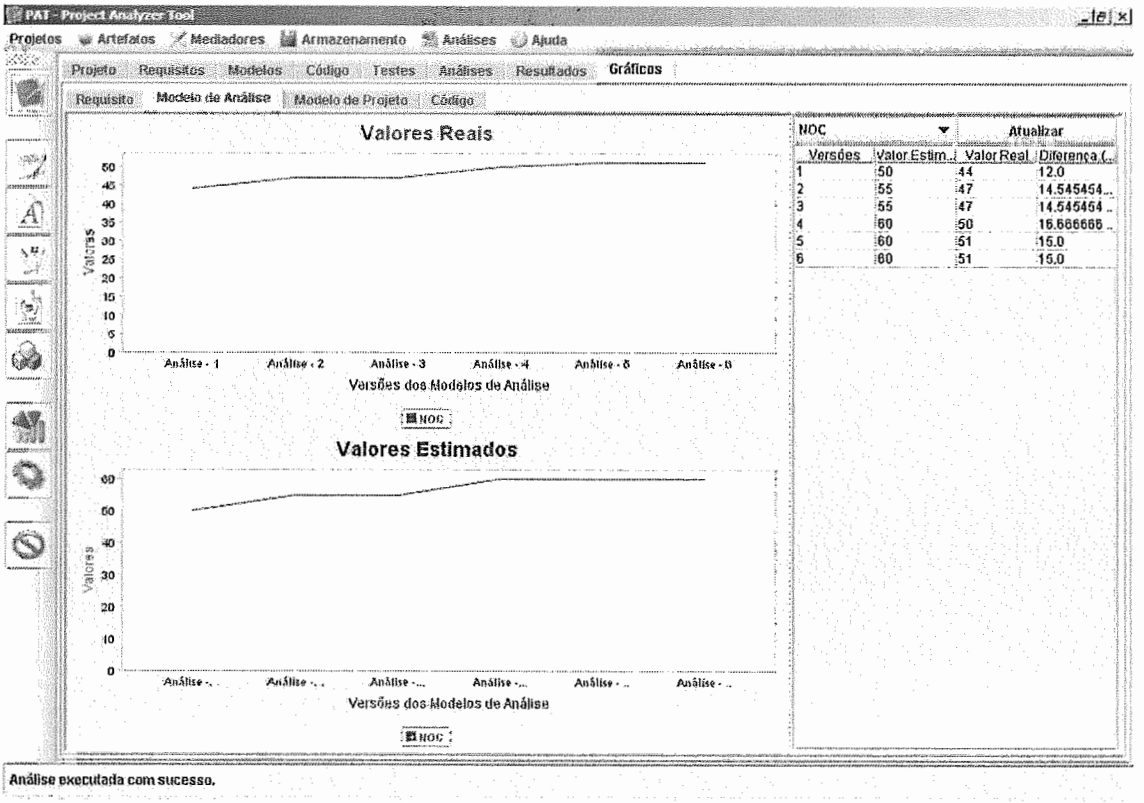

Figura 6.1-A – Imagem da Ferramenta: Avaliação dos Modelos de Análise.

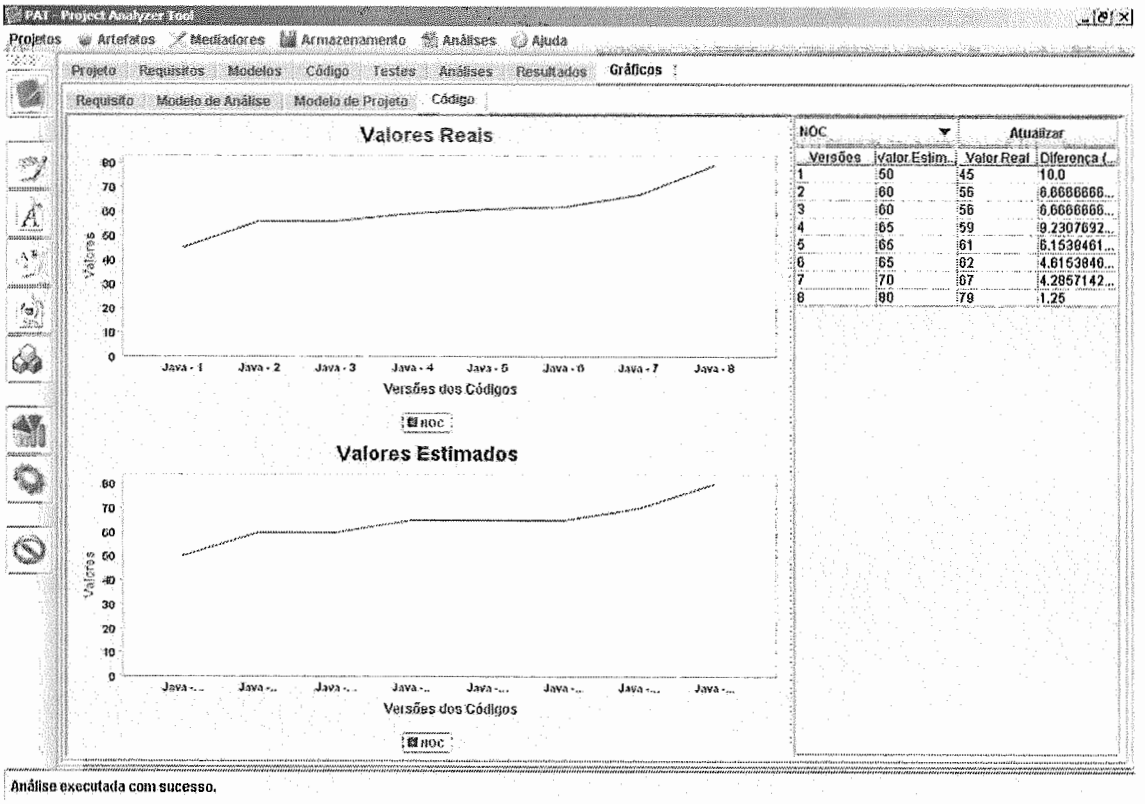

Figura 6.1-B – Avaliação dos Pacotes de Código Fonte.

A ferramenta também proporciona uma avaliação em relação ao percentual de desenvolvimento dos requisitos do sistema. A tabela 6.2 identifica esta evolução comparativamente com o desenvolvimento dos pacotes de código fonte.

Tabela 6.2 – Evolução de Desenvolvimento dos Requisitos do Sistema. Valores baseados na comparação entre os documentos de requisitos e os pacotes de código da versão 5.

Requisitos	Versão	Percentual de Desenvolvimento
Controle de Acesso	1	100%
Controle de Provas	1	55%
Controle de Cursos	1	89%
Impressão de Diplomas	1	11%
Manutenção de Especialidade	1	100%
Controle de Formação	1	90%
Internacionalização	1	22%
Controle de Instituições de Ensino	1	91%
Controle de Kits Didáticos	1	97%
Manutenção de Funcionários	1	78%
Manutenção de Papel	1	100%
Manutenção de Professores	1	92%
Manutenção de Questões de Prova	2	81%
Relatórios Gerenciais	2	93%
Manutenção de Módulos	2	100%
Manutenção de Unidades Federativas	2	100%
Manutenção de Unidades de Negócio	2	99%
Controle de Documentos de Reação	2	34%

Uma outra forma de avaliação pode ser feita utilizando-se diretamente um arquivo fonte ou um pacote. Em se tratando de requisitos, entendemos que são representados por um conjunto de pacotes de arquivos de código fonte. Contudo o gerente tem a opção de identificar pontualmente a evolução de um arquivo fonte, particularmente importante, ou

que defina uma estrutura central da aplicação. Este arquivo pode ser necessário para continuação de outros componentes do software, implicando em um atraso prejudicial ao projeto caso não fique pronto no tempo requerido. A figura 6.2 identifica como os arquivos fonte podem ser avaliados e identificados como concluídos, ou em desenvolvimento. Também apresenta como os testes fornecem informações a respeito desse arquivo fonte, tornando-se fundamental na avaliação final do arquivo.

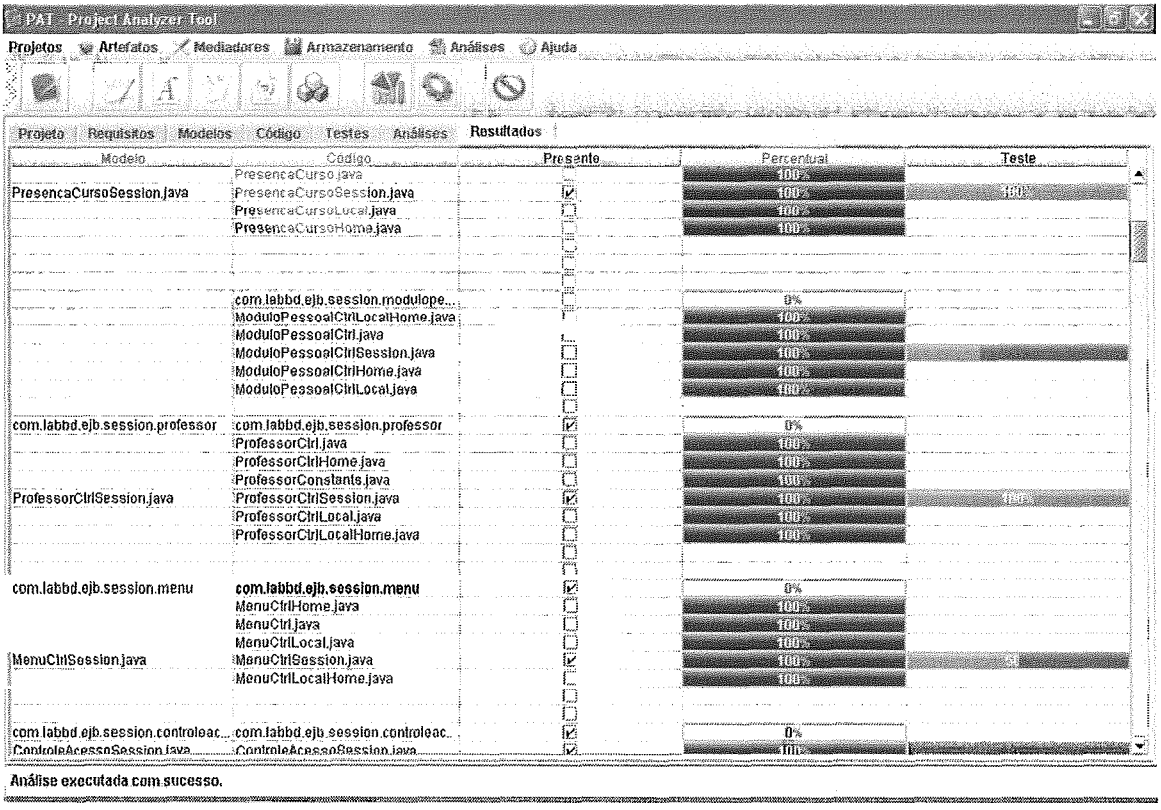

Figura 6.2 – Avaliação individual das classes.

A primeira coluna identifica o modelo de análise responsável por definir os componentes necessários na construção de um requisito. A segunda coluna apresenta as classes construídas, efetivamente, para atender a modelagem, e transitivamente atender a um determinado requisito. Em seguida, na terceira coluna, temos um indicador de presença. Este indicador determina se um componente descrito no modelo de análise está presente no pacote de classes. A quarta coluna apresenta o resultado da contabilização das métricas extraídas do modelo de análise e da classe, e verificamos se pontos chave,

como linhas de código, número de métodos e atributos estão de acordo com o que foi apresentado no modelo de análise. Por fim, a última coluna apresenta os resultados dos testes executados nos arquivos fonte e demonstra se este está fechado, ou seja, se passou em todos os testes, e conseqüentemente indica que a classe está finalizada, ou, em caso de falha, que as classes definidas no modelo de análise não foram corretamente desenvolvidas e, portanto, devem sofrer um novo processo de desenvolvimento e testes.

6.7. Métricas Utilizadas

A seguir é apresentado o conjunto de métricas que fizeram parte dos procedimentos de avaliação da ferramenta. Todas as métricas foram extraídas de acordo com os cálculos efetuados pelo plugin *metrics* [88] em conjunto com a IDE de desenvolvimento Eclipse [23].

Métrica 1	**NBD - Nested Block Depth**
Definição	Valor total da profundidade dos blocos aninhados dentro do código.

Métrica 2	**NOM - Number of Methods**
Definição	Número total de métodos calculados dentro de uma classe.

Métrica 3	**NSM – Number of Static Methods**
Definição	Total de métodos estáticos dentro da classe.

Métrica 4	**NOI - Number of Interfaces**
Definição	Total de interfaces dentro de um pacote.

Métrica 5	**LOC - Lines of Code**
Definição	Total de linhas de código dentro de um escopo. O escopo pode ser uma classe um pacote ou um método. Somente são avaliadas linhas efetivamente válidas, ou seja, linhas em branco e linhas comentadas não são contabilizadas.

Métrica 6	NOF - Number of Attributes
Definição	Total de atributos dentro de uma classe.

Métrica 7	NORM - Number of Overridden Methods
Definição	Total de métodos de uma classe que redefinem métodos herdados.

Métrica 8	NOC - Number of Classes
Definição	Total de Classes em um pacote.

Métrica 9	PAR – Number of Parameters
Definição	Total de parâmetros de um método.

Métrica 10	DIT - Depth of Inheritance Tree
Definição	Distância da classe Object até a classe avaliada.

Métrica 11	CE - Efferent Coupling
Definição	Número de classes de um pacote que dependem de classes fora do pacote.

Métrica 12	VG - McCabe Cyclomatic Complexity
Definição	Contabiliza o total de fluxos possíveis dentro de um bloco de código. Efetivamente avaliados em métodos, cada vez que uma escolha é identificada, o valor é incrementado em 1.

Métrica 13	MLOC - Method Lines of Code
Definição	Número de linhas de código dentro de um método.

Métrica 14	SIX - Specialization Index
Definição	Média do índice de especialização da classe. Calculada pela fórmula: NORM * DIT / NOM. Esta métrica é aplicada a classes.

Métrica 15	WMC - Weighted methods per Class
Definição	Somatório das complexidades ciclomáticas de todos os métodos da classe.

Métrica 16	NSF - Number of Static Attributes
Definição	Total de atributos estáticos em uma classe.

Métrica 17	NSC - Number of Children
Definição	Total de filhos em uma classe.

6.8. Conclusões

O estudo realizado apresentou informações relevantes quanto ao uso do protótipo no processo de avaliação do andamento do projeto. Com a aplicação das regras definidas dentro do trabalho, foi verificado que é possível avaliar o andamento de um projeto utilizando seus artefatos em conjunto com um planejamento adequado ao tipo de software alvo.

A intenção era tentar obter informações relacionadas ao projeto que pudessem ajudar a identificar indícios sobre o seu andamento. O nível de riqueza das informações obtidas, por sua vez, demonstrou que é possível utilizar a abordagem para produzir resultados satisfatórios e permitiu identificar pontos de falhas recorrentes dentro do desenvolvimento.

Atuando como um painel de controle, o protótipo conseguiu superar expectativas. Trabalhando como uma ferramenta capaz de manipular facilmente dados históricos, a visualização evolutiva do desenvolvimento do projeto tornou-se tarefa simples.

Apesar de ainda existirem pontos a serem melhorados, sem dúvida, o protótipo se comportou como um agente importante no entendimento do desenvolvimento de projetos de software.

Capítulo 7

Considerações Finais

7.1 Considerações sobre a Dissertação

Este trabalho apresentou uma abordagem para o acompanhamento de projetos de software baseada na extração, processamento e visualização de métricas de produto e qualidade retiradas dos artefatos gerados no processo de desenvolvimento de software.

O foco na visualização da evolução histórica e da visualização dos dados de produto e qualidade foi determinante para que a plataforma pudesse ser usada como mecanismo de aquisição de informações do projeto.

A abordagem foi complementada com definição da plataforma de avaliação, que incluiu a escolha de ferramentas adequadas à extração das métricas e pela criação de uma ferramenta que centralizasse essas métricas e permitisse visualizá-las em forma similar a um painel de controle do projeto.

O protótipo funcional do trabalho apresentou uma solução ao problema, e foi capaz identificar focos de problemas enfrentados durante o desenvolvimento do projeto, como, por exemplo, identificar conjuntos de classes que constantemente eram reprovadas em testes. Dessa forma foi possível identificar focos potenciais de problemas e resolvê-los de forma razoavelmente rápida. Além disso, a disponibilização de mecanismos de importação de estimativas ajudou a identificar como o desenvolvimento do software se comportou ao longo do ciclo de vida do projeto.

As equipes em geral são extremamente heterogêneas e conseqüentemente, mesmo quando as características de um projeto são semelhantes às características de outro projeto, diferenças na equipe de desenvolvimento podem caracterizar falhas ou sucessos durante o desenvolvimento. Com a utilização do protótipo o gerente consegue obter informações capazes de identificar problemas no desenvolvimento e reprimir eventuais falhas com orientações e treinamento adequado ao profissional responsável.

Quanto ao projeto avaliado, conseguimos obter informações satisfatórias sobre o seu comportamento ao longo do tempo, quando identificamos a evolução dos requisitos de acordo com versões incrementais do mesmo. Conseguimos, por exemplo, entender o processo de evolução do desenvolvimento do sistema.

A abordagem conseguiu contribuir, principalmente, no que diz respeito ao tipo de informação que o protótipo da ferramenta produz. Agindo como um painel de controle,

esta fornece dados interessantes sobre o projeto, principalmente no que diz respeito ao controle da qualidade e no tempo de desenvolvimento.

O tratamento histórico foi bem aproveitado, pois a facilidade de armazenamento e recuperação das informações das diferentes versões dos artefatos, permitiu visualizar o comportamento dos mesmos durante o desenvolvimento do projeto.

Restrições enfrentadas ao longo do estudo determinaram que as ferramentas usadas como apoio possuem limitações em relação ao tamanho do projeto avaliado. Caso o projeto seja muito grande, ou seja, possua um número elevado de classes, essas ferramentas acabam por perder desempenho na produção das medições.

7.2 Trabalhos Futuros

Durante a pesquisa e o desenvolvimento da ferramenta de avaliação de projetos, identificamos algumas oportunidades de desenvolvimento que não puderam ser implantadas devido o tempo disponível para conclusão dos trabalhos.

7.2.1 Ferramenta de Análise para Plataforma Internet

Uma oportunidade encontrada durante o desenvolvimento do protótipo foi a criação de uma ferramenta de comunicação de resultados via sistema WEB. De acordo com os resultados gerados, esta aplicação ajudaria na propagação das informações, facilitando o acesso as informações pertinentes ao projeto.

A integração seria feita de forma bastante simples, uma vez que a ferramenta de análise estaria toda desenvolvida em Java. A base central de informações é representada por um banco de dados XML, sendo assim, todos os dados manipulados seriam mais bem avaliados e, conseqüentemente, mais facilmente propagados.

A ferramenta também poderia ser complementada por módulos de análises on-line, os quais permitissem que algumas análises pudessem ser realizadas remotamente.

Este trabalho se encontra atualmente em desenvolvimento como projeto de graduação para o curso de Bacharelado em Informática na Universidade Federal do Rio de Janeiro.

É perfeitamente viável a exploração de novos caminhos e definições de novas atividades para melhoria da plataforma de avaliação. Opções como a melhoria da gestão dos resultados através de técnicas de gestão de competências pode ajudar no processo de aquisição de soluções para projetos de desenvolvimento, como os que foram apresentados nesta dissertação.

Apêndice A

Banco de Dados XML

Introdução

Este apêndice faz uma breve introdução ao tema de Banco de Dados XML Nativo, apresentando o funcionamento destes tipos de bancos de dados, citando como exemplo o banco utilizado pela ferramenta.

Bancos de Dados XML Nativo

O termo Banco de Dados XML Nativo (NXD – Native XML Database) foi designado àqueles programas capazes de armazenar arquivos XML em sua forma nativa, texto.

Conforme definido em [85 e 86], um sistema de armazenamento de arquivos XML deve tratá-los como uma coleção de documentos que são formados por diferentes partes. O banco de dados deve ser capaz de manipular as coleções, bem como informações pertencentes a elas. Um Banco de Dados XML deve possuir um papel que não seja somente de um repositório de dados estruturados, portanto, deve ser capaz de gerenciar persistência, independência de dados, integração, controlar acesso, versão, garantir integridade, redundância, consistência, ser capaz de se recuperar quando ocorrerem falhas e basear suas estrutura arquitetural em padrões bem definidos.

Um banco de dados XML deve possuir características específicas para garantir um mínimo de funcionalidades que demonstrem sua capacidade e eficiência no controle de documentos XML. Estas características são: (1) Criação dinâmica de Coleções; (2) Transformações Automáticas; (3) Acesso ao nível de elementos; (4) Acesso direto a elementos (versão de componentes); (5) Controle de Versão; (6) Mecanismos de Busca; (7) Workflow baseado em documentos;

Finalizando, toda informação armazenada em Bancos de Dados XML, deve ser persistida de maneira eficiente e estar acessível em qualquer ocasião, independente das mudanças referentes aos padrões que regem tais tecnologias.

O Banco de Dados XML eXist

Fundanda por Wolfgang Méier no final de 2000, o projeto eXist destina-se em desenvolver um banco de dados XML de código livre. O eXist possui características comuns a maioria dos bancos de dados atuais e incorpora processamento XQuery indexado, de maneira eficiente e automática. Também suporta busca via XUpdate, além de total integração com a maioria das ferramentas de desenvolvimento XML existentes atualmente. O DBMS suporta a versão 1.0 (working draft de Novembro de 2003) da linguagem XQuery, com exceção de schemas xml.

O suporte a linguagem de query XQuery no eXist possibilita escrever aplicações web inteiras com somente instruções XQuery e XSLT. Arquivos XQuery podem ser diretamente submetidos ao banco de dados, utilizando-se componentes específicos do banco de dados. Como o DBMS é feito em Java, é natural que este seja facilmente embarcado em códigos desenvolvido nesta linguagem, pode executar como servidor dedicado, através de um servlet ou diretamente embarcado na aplicação. O eXist suporta mecanismos de busca full-text, módulos XQuery e Java, extensões para http e transformação XSL. Também utiliza métodos eficientes de estruturas de indexação, as quais são baseadas em esquemas de indexação numérica para identificação de nós XML no índice. Trata o armazenamento dos documentos XML como conjunto de coleções, permitindo que estas sejam criadas dinamicamente, como define o padrão estrutural para Bancos de Dados XML. Maiores detalhes são fornecidos em **http://exist.sourceforge.net/.**

Apêndice B

O Modelo XML

Introdução

Extensible Markup Language (XML) é um protocolo de armazenamento de dados configurável e capaz de veicular todo e qualquer tipo de informação necessária num contexto de negócio. Possui uma rigorosa especificação mantida pela *World Wide Web Consortiun* (W3C) [82], onde são definidas todas as regras de criação dos documentos, bem como documentos auxiliares que impõe correção sintática e semântica.

O uso deste modelo se popularizou devido a sua flexibilidade, refino e capacidade de representação estruturada dos inúmeros conceitos de negócios existentes. Documentos XML conseguem difundir o conhecimento específico de forma inteligente, estruturada e eficaz.

Breve Histórico

Baseada na Standard Generalized Markup Language (SGML), em 1996, 80 peritos uniram forças ao W3C [82] para definir uma linguagem de marcação com o poder da SGML, porém com maior facilidade de implementação.

O surgimento da linguagem XML transformou o modo como dados são representados. A partir deste momento a XML se transformou em um padrão para descrição de dados na WEB.

Sintaxe

Conforme descrito em [82], documentos XML são compostos por elementos XML. Todo elemento XML é delimitado por *tags* (tags são elementos XML que representam uma parte da informação armazenada pelo documento).

Elementos podem ser vazios, simples ou conter outros elementos. Sua estrutura geral pode ser representada por uma árvore, pois todo documento deve ser iniciado por um elemento raiz ou *root.* Em termos práticos, cada nó da arvore representa um elemento (uma tag) e as folhas representam elementos associados a textos somente.

Documentos XML podem ser escritos em qualquer editor de texto ASCII. Atualmente, existem ferramentas próprias para edição de documentos XML, muitas delas gratuitas e disponíveis na Internet.

Um documento XML deve apresentar a extensão "*.xml*". Esta extensão serve para orientar programas de manipulação de forma a identificar como devem interpretar, verificar e validar um documento.

Documentos XML possuem regras para construção, entre elas, a declaração da versão da página XML é importante para identificar a versão atual da especificação XML definida pelo W3C [82]. Esta declaração é feita conforme o elemento: **<?xml version = "1.0"?>**

Exemplo de Trecho de Documento XML:

```
<elemento-raiz>
    <informação-um>Informação Um</informação-um>
    <informação-dois>Informação Dois</informação-dois>
                ...
</elemento-raiz>
```

Exemplo de um Atributo:

```
  <figura nomearquivo="fig1.jpg"/>
```

Características como *namespaces*, DTD's, SCHEMAS, entre outros, fazem parte do estudo que rege o uso de documentos XML. Para maiores informações veja [81,82].

XML Como Estrutura de Armazenamento

No que diz respeito a armazenamento de dados, o XML provou ser um dos melhores modelos existentes atualmente, pois é independente de tecnologia e possui uma estrutura textual. Em [81], por exemplo, o formato de armazenamento XML é utilizado num contexto de Ensino a Distância. Trabalhando em larga escala e com uma massa de dados muito grande, os documentos manipulados necessitavam de uma organização

estrutural que fornecesse tanto poder de representação semântica quanto facilidade de representação do conteúdo para os usuários.

Assim, decisões de projetos foram tomadas baseadas em modelos de representação XML, os quais impunham os requisitos necessários para armazenamento e formalização de conceitos da aplicação. Baseado nestas decisões, *templates XML* foram criados de forma a se manter os documentos de forma genérica e capaz de atender as demandas dos usuários do sistema.

Como proposto por [81], um *framework* de manipulação de documentos XML. Características de armazenamento em bancos de dados relacionais são utilizadas em um sistema *web*, permitindo um manuseio eficiente e somente instanciando ou utilizando as informações que são realmente necessárias em um dado momento da aplicação.

Em ambientes onde grandes quantidades de informação são trocadas, ao se manipular documentos XML requisitos como: flexibilidade, poder semântico, estruturação, generalidade, entre outros; tornam-se presentes, permitindo maior dinamismo, rapidez e simplicidade.

Contudo, para se conseguir manipular documentos XML de maneira adequada, alguns elementos são necessários. Conforme definido por [82], toda a especificação para documentos XML, desde sua estrutura (bem formado) até sua validação (válido), foi criada de maneira a formalizar os conceitos citados anteriormente. Portando para utilizar documentos XML, algumas regras devem ser seguidas:

- Um documento XML deve ser bem formado;
- Um documento XML deve ser válido, quando se desejar formalização na representação das informações.

Um documento bem formado é definido pela correção na estrutura interna do documento, ou seja, o XML deve conter somente uma tag raiz, todas as tags devem possuir tags de fechamento, atributos devem ser únicos dentro da tag, além de algumas outras regras simples de formatação sintática.

Documentos válidos dependem da avaliação de outros documentos como DTD's e SCHEMAS. Estes definem tipos de dados que as tags manipulam como seus atributos se

comportam e o que representa efetivamente cada elemento dentro de um documento XML.

Sem dúvida, o modelo XML tornou-se, ao longo dos anos, um meio poderoso para representação de dados e troca de informações. Sua característica simplista e padronizada permite a propagação da informação mais eficientemente, e ajuda na manipulação deste tipo de tecnologia, aumentando seu contínuo crescimento e constante melhoria. Uma explicação mais criteriosa pode ser encontrada em [82].

Apêndice C

Ferramentas de Apoio

A Ferramenta Eclipse

Eclipse é uma plataforma de desenvolvimento de código aberto, construída por uma comunidade de desenvolvedores com foco na criação de ferramentas gratuitas. Operando sob esse paradigma, e atuando com o apoio de licenças públicas, que disponibilizam código fonte livre de *royalties*, bem como direitos de livre distribuição em todo o mundo, a plataforma eclipse fornece a desenvolvedores de sistemas, flexibilidade e controle sobre seus projetos.

Eclipse é escrito em Java e é composto por inúmeros conjuntos de *plug-ins* e exemplos, os quais já foram avaliados em sistemas operacionais diversos.

1. A tecnologia **Eclipse** culmina com uma plataforma de desenvolvimento, fornecendo *frameworks* e ferramentas explicativas e extensivas. As ferramentas da plataforma são explicativas, uma vez que estas verificam a utilidade de cada *framework*, ilustrando seu uso apropriado e suportando e mantendo o desenvolvimento da plataforma em si. As ferramentas são extensivas, uma vez que suas funcionalidades são acessíveis via interfaces programáticas bem documentadas.

Para maiores informações sobre a estrutura e a missão da **Fundação Eclipse,** visite a página do projeto e leia os documentos formais que estabelecem como a fundação opera. Leia também seu *press release*, sobre novidades na criação de organizações independentes. Uma descrição da comunidade Eclipse e seus white papers documentando o projeto e uso da plataforma estão disponíveis em http://www.eclipse.org.

Eclipse.org é um website de propriedade da fundação **Eclipse**.

O Plugin Metrics

O plugin **Metrics** é responsável por calcular diversas métricas orientadas a objetos, durante ciclos de empacotamento de código e avisa, através de uma lista de tarefas, sobre violações de limites em cada métrica. Isso permite que o desenvolvedor

fique constantemente atualizado sobre a saúde do código desenvolvido. **Metrics** também permite a exportação das medições em arquivos XML, HTML para apreciação pública ou em formato CSV para maiores análises.

A Ferramenta de Modelagem ArgoUml

ArgoUML é uma ferramenta interativa e gráfica, com suporte ao desenvolvimento de modelos e documentação para softwares orientados a objetos. Desenvolvedores familiarizados com ferramentas CASE (Computer Aided Software Engineering), acharão a ferramenta de fácil utilização.

Seu público alvo são projetistas, arquitetos e desenvolvedores de software, analistas de negócio e sistemas, e outros profissionais envolvidos em análise, projeto e desenvolvimento de software. É uma ferramenta desenvolvida em Java para modelagem UML. Ela é capaz de criar e salvar a maioria dos nove padrões de diagramas UML. Também possui a habilidade de efetuar engenharia reversa de código fonte Java, sendo possível a construção dos diagramas UML para o código.

Como a ferramenta **Eclipse**, a ferramenta **ArgoUML** também é desenvolvida sob licença pública para código aberto. Ainda em estágio de experimentação, a **ArgoUML** não está pronta para produção comercial. Isto significa que muitas funcionalidades disponíveis em ferramentas comerciais não estarão implantadas na ferramenta. Por outro lado isto também significa que o desenvolvedor poderá resolver tais problemas a sua maneira, uma vez que o código é aberto à comunidade. A versão da UML suportada atualmente é a 1.3.

Apêndice D

XQuery

A Linguagem XQuery

De acordo com o *World Wide Web Consortiun* - W3C [82], XQuery é uma linguagem de consulta especialmente desenvolvida para manipulação de documentos XML. XQuery promove a facilidade na extração de dados de documentos reais e virtuais, permitindo uma melhor integração entre a Internet e bancos de dados. Esta possui o status de ser o primeiro padrão de consulta para documentos Web, desenvolvido até hoje.

Dentro dos aspectos que envolvem a linguagem, os grupos de trabalho da *XML Query*, também definiram inúmeras especificações, que são compartilhadas por diversos outros grupos de pesquisas relacionados com XML, incluindo os grupos de trabalho XSL e o XSLT. Dentre essas especificações, encontram-se seleção de documentos XML, linguagens de expressão (XPath), bem como funções e operações sobre dados XML, serialização, modelo de dados e busca dentro de documentos XML [84].

Contudo, XQuery não representa somente uma linguagem de consulta, mas também pode ser usada para diversos outros tipos de processamento de documentos XML. É uma linguagem tipada, capaz de manipular dados eficientemente. Seus tipos são os mesmo que os definidos em documentos XML, permitindo uma melhor integração entre as duas tecnologias. Sua versão atual consegue trabalhar com diferentes níveis de estruturas tipadas, podendo utilizar Schemas ou DTD´s para validação estrutural.

A linguagem foi desenvolvida para ser compacta e bem encaixada em diferentes visões de dados armazenados em XML. Tanto integração conceitual de dados, como processamentos XML são tratados pela linguagem. Em prática, consultas escritas em XQuery tendem a ser mais eficientes para os tipos de atividades as quais o formato XML é aplicado [85].

O Modelo de Dados

A linguagem XQuery é definida como um modelo de dados formal e não como um simples documento XML. Cada execução de uma consulta, feita através da linguagem, bem como seus resultados, representam uma instância deste modelo de dados.

Dentro do modelo, cada documento XML é representado como uma árvore de nós. Os tipos de nós que podem ocorrer são: documentos, elementos, atributos, texto, *namespaces*, instruções de processamento e comentários. Cada nó possui uma representação única que a distingue dos outros nós pertencentes a árvore [85].

O primeiro nó de cada documento XML é o nó *document,* o qual representa em sua totalidade o documento XML. Este nó não possui representação real, pois serve apenas para definir, conceitualmente, um arquivo XML. O restante dos nós, que formam o modelo, são os que realmente criam a representação do documento XML. Cada um dos nós é tratado da mesma forma que em documentos XML. Veja apêndice D para maiores detalhes.

A linguagem se encontra em desenvolvimento, e algumas informações aqui apresentadas estão sujeitas a modificações. Para informações recentes sobre o modelo de dados e a linguagem, visite o site http://www.w3.org/XML/Query [82].

Funcionalidades

Conforme definido pela especificação de requisitos da linguagem XQuery [84], a linguagem deve suportar operações diversas em todos os tipos de dados representados pelo modelo de dados, bem como se portar como uma linguagem completa e robusta para manipulação eficiente de documentos XML. Existem ainda diversos outros requisitos referentes a linguagem. Este apêndice possui somente característica introdutória e não tem a pretensão de ser um documento exaustivo e completo.

Referências Bibliográficas

[1] PUTNAM, L. H., Myers, W., *Industrial Strength Software*, Primeira ed. 1997, pp. 1-309.

[2] COCKBURN, A., *Agile Software Development*, Segunda Edição ed. 2002, pp. 1-278.

[3] SIMMONS, D. B., ELLIS, N. C., FULIHARA H., KUO, W, *Software Measurement - A Visualization Toolkit* New Jersey: Hewlett-Packard, 1997, pp. 3-442.

[4] GAFFNEY JUNIOR, J.E., *Metrics in Software Quality Assurance* 1981.

[5] FENTON, N., NEIL M., *Sotware Metrics: Roadmap* 2000.

[6] KAFURA , D., *A Survey of Software Metrics* 1985.

[7] FENTON N., PFLEEGER, S. L., *Software Metrics, A Rigorous and Practical Approach*, Segunda ed. PWS Publishing Company, 1997.

[8] KAN, S. H., *Metrics and Models in Software Quality Engineering*, Second Edition ed. Addison Wesley, 2002, pp. -560.

[9] AKIYAMA, F., *An Example of Software System Debugging* 1971.

[10] ROYCE, W. *Pragmatic Quality Metrics for Evolutionary Software Development Models*. 1990.

[11] MCCALL, J.A., *An Introduction to Software Quality Metrics*. 1979.

[12] PRESSMAN, R. S., *Software Engineering - A Practitioner's Approach*. Quinta Edição. 2003.

[13] International Standard Organization. *ISO/IEC 9126 - Software Engineering - Product Quality*. 15-6-2001.

[14] International Standard Organization. *ISO/IEC 12207 - Software Engineering - Life Cycle Processes*. 2002.

[15] ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C., *Qualidade de Software - Teoria e Prática*, Primeira ed. Rio de Janeiro: Prentice Hall, 2001.

[16] LORENZ, M., KIDD, J., *Object-Oriented Software Metrics* 1995.

[17] CHIDAMBER S. R., KEMERER, C. F., *Towards a Metrics Suite for Object Oriented Design* 1991.

[18] ALBRECHT, A. J., *Measuring Application Development Productivity* 1979.

[19] BASILI, V., PHILLIPS T.Y., *Evaluating and Comparing Software Metrics in the Software Engineering Lab* 1981.

[20] YU, P., SYSTÄ, T., MULLER, H., *Prediciting Fault-Proneness using OO Metrics* IEEE, 2002.

[21] Software Engineering Institute, *CMMI - Capability Maturity Model Integration* 2002.

[22] MENACHEM, M. B., GELBARD, R., *Integrated IT Management Tool Kit*, 2002.

[23] IBM. Eclipse Open Source Project. 2004.

[24] ArgoUML Design Tool. 2004.

[25] WOLFGANG, M., *eXist - Native XML Database*. 2000.

[26] Project Management Institute, *A Guide to the Project Management Book of Knowledge*, 2000 Edition ed. Newtown Square, Pennsylvania USQ: 2004.

[27] TAI, H., KOSAKA, K., *The Aglets Project* Comunications of the ACM, 1999.

[28] SUNDARESAN, N., RAJAGOPALAN, V., *Java Paradigms for Mobile Agent Facilities* 1997.

[29] AGARWAL, R., DEO, A., DAS, S., *Intelligent Agents in E-Learning* Software Engineering Notes, 2004.

[30] WONG, J. S. K., MIKLER, A. R., *Intelligent Mobile Agents in Large Distributed autonomous Cooperative Systems* The Journal of Systems and Software, 1999.

[31] HOLSAPPLE, C. W., JOSHI, K. D., *A Knowledge Management Ontology*, 2002.

[32] IEEE Computer Society. *IEEE Recommended Practice for Software Requirements Specifications*. [IEEE Std 830-1998]. 1998.

[33] MELFRY, BARROS. *Ambiente Baseaso em Agentes Autônomos para Controle de Execução de Processos*. 2003.

[34] DING, Y., MALAKA, R., KRAY, C., SCHILLO, M., *RAJA - A Resource-Adaptive Java Agent Infrastructure*. 2001.

[35] GERGENTI, F., POGGI, A., *A Development Toolkit to Realize Autonomous and Inter-operable Agents*. 2001.

[36] HOLMES, V. P., JOHNSON, W. R., MILLER, D. J., *Integrating Metadata Tools with Data services Archive to Provide Web-based Management of Large-Scale Scientific Simulation Data.* Proceedings of the 37th Annual Simulation Symposium, 2004.

[37] NIENABER, R., CLOETE, E., *Software Agent Framework for the Support of Software Project Management*. 2003. University of South Africa, Proceedings of SAICSIT.

[38] SVEIBY, K. E., *What is Knowledge Management*. 2001.

[39] MIYAZAKI, Y., MORI, K., *COCOMO Evaluation and Tailoring*. 1985. IEEE.

[40] KITCHENHAM, B., PFLEEGER, S. L., FENTON, N., *Towards a Framework for Software Measurement Validation*. 1995. IEEE.

[41] *XQuery 1.0: An XML Query Language*. 2004.

[42] BOBKOWASKA, A., *Quantitative and Qualitative Methods in Process Improvement and Product Quality Assessment*. 2004.

[43] ISHIGAKI, D., JONES, C., *Practical Measurement in the Rational Unified Process*. 2003. The Rational Edge.

[44] Project Management in the Rational Unified Process. 29-10-2001. CS2 Software Engineering note 3.

[45] JAAKSI, A., *Assessing Software Projects - Tool for Business Owners*. 1-8-2003. ESEC/FSE. Nokia.

[46] TAKEUCHI, H., *Beyond Knowledge Management: Lessons from Japan*. 1998.

[47] FIEBIG, T., HELMER, S., KANE, C. C., MOERKOTTE, G., NEUMANN, J., SCHIELE, R., WESTMANN, T., *Anatomy of a Native XML Base Management System*. 2002.

[48] SALMINEN, A., TOMPA, F. W., *Requirements for XML Document Database Systems*. 10-11-2001. Atlanta, Georgia, USA.

[49] MEGGINSON, D., *SAX: A Simple API for XML*. 2001.

[50] HORS, A.L., HÉGARET, P. L., WOOD, L., NICOL, G., ROBIE, J., CHAMPION, M., BYRNE, S., *Document Object Model (DOM) level 2 core specification*. 2000. W3C Consortium.

[51] KEMERER, C. F., *Reliability of Function Points Measurement*. 36. 1993. Communications of the ACM.

[52] KUSUMOTO, S., MORIMOTO, S., IMAGAWA, M., INOUE, K., MATSUSITA, K., TSUDA, M.. *Function Point Measurement from Java Programs*. 2002. ICSE Orlando, Florida.

[53] SHOOMAN, M., *Software Engineering* McGraw-Hill, 1983.

[54] MCCABE, T. J., WATSON, A. H., *Software Complexity*, 1994.

[55] MCCABE, T. J., A *Software Complexity Measure*. 1-12-1976. IEEE Trans. Software Engineering.

[56] MCCABE, T. J., BUTLER, C.W., *Design Complexity Measurement and Testing*. 1989. CACM.

[57] ZUSE, H., *Software Complexity: Measures and Methods*. 1990.

[58] ZUSE, H., *A Framework of Software Measurement*. 1997.

[59] ISO/IEC 14598-5 Information Technology - Software Product Evaluation. [Primeira Edição]. 1998.

[60] OGASAWARA, H., YAMADA, A., KOJO, M., *Experiences of Software Quality Management Using Metrics through the Life-Cycle*. 1996. IEEE Proceeding of ICSE-18.

[61] PATENAUDE, J. F., LANGUË, B. *Extending Software Quality Assessment Techniques to Java Systems*. 2000.

[62] ALKADI, G., ALKADI, I., *Applying a Revised RFC Metric to Redeseign an OO Design*. 2000. IEEE.

[63] SYSTÄ, T., YU, P., *Analyzing Java Software by Combining Metrics and Program Visualization*. 2000.

[64] CHAMPEAUX, D., HORNER, S., MILLER, G., *OO Process and Metrics for Effort Estimation*. 1995. Austin, TX.

[65] SCOTTO, M., SILLITTI, A., SUCCI, G., VERNAZZA, T., *A Relational Approach to Software Métrics*. 2004. ACM. SAC 2004.

[66] DUFOUR, B., DRIESEN, K., HENDREN, L., VERBRUGGE, C., *Dynamic Metrics for Java*. 2003. ACM - OOPSLA 2003.

[67] *An Analogy-Based Approach for predicting design stability of Java Classes.* 2003. Metrics 2003.

[68] IFPUG. 2004.

[69] MENS, T., DEMEYER, S., *Future Trends in Software Evolution Metrics*. 2001. ACM - IWPSE 2001.

[70] DHYANI, D., *A Survey of Web Metrics*. 2002. ACM Computing Surveys.

[71] DUNCAN, A. S., *Software Development Productivity Tools and Metrics*. 1988. Digital Equipament Corporation, Nashua NH USA.

[72] DUTTA, S., WASSENHOVE, L. N.V., KULANDAISWAMY, S., *European Software Management Practices*. 1998. Communications of the ACM.

[73] MAUI. 2005.

[74] MCGARRY. *Measurement: Key Concepts and Practices*. 7-9-2001.

[75] ISO/IEC 15939 - *Software Measurement Process*. 2004.

[76] SOLINGEN, R. V., BERGHOUT, E., *The Goal Question Metric Method - A practical Guide for Quality Improvement of Software Development*. 1999. McGraw-Hill International.

[77] EMAN, K. E., MOUKHEIBER, N., MADHAVJI, N., *An Empirical Evaluation of the G/Q/M Method*. 2002. IBM Company.

[78] FUGGETTA, A., LAVAZZA, L, MORASCA, S., *Applying GQM in an Industrial Software Factory*. 1998. ACM.

[79] PBQP. *Programa Brasileiro de Qualidade e Produtividade de Software - PBQP - Indicadores de Metas de Qualidade e Produtividade de Software*. 2001. PBQP.

[80] SCHIMIDT, A. KERTEN, M. *Bulkloading and Maintaining XML Documents.* 2002.

[81] MANIRUPA, D., LAWHEAD, P. B., *Information Storage and Management in Large Web-Based Application Using XML.* 2001.

[82] *World Wide Web Consortium - W3C.* 2005.

[83] FANKHAUSER, P., MARCHIORI, M., ROBIE, J., *XML Query (XQuery) Requirements.* 2005.

[84] MACGOVERN, J., BOTHNER, P., CAGLE, K., LINN, J., NAGARAJAN, V., *XQuery Kick Start* Sams Publishing, 2003.

[85] KATZ, H., CHAMBERLIN, D., DRAPER, D., FERNANDEZ, M., *XQuery from the Experts: A Guide to the W3C XML Query Language* Addison Wesley, 2003.

[86] ROCHA, A. R. C., *TABA* 2005.

[87] *Framework de Testes Cactus* 2005.

[88] *Plugin Metrics* 2005.

[89] COPPE/UFRJ, SOFTEX, and CESAR, *Modelo de Processo de Software Brasileiro - MPS-BR* 2005.

[90] SOFTEX, *MPS.BR - Melhoria de Processo de Software Brasileiro - Guia Geral* 2005.

[91] ROCHA, A. R. C., MALDONADO, J. C., *TABA: A Heuristic Workstation for Software Development.* 1990. Tel Aviv, Israel, Proceedings of COMPEURO 90.

[92] TRAVASSOS, G. H., *O Modelo de Integração de Ferramentas da Estação TABA.* 1994. COPPE/UFRJ - Rio de Janeiro, Tese de Doutorado.

[93] ROCHA, A. R. C., MONTONI, M., SANTO, G., *Uma Abordagem para Medição e Análise em Projetos de Desenvolvimento de Software*. 2004. Brasília, DF, III Simpósio Brasileiro de Qualidade de Software.