Universidade Federal de Pernambuco
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gabriela Moreira Carneiro Campêlo

"A utilização de métricas na Gerência de Projetos de Software Uma abordagem focada no CMM Nível 2"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: HERMANO PERRELLI DE MOURA

RECIFE, JULHO/2002

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Gabriela Moreira Carneiro Campêlo

“A utilização de métricas na Gerência de Projetos de Software Uma abordagem focada no CMM Nível 2”

ORIENTADOR
Prof. Hermano Perrelli de Moura

Recife, julho/2002

Dedico este trabalho aos meus queridos pais que sempre me apoiaram e incentivaram meus esforços.

Agradecimentos

Acima de tudo agradeço a Deus, que sempre esteve presente na minha vida como fonte de inspiração e fortaleza, sendo meu principal apoio nos momentos de dificuldade.

Agradeço aos meus pais, que sempre estiveram ao meu lado com palavras de apoio e incentivo, e ao meu noivo, Bruno, pela sua incessável paciência e compreensão com meu trabalho.

Agradeço ao Prof. Hermano Moura que como meu orientador, sempre incentivou meu trabalho e mostrou-me os melhores caminhos a serem seguidos. Agradeço também ao professor Alexandre Vasconcelos que me apoiou bastante e deu dicas importantes para o desenvolvimento do trabalho, e a todos os professores do Centro de Informática que de alguma forma contribuíram para este trabalho como professores, orientadores e exemplos de profissionais dedicados e competentes.

Por fim, agradeço ao C.E.S.A.R pela oportunidade de aplicar esse trabalho em seus projetos, mas especificamente aos gerentes dos projetos que utilizei como experimento deste trabalho, pela presteza das informações que me foram passadas e pela disponibilidade que sempre me foi oferecida. Em especial, agradeço à equipe de qualidade do C.E.S.A.R, na pessoa de Teresa Maciel e Renata Campelo, que acreditaram no meu trabalho e me ajudaram com sua experiência, oferecendo uma valorosa contribuição para conclusão deste trabalho.

Resumo

Uma das maiores dificuldades encontradas no gerenciamento de projetos de software, é saber a dimensão do que está sendo gerenciado. Inúmeras dúvidas são pertinentes aos gerentes de projeto quando se fala em dimensionamento, prazo e custo dos projetos.

Diante dos problemas encontrados no desenvolvimento de software, tais como software que não atende aos requisitos de funcionalidade e qualidade esperados pelo cliente, projetos que extrapolam prazo e custo previsto, entre outros, pesquisas têm mostrado que o elevado número de projetos fracassados são decorrentes de uma gerência de projetos ineficiente. A gestão de projetos e produtos de software somente atinge determinado nível de eficácia e exatidão se houver medidas que possibilitem gerenciar através de fatos.

Dessa forma, identificamos as métricas como uma das principais ferramentas de apoio ao gerente de projetos, pois fornecem um conjunto de informações tangíveis para planejar o projeto, realizar estimativas, gerenciar e controlar os projetos com maior precisão.

Por outro lado, um sistema efetivo de medição é recomendado por vários modelos de qualidade de software, como aspecto fundamental para subsidiar as atividades de planejamento e gerenciamento de projetos. Entre esses modelos, o *Capability Maturity Model for Software* (CMM) destaca-se como um dos modelos de qualidade de software mais adotados no mundo e recomenda fortemente o uso de métricas. Particularmente o nível 2 de maturidade do CMM, recomenda que sejam estabelecidos processos básicos de gerência de projetos de software para controlar e acompanhar custos, cronogramas e funcionalidades, todos recomendando o uso de métricas.

O trabalho apresentado nesta dissertação foca na qualidade do desenvolvimento de software, através de uma gerência de projetos eficiente, guiada pelas normas do CMM nível 2 e fazendo uso de métricas como ferramenta fundamental para uma efetiva gerência de projetos. Sem a utilização de métricas, o planejamento e acompanhamento de

projetos tornam-se atividades empíricas, realizadas com base apenas no sentimento e experiência dos gerentes de projetos.

Com este trabalho buscamos contribuir para o aperfeiçoamento do gerenciamento de projetos nas organizações que realizam desenvolvimento de software, introduzindo nas mesmas uma cultura de utilização de métricas para realização de acompanhamento dos projetos, gerenciamento da qualidade do produto gerado e construção de uma base histórica para estimativas de projetos futuros, segundo as diretrizes do CMM 2.

Abstract

One of the greatest difficulties found in the management of software projects is to know the true dimension of what is being managed. There are several concerns that are relevant to project managers when one talks of estimating the size, duration, and cost of a project.

Research has shown that, when facing the problems found in the development of software, such as applications that do not satisfy the quality and functionality requirements of the client or projects that extrapolate the estimated time and cost, the high rate of failed projects is generally due to inefficient project management. Software project and product management can reach a given level of efficacy and exactness only if certain measurements are made in order to make it possible to manage based on facts.

Therefore, we identify the choice of an adequate set of metrics as one of the main support tools for a project manager, since they supply tangible information in order to one can estimate, plan, and control their projects with increased precision.

An effective measurement system is recommended by various models of software quality assurance as a fundamental aspect that is necessary to support the activities of project planning and management. The *Capability Maturity Model for Software* (CMM), which stands out as one the most widely adopted software quality models in the World, strongly recommends the use of metrics. Specifically, the level 2 CMM recommends the establishment of basic software project management processes in order to monitor and control costs, schedules, and functionalities, each with it's own set of metrics.

This dissertation focuses on the quality of software development through an efficient project management guided by the norms of the level 2 CMM and through the use of metrics as a fundamental tool for an effective project management. The basic assumption is that, without the use of adequate metrics, the planning and monitoring of projects become empirical activities, based solely on the feeling and experience of the project manager.

With the present work we intend to contribute to the improvement of project management activities in software development organizations, introducing into such companies a work culture that incorporates the use of metrics for project monitoring, management of product quality, and the building of a historical database for use in the forecasting of future projects according to the CMM 2 directives.

Índice

Tabelas

Figuras

Capítulo 1
Introdução

A Gerência de Projetos é uma disciplina relativamente nova, mas que vem despertando interesse em profissionais de várias áreas [26]. Na Engenharia de Software, a percepção da importância do gerenciamento de projetos de software, e consequentemente seu desenvolvimento como uma disciplina desta área, vem crescendo continuamente. O aumento no interesse sobre as técnicas de gerenciamento de projeto é resultado de um crescimento das incertezas dentro do ambiente de negócio. Mudanças políticas, o avanço na tecnologia da informação, a globalização, entre outros aspectos, têm feito o universo dos negócios muito mais instável e de alto risco. E nesse ambiente de mudanças, as organizações estão buscando utilizar técnicas de gerenciamento para melhor controlar, acompanhar e garantir o sucesso de seus projetos [25].

Atualmente, com a dependência cada vez maior das organizações em relação à tecnologia da informação, a geração de produtos de software com qualidade e a um custo compensador, torna-se um fator crítico de sucesso [17].

A gestão de negócios está cada vez mais focada em garantir a satisfação do cliente através de produtos e serviços com elevados e comprovados padrões de qualidade [24]. No contexto da indústria de software, o mercado tem exigido produtos ainda mais sofisticados e em prazos de desenvolvimento mais curtos, o que tem impulsionado o número de pesquisas na área de qualidade de software, objetivando encontrar meios para garantir que o software produzido atenda às expectativas do cliente e aos atributos de qualidade definidos pela empresa fornecedora de software.

Um projeto de sucesso pode ser definido como aquele entregue no prazo, dentro do orçamento previsto e atendendo aos requisitos de funcionalidade e de qualidade acordados com o cliente [26]. Nesse contexto, é consenso na literatura que a gerência de projeto é um dos aspectos mais críticos para o sucesso dos projetos de software, pois esta é a responsável por planejar o projeto, definindo seu escopo, estimando seus custos, recursos alocados e prazo, bem como controlar o projeto e acompanhar sua execução segundo o que foi planejado.

O gerenciamento de projetos de software não é uma tarefa trivial. Controlar grandes equipes de desenvolvimento de modo que o aproveitamento das mesmas seja satisfatório exige um acompanhamento intenso, além da utilização de técnicas e modelos de métricas que quantifiquem e qualifiquem o andamento do projeto [29]. Dessa forma, as métricas de software têm se tornado uma ferramenta essencial para apoiar o gerente de projetos na captura das informações relevantes para o gerenciamento da qualidade do produto e do processo de desenvolvimento. Isto porque, as métricas fornecem um conjunto de informações tangíveis para planejar, realizar estimativas, gerenciar e controlar os projetos com maior precisão, identificando os desvios em relação ao que foi planejado.

O estudo das métricas de software evoluiu bastante nos últimos anos, devido ao grande interesse de pesquisadores que buscam soluções e padronizações para os diversos desafios e problemas existentes na área. Apesar dessa evolução, ainda é necessária a realização de muito trabalho para que a engenharia de software e, consequentemente, as métricas de software adquiram alicerce teórico suficiente para garantir a construção de um software que atenda a demanda dos usuários com a qualidade desejada e a produtividade esperada [25].

1.1 Motivação

A gerência de projetos abrange todo o processo de desenvolvimento de software. Ela inicia-se com a definição do escopo e o planejamento do projeto a ser desenvolvido, e segue com o acompanhamento do projeto, coleta de dados, avaliação das métricas e

revisões dos planos do projeto, visando a entrega do produto dentro do prazo e custo esperados e com a qualidade adequada, sendo de fundamental importância para o sucesso do projeto.

Dentro do contexto de busca pela qualidade do processo de desenvolvimento de software, várias normas e padrões específicos para software têm sido propostos. Entre eles, podemos destacar o modelo SW-CMM (*Capability Maturity Model for Software*), que tem sido bastante difundido entre as empresas de software [4]. Este modelo propõe um caminho gradual, através de níveis de maturidade da capacitação, que leva as organizações a se aprimorarem continuamente na busca da suas próprias soluções para os problemas inerentes ao desenvolvimento sistemático de software. Particularmente o nível 2 de maturidade proposto pelo CMM, chamado Repetível, sugere que sejam estabelecidos processos básicos de gerência de software para controlar e acompanhar custos, cronograma e funcionalidades.

Diante dos problemas encontrados no desenvolvimento de software, tais como software que não atende aos requisitos de funcionalidade e qualidade esperados pelo cliente, projetos que extrapolam prazo e custo previsto, entre outros, pesquisas têm mostrado que o elevado número de projetos fracassados são decorrentes de uma gerência de projetos ineficiente. Para reforçar essa constatação, Fernandes em [17] considera que a tão falada crise do software é um problema eminentemente gerencial. Uma pesquisa realizada pelo Standish Group, no ano de 2000, mostra que apenas 28% dos projetos são bem sucedidos, ou seja, são finalizados no tempo e orçamento estimados e com todas as funcionalidades acordadas implementadas. Dos demais projetos, 49% são finalizados, porém com prazos e custos ultrapassados, e 23% nem chegam a ser concluídos. Outra pesquisa realizada em 1995 revelou que apenas 9% dos projetos de software em grandes empresas são finalizados dentro do prazo e custo esperados. Esse número aumenta para 16% nas empresas de porte médio e para 28% nas empresas consideradas de pequeno porte. Esses dados mostram que à medida que os projetos aumentam sua complexidade e o tamanho

da sua equipe, aumenta a dificuldade de gerenciá-los e de acompanhar seus prazos e custos.

A gestão de projetos e produtos de software somente alcança um nível adequado de eficácia e exatidão se houver medidas que permitam acompanhar a situação do projeto. Não se consegue controlar o que não se mede, e se não medimos, não conseguimos gerenciar [16]. Apesar da importância do uso de métricas para o gerenciamento de projetos de software, a realidade nos mostra a dificuldade de implementá-las em organizações de desenvolvimento de software. Segundo Howard Rubin em [8], mais de 80% das iniciativas de se implantar métricas de software nas organizações fracassam dentro de dois anos.

O trabalho apresentado nesta dissertação busca a qualidade do desenvolvimento de software, através de uma gerência de projetos eficiente, guiada pelas normas do CMM Nível 2 e fazendo uso de métricas como ferramenta fundamental para uma efetiva gerência de projetos. As métricas aqui propostas fornecem ao gerente uma ampla visão do projeto, considerando aspectos de qualidade, progresso, utilização de esforço e recursos.

1.2 Objetivo

Neste trabalho propomos um conjunto inicial de métricas para o gerenciamento de projetos de software, baseado na literatura e em princípios e orientações decorrentes de relatos de experiências na implantação de programas de métricas. Esse conjunto de métricas proposto tem como objetivo auxiliar as atividades de gerenciamento de projetos de software, focando nas metas e objetivos das KPAs do CMM Nível 2, mais especificamente nas KPAs Gerenciamento de Requisitos, Planejamento do Projeto de Software, Supervisão e Acompanhamento do Projeto de Software e Garantia da Qualidade.

No momento da escolha das métricas não estaremos analisando as KPAs Gerência de Contrato de Software e Gerência de Configuração de Software, pois analisando as demais KPAs já estaremos também atendendo às metas destas, uma vez que a KPA Gerência de Contrato de Software combina todas as outras KPAs do Nível 2 para o controle gerencial básico do contrato e das atividades da contratada [1]. Da mesma forma, a KPA Gerência de Configuração de Software, que se preocupa em manter a integridade dos produtos do projeto de software ao longo de todo o ciclo de vida do projeto, também estará sendo contemplada de forma indireta, uma vez que suas metas estão bastante atreladas às metas das demais KPAs do Nível 2, especificamente a KPA Supervisão e Acompanhamento de Projeto de Software.

É consenso nas normas e padrões de qualidade a necessidade da definição e adoção de um processo de desenvolvimento de software disciplinado, que torne o projeto mais independente das pessoas envolvidas e possibilite a repetição de sucessos. Nesse contexto, iremos trabalhar considerando a adoção do processo RUP (*Rational Unified Process*) para o desenvolvimento de software, mais especificamente do seu fluxo de Gerenciamento de Projetos.

O RUP oferece suporte às organizações no alcance do Nível 2 de maturidade do CMM [5]. Por exemplo, o CMM utiliza as métricas para monitorar e verificar a implementação das atividades das KPAs para o alcance dos objetivos do nível de maturidade do processo [30]. Por sua vez, o RUP faz uso de métricas para acompanhamento do progresso do projeto, medidas de qualidade, maturidade, estabilidade do software, entre outras, documentando essas medidas em relatórios de avaliações. Por outro lado, as revisões do RUP propostas ao final de cada iteração, contribuem para o acompanhamento dos resultados do projeto, que é um dos objetivos da KPA Supervisão e Acompanhamento do Projeto de Software.

No entanto, o RUP não apresenta atividades sistematizadas para coleta, armazenamento e análise de métricas. Ele apenas sugere, como na atividade de Avaliação da Iteração, que tais atividades sejam realizadas [32]. Dessa forma, por termos nas métricas uma

ferramenta de extrema importância para as atividades de planejamento e gerenciamento de projetos de software, propomos modificar o fluxo de atividades de Gerenciamento de Projetos do RUP, para definir e incluir estas atividades de forma mais enfática.

Por fim, iremos apresentar um relato de experiência da aplicação das métricas aqui propostas em projetos reais, desenvolvidos por uma organização que está buscando CMM Nível 2.

1.3 Metodologia de Trabalho

Buscando a melhoria do desenvolvimento de software através de uma gerência de projeto mais efetiva, facilitada pelo uso métricas de software, seguimos as etapas abaixo para alcançar os objetivos desse trabalho.

Estudo Inicial

Primeiramente, iniciamos um estudo dos sistemas de qualidade de software, avaliando os conceitos fundamentais da qualidade, normas e modelos de avaliação do processo de desenvolvimento de software. Com esse estudo, optamos por trabalhar com o modelo CMM, por ser um modelo já maduro e bastante difundido entre as empresas de desenvolvimento de software. Neste modelo, focamos no Nível 2 de maturidade do processo, que prioriza a eficiência do gerenciamento do projeto como passo inicial e fundamental para a melhoria do processo de desenvolvimento de software, onde os sucessos obtidos em projetos anteriores são repetidos em outros projetos semelhantes e o desenvolvimento de software torna-se cada vez mais independente das pessoas, seguindo um processo bem definido.

Estudo Aprofundado

Identificamos a necessidade de priorizar a gerência de projeto para melhor planejarmos, acompanharmos e controlarmos o desenvolvimento de software. Dessa forma, iniciamos o estudo aprofundado das melhores práticas de gerência de projetos, atribuições, responsabilidades e características dos gerentes de projeto. Neste estudo, foi destacada a

utilização de métricas como ferramenta fundamental para uma gerência de projetos efetiva, sendo de essencial importância para o planejamento do projeto, bem como acompanhamento do mesmo.

Prosseguimos com um estudo detalhado das métricas de software mais utilizadas, suas características, atributos, definições e implantações de programas de métricas. Identificamos os problemas e dificuldades na implantação de programas de métricas nas organizações de software, quanto a escolha das métricas a serem coletadas e a efetiva utilização das mesmas para a gerência de projetos.

Seleção e Especificação das Métricas

Com base no estudo aprofundado, selecionamos e especificamos um conjunto inicial de métricas a serem coletadas para auxiliar no planejamento e acompanhamento de projetos de desenvolvimento de software, segundo os objetivos das KPAs do CMM Nível 2. Identificamos a necessidade de se ter um processo de desenvolvimento de software com atividades bem definidas para guiar o planejamento e acompanhamento do projeto de software, bem como a coleta e avaliação de métricas. Dessa forma, adaptamos o Fluxo de Gerenciamento de Projetos do RUP para contemplar a coleta e interpretação das métricas de projeto de software.

Aplicação em Projetos Reais

Para avaliar a efetividade das métricas selecionadas e sua contribuição para o gerente de projeto e para a organização que busca o CMM Nível 2, aplicamos essas métricas durante o desenvolvimento de quatro projetos de software. Os resultados serão apresentados ao longo desta dissertação.

Documentação do Trabalho

Por fim, analisamos e comparamos os resultados encontrados, os problemas e conclusões obtidas a partir da aplicação prática das métricas aqui propostas.

1.4 Organização

Além deste capítulo introdutório, a dissertação está organizada da forma descrita a seguir: O Capítulo 2 apresenta uma introdução ao modelo CMM, detalhando as atividades e metas das suas KPAs do Nível 2 que abordam o planejamento e gerenciamento de projetos. Também é apresentado o processo de desenvolvimento de software RUP, mais especificamente seu fluxo de atividades de Gerenciamento de Projetos, ressaltando como esse processo oferece suporte às organizações no alcance do Nível 2 do CMM.

O Capítulo 3 apresenta o conceito de métricas, seus atributos e sua importância para o planejamento e acompanhamento de projetos. Neste capítulo também são apresentadas algumas orientações para a escolha e especificação do conjunto de métricas a ser adotado em uma organização, bem como para a preparação e envolvimento dos membros da organização no programa de métricas a ser implantado.

No Capítulo 4 propomos um conjunto inicial de métricas com base nas metas das KPAs do CMM 2, apresentando o processo utilizado para a escolha das métricas, detalhando os atributos que caracterizam tais métricas e os objetivos a serem alcançados com as métricas escolhidas. Também apresentamos as modificações propostas para fluxo de atividades de Gerenciamento de Projetos do RUP, para melhor suportar um programa de implantação de métricas nas organizações que adotam este processo.

No Capítulo 5 apresentamos o relato da aplicação das métricas propostas em projetos reais, os resultados alcançados com a utilização das métricas, focando nos ganhos obtidos e nos problemas encontrados.

Por fim, no Capítulo 6 apresentamos as conclusões do trabalho, as contribuições trazidas por este, trabalhos relacionados e extensões e melhorias para trabalhos futuros.

Capítulo 2

Capability Maturity Model e Rational Unified Process

Neste capítulo iremos apresenta o Capability Maturity Model (CMM), modelo de qualidade escolhido para direcionar o trabalho desta dissertação, bem como o Processo Unificado de Desenvolvimento de Software da Rational (RUP), que será adaptado neste trabalho para contemplar atividades de coleta e avaliação de métricas para gerenciamento de projetos de software.

2.1 CMM – Capability Maturity Model

Visando a melhoria do desenvolvimento de software, vários modelos para avaliação do processo de produção de software têm sido propostos por instituições no mundo inteiro. Dentre os mais utilizados, podemos citar o Capability Maturity Model (CMM), do Software Engineering Institute (SEI), o qual tem sido bastante utilizado pelas empresas de software [4]. Entre estas, podemos citar EDS Wireless Resource Center, Motorola, Boeing, IBM, Bellcore, entre outras [34].

O CMM fornece às organizações diretrizes para controlar seus processos de desenvolvimento de software, de modo a desenvolver e manter software de melhor qualidade, bem como instituir uma cultura de excelência em engenharia e gerenciamento de projetos de software. O CMM propõe um caminho gradual, através de níveis de maturidade da capacitação, que leva as organizações a se aprimorarem continuamente na busca das suas próprias soluções para os problemas inerentes ao desenvolvimento sistemático de software. A capacitação aqui mencionada refere-se a habilitação que a organização tem em sistematicamente produzir software com a qualidade esperada, dentro dos prazos acordados e com os recursos estimados [1] .

A estrutura do CMM consiste de cinco (1 a 5) níveis de maturidade, conforme ilustrado na Figura 2.1.

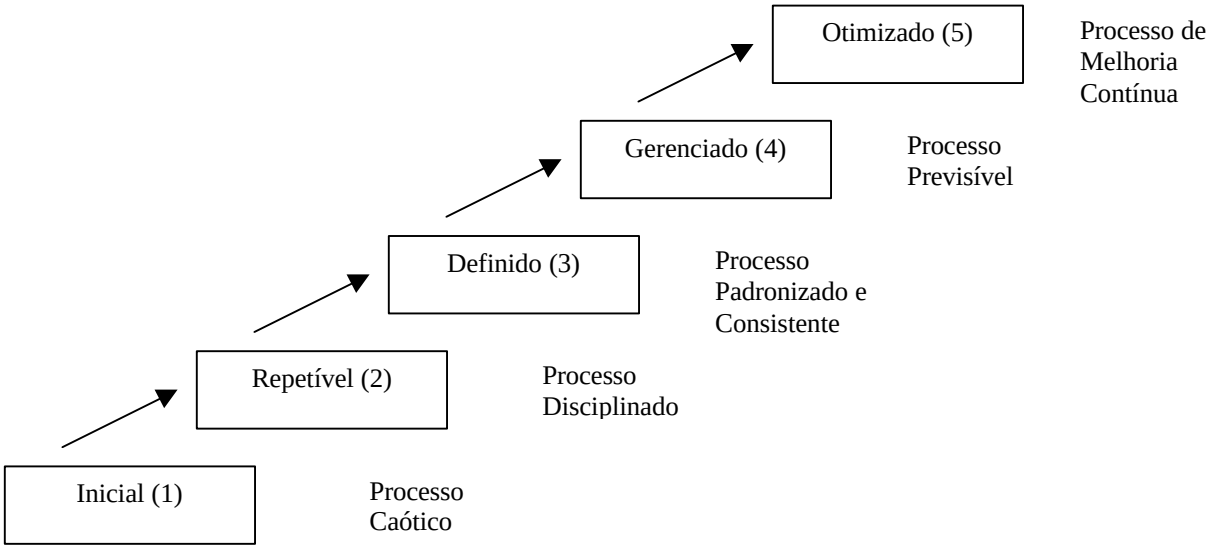

Figura 2.1 Os cinco níveis de maturidade do CMM

Nível 1: Inicial. O processo de desenvolvimento de software é caracterizado como *ad hoc*, podendo facilmente chegar ao caos. Poucos processos estão definidos e o sucesso do projeto depende do esforço individual de cada um envolvido.

Nível 2: Repetível. Processos básicos para gerenciamento de software são estabelecidos para controlar e acompanhar custos, cronograma e funcionalidades. Neste nível, o processo é caracterizado como disciplinado, estando sob o controle efetivo de políticas de gerenciamento de projetos, seguindo planos realistas, baseado em desempenho de projetos similares já realizados.

Nível 3: Definido. Os processos de gerenciamento e das atividades de engenharia de software estão documentados e padronizados, integrando o padrão de processo de software da organização. Todos os projetos utilizam esses processos padrões.

Nível 4: Gerenciado. Medidas detalhadas do processo de software e da qualidade dos produtos são colhidas. Tanto o processo de software quanto o produto são quantitativamente entendidos e controlados.
Nível 5: Otimizado. Melhorias contínuas no processo são realizadas baseadas nos *feedbacks* quantitativos dos processo e produtos.

Cada nível de maturidade indica o nível de capacidade do processo de desenvolvimento de software da organização. Por exemplo, no Nível 2 a capacidade do processo da organização foi elevada de *ad hoc* para disciplinada por terem sido estabelecidos controles para o gerenciamento do projeto [13].

Com exceção do Nível 1, cada nível de maturidade é decomposto em áreas-chave de processo, ou KPA (K*ey Process Area*), que conduzem ao alcance de metas de melhoria do processo, indicando os pontos que as organizações devem focar para melhorar seu processo de software. As áreas-chave estão organizadas em cinco seções, chamadas de características comuns, que por sua vez, contêm práticas-chave. As características comuns determinam as características de institucionalização ou de implementação das práticas-chave, enquanto que as práticas-chave descrevem as atividades e/ou infra-estrutura necessárias para satisfazer as metas de uma KPA. A estrutura do CMM é ilustrada na Figura 2.2.

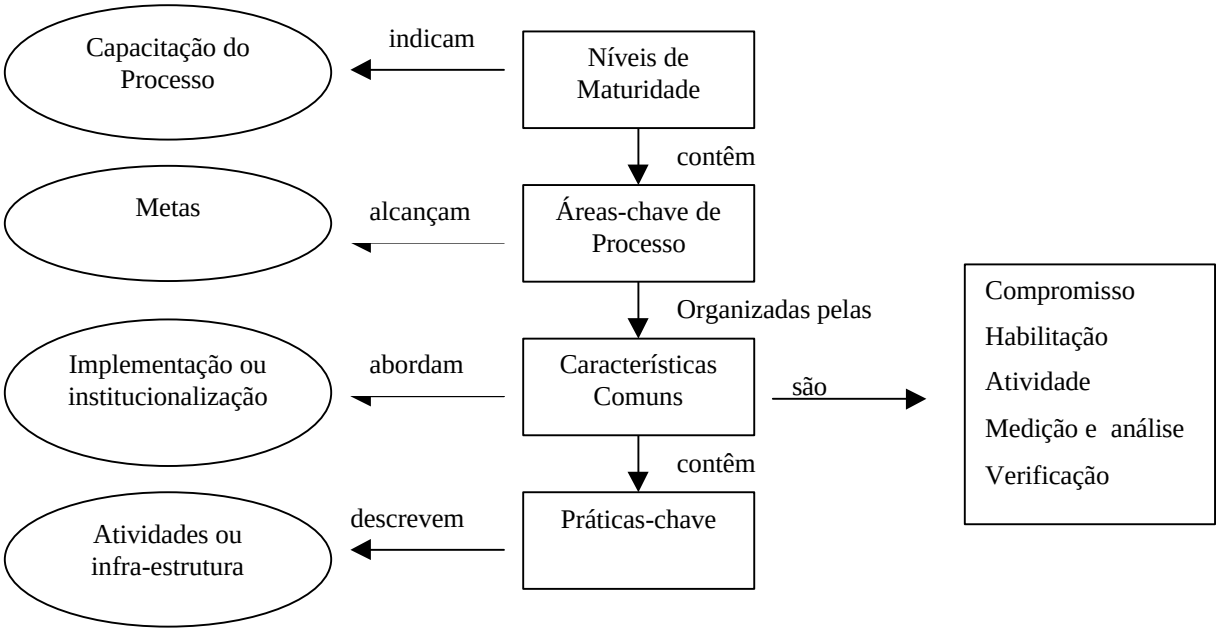

Figura 2.2 A estrutura do CMM

Cada KPA identifica um conjunto de atividades relacionadas que, quando realizadas corretamente, alcançam os objetivos considerados importantes para acentuar a capacidade do processo. O caminho para alcançar os objetivos de uma KPA difere de acordo com os projetos de diferentes ambientes, domínios e aplicações. No entanto, todos os objetivos de uma KPA devem ser alcançados para a organização poder satisfazer a dada KPA.

Santander [14] apresenta uma opinião bastante otimista com relação ao grau de maturidade dos processos de software das empresas. Segundo Santander, apesar de grande parte das organizações envolvidas com desenvolvimento de software possuírem processos imaturos, os quais são enquadrados no nível Inicial (1) do CMM, existe a preocupação por parte destas empresas em melhorar e disciplinar seus processos de software, objetivando atingir os requisitos mínimos exigidos para alcançar o nível Repetível (2) do CMM.

Além disso, Fiorino [1] ressalta a dificuldade de uma empresa sair do nível Inicial (1), que é a realidade da grande maioria das empresas, e iniciar um processo de mudança cultural com ênfase em processos e foco gerencial para atingir o nível Repetível (2). Nesse contexto, ele acredita que o maior ganho para as empresas está na adoção dos níveis de maturidade 2 e 3, tendo em vista o estado inicial que se encontram.

Por outro lado, como apresenta Silva, as KPAs abordadas nesses dois níveis de maturidade satisfazem a maioria das cláusulas da norma ISO 9001 aplicada a software [15]. Em linhas gerais, esta norma recomenda um controle efetivo do planejamento e desenvolvimento do projeto de software, o que segue as mesmas diretrizes das KPAs do Nível 2 do CMM, pois também focam no planejamento e acompanhamento do projeto de software.

Pelos motivos citados acima, escolhemos direcionar esse trabalho seguindo as diretrizes do CMM 2, que focam na parte do projeto de software relacionada a estabelecer uma base de controle para o gerenciamento de projeto.

2.1.1 Nível 2 – Repetível

Neste nível de maturidade, espera-se que a capacitação do processo seja melhorada a cada novo projeto, através do estabelecimento de processos de gerência que guiem os projetos de software. Os requisitos do cliente e os produtos resultantes do processo são controlados, e práticas para o gerenciamento do projeto são estabelecidas. O processo de software em empresas no Nível 2 pode ser visualizado como uma seqüência de caixas pretas, existindo pontos de controle entre elas para acompanhamento do progresso do projeto, que são denominados marcos de referência. Apesar do gerenciamento não acompanhar os detalhes das atividades realizadas internamente nessas caixas pretas, os produtos gerados durante o processo e os marcos de referência para verificação do andamento do processo podem ser identificados e acompanhados durante o gerenciamento, possibilitando que ações sejam tomadas mediante a identificação de problemas no decorrer do processo.

A seguir, apresentaremos as áreas-chave de processo do Nível 2, as metas base para cada área-chave e as atividades a serem realizadas para o alcance dessas metas.

GR – Gerenciamento de Requisitos

O objetivo desta KPA é entender as necessidades e expectativas do cliente, garantindo um entendimento comum entre as partes, determinando os requisitos que devem ser contemplados no projeto do software e o que se espera receber como resultado do projeto. O gerenciamento de requisitos envolve estabelecer e manter um acordo com o cliente sobre os requisitos que o software deverá atender. O acordo trata tanto dos requisitos técnicos do sistema, como dos requisitos não técnicos, isto é, requisitos que descrevem acordos, condições ou termos contratuais que podem afetar e determinar as atividades de gerência de um projeto de software.

Uma vez firmado este acordo, ele servirá como base para as estimativas, planejamento e acompanhamento das atividades do projeto de software durante todo seu ciclo de desenvolvimento. Sempre que houver mudança nos requisitos acordados, o plano do projeto, cronograma, atividades e os produtos gerados deverão ser ajustados para garantir a consistência com os requisitos alterados.

As metas desta KPA são:

Meta 1	Documentar e controlar os requisitos alocados para estabelecer uma base para todo o processo de desenvolvimento.
Meta 2	Manter planos, artefatos e atividades de software consistentes com os requisitos alocados.

As atividades a realizar para alcançar estas metas são apresentadas a seguir:

Atividade 1	Revisar requisitos alocados antes de incorporá-los ao projeto de software.

Atividade 2 Utilizar requisitos alocados como base para o desenvolvimento de software

Atividade 3 Revisar alterações nos requisitos alocados antes de incorporá-los ao projeto de software.

PPS – Planejamento de Projeto de Software

A KPA Planejamento de Projeto de Software tem o propósito de estabelecer planos de projeto consistentes e realísticos para realizar as atividades de engenharia de software e para gerenciar o projeto de software.

O planejamento de projeto de software envolve a realização de estimativas para o trabalho a ser realizado (incluindo estimativas do tamanho do software a ser produzido, recursos necessários e cronograma), identificação de riscos, e definição do plano para realização do trabalho especificado. A primeira atividade do planejamento de projeto deve ser a identificação do escopo e das restrições do software a ser construído, que são estabelecidos através das práticas de gerenciamento de requisitos.

Como metas desta KPA temos:

Meta 1 Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do projeto.

Meta 2 Planejar e documentar as atividades e os compromissos do projeto de desenvolvimento de software.

Meta 3 Obter a concordância dos grupos e das pessoas envolvidas quanto aos respectivos compromissos relacionados ao projeto de desenvolvimento de software.

As atividades a realizar para alcançar essas metas são:

Atividade 1 O grupo de engenharia de software deve participar da proposição do projeto.

Atividade 2	O planejamento do projeto de software deve ser iniciado nos primeiros estágios do projeto, ou seja, logo durante a elaboração da proposta técnica de desenvolvimento.
Atividade 3	O grupo de engenharia de software, juntamente com os outros grupos envolvidos, devem participar do planejamento global do projeto ao longo da duração deste.
Atividade 4	Os compromissos externos do projeto devem ser revisados junto à gerência sênior, de acordo com um procedimento documentado.
Atividade 5	Um modelo de ciclo de vida de software deve ser definido.
Atividade 6	O plano de desenvolvimento de software do projeto deve ser elaborado de acordo com um procedimento documentado.
Atividade 7	O plano de desenvolvimento de software deve ser documentado.
Atividade 8	Os artefatos de software que precisam ser submetidos à gerência de configuração de forma a garantir o controle do projeto devem ser identificados.
Atividade 9	Os tamanhos dos artefatos devem ser estimados.
Atividade 10	O esforço e o custo do projeto de software devem ser estimados.
Atividade 11	Os recursos computacionais críticos devem ser estimados de acordo com um procedimento documentado.
Atividade 12	O cronograma de software do projeto deve ser estabelecido.
Atividade 13	Os riscos do projeto de software associados com os custos, recursos, cronograma e aspectos técnicos devem ser identificados, avaliados e documentados.
Atividade 14	Planos para as ferramentas de suporte e facilidades de engenharia de software devem ser elaborados.
Atividade 15	Os dados de planejamento devem ser registrados, gerenciados e documentados.

SAPS – Supervisão e Acompanhamento de Projeto de Software

Esta KPA tem o propósito de estabelecer uma visibilidade adequada do progresso do projeto, de forma que ações efetivas possam ser tomadas em tempo quando o projeto de software sofrer desvios significativos do seu planejamento.

A SAPS envolve acompanhar e revisar a execução e os resultados do desenvolvimento do projeto, confrontando-os com as estimativas documentadas, os compromissos e os planos do projeto, bem como ajustar esses planos com base na execução e nos resultados reais do projeto.

O plano de projeto de software é usado como base para o acompanhamento das atividades do projeto, devendo contemplar os compromissos acordados, obedecendo aos recursos disponíveis, às restrições e às capacidades do projeto de software.

As metas básicas para satisfazer essa KPA são:

Meta 1	Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software.
Meta 2	Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software.
Meta 3	Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e as pessoas envolvidas.

As atividades a realizar para alcançar as metas supracitadas são:

Atividade 1	O plano de desenvolvimento de software deve ser utilizado para acompanhar as atividades de software e comunicar o estado de progresso do projeto.
Atividade 2	O plano de desenvolvimento de software deve ser revisado de acordo com procedimentos documentados.

Atividade 3	Os compromissos e alterações de compromissos devem ser revisados junto a gerência sênior, de acordo com procedimentos documentados.
Atividade 4	As alterações nos compromissos que afetam o projeto e que tiverem sido aprovadas devem ser comunicadas ao grupo de engenharia de software e aos outros grupos envolvidos.
Atividade 5	O tamanho dos artefatos de software e as alterações dos mesmos devem ser acompanhados.
Atividade 6	Esforço e custo do software devem ser acompanhados e ações corretivas devem ser tomadas quando necessário
Atividade 7	A utilização dos recursos computacionais críticos deve ser acompanhada e ações corretivas devem ser tomadas quando necessário.
Atividade 8	O cronograma de software deve ser acompanhado e ações corretivas devem ser tomadas quando necessário.
Atividade 9	As atividades técnicas de engenharia de software devem ser acompanhadas e ações corretivas devem ser tomadas quando necessário.
Atividade 10	Os riscos de software relativos a custos, recursos, cronograma e aspectos técnicos do projeto devem ser acompanhados.
Atividade 11	Os dados de medições reais e de replanejamento devem ser registrados.
Atividade 12	O andamento do projeto deve ser acompanhado.
Atividade 13	Revisões formais devem ser conduzidas nos marcos de acompanhamento do projeto.

GC – Gerência de Contrato de Software

O principal objetivo da KPA Gerência de Contrato de Software é selecionar e contratar empresa de software, quando houver necessidade de terceirizar serviços específicos do

projeto, garantindo a qualificação adequada da empresa contratada e gerenciando seu trabalho efetivamente.

Na contratação, é estabelecido um acordo documentado cobrindo os requisitos técnicos e não técnicos, o qual deve ser utilizado como base para gerenciar o contrato. O trabalho a ser realizado pela contratada, os planos para o trabalho e os padrões a serem seguidos devem ser documentados. As atividades de planejamento, acompanhamento e supervisão do desenvolvimento do software contratado devem ser realizadas pela própria contratada. O papel da contratante é garantir que essas atividades sejam executadas apropriadamente e que os produtos de software entregues pela contratada, satisfaçam aos critérios de aceitação da contratante.

As metas básicas desta KPA são:

Meta 1	Contratar empresa de software qualificada.
Meta 2	Estabelecer acordo entre a contratante e a contratada com a reciprocidade de seus compromissos.
Meta 3	Manter uma comunicação efetiva entre a contratante e a contratada ao longo de todo o contrato.
Meta 4	A contratante deve acompanhar o desempenho e resultados reais da contratada, comparando-os com os compromissos assumidos.

A seguir apresentaremos as atividades a realizar para alcançar as metas supracitadas.

Atividade 1	O trabalho a ser contratado deve ser planejado e definido de acordo com um procedimento documentado.
Atividade 2	A contratante deve selecionar a contratada de acordo com um procedimento documentado.
Atividade 3	O contrato de software deve ser gerenciado com base no acordo estabelecido entre as partes.
Atividade 4	A contratante deve revisar e aprovar o plano documentado de desenvolvimento de software apresentado pela contratada.

Atividade 5	O plano de desenvolvimento de software da contratada deve servir como base para o acompanhamento do seu trabalho.
Atividade 6	Alterações no trabalho da contratada, compromissos, termos e condições contratuais de acordo devem seguir um procedimento documentado, envolvendo tanto a contratada quanto a contratante.
Atividade 7	A gerência da contratante deve conduzir revisões periódicas de coordenação e de avaliação do estado de progresso junto à gerência da contratada.
Atividade 8	O cronograma de software deve ser acompanhado e ações corretivas devem ser tomadas quando necessário.
Atividade 9	A contratante deve realizar revisões formais nos marcos de acompanhamento de progresso para avaliar os resultados e as atividades de engenharia de software concluídas pela contratada.
Atividade 10	O grupo da garantia da qualidade da contratante deve monitorar as atividades de garantia da qualidade de software da contratada de acordo com um procedimento documentado.
Atividade 11	O grupo de gerência de configuração de software da contratante deve monitorar as atividades de gerência de configuração de software da contratada, de acordo com procedimentos documentados.
Atividade 12	Testes de aceitação do produto entregue pela contratada devem ser realizados de acordo com um procedimento documentado.
Atividade 13	A contratante deve avaliar periodicamente o desempenho da contratada.

GQS – Garantia de Qualidade de Software

O propósito desta KPA é fornecer à gerência uma visibilidade apropriada do processo utilizado para o desenvolvimento do projeto de software e dos produtos gerados pelo projeto.

A GQS envolve rever e auditar os artefatos e atividades de software, de maneira a verificar se estão de acordo com os procedimentos e padrões aplicáveis, bem como prover os resultados dessas auditorias e revisões do projeto de software aos gerentes envolvidos no projeto.

O grupo de GQS trabalha no projeto de software desde seu início estabelecendo planos, padrões e procedimentos que irão adicionar valor ao projeto de software, e que satisfaçam às restrições do projeto e às políticas da organização.

As metas desta KPA são:

Meta 1	Planejar atividades de GQS
Meta 2	Verificar objetivamente a conformidade das atividades e dos artefatos de software com os padrões, procedimentos e requisitos aplicáveis.
Meta 3	Informar aos grupos e às pessoas envolvidas quanto às atividades e resultados de GQS.
Meta 4	Encaminhar à gerência todas as questões de não-conformidade que não possam ser resolvidas no âmbito do projeto de software.

Para o alcance dessas metas, temos as seguintes atividades a realizar.

Atividade 1	Um plano de GQS deve ser preparado para o projeto de software, de acordo com procedimentos documentados.
Atividade 2	O grupo de GQS deve executar suas atividades de acordo com o plano de GQS.
Atividade 3	O grupo de GQS deve participar na preparação e revisão dos planos, padrões e procedimentos.
Atividade 4	O grupo de GQS deve revisar as atividades de engenharia de software para verificar o seu cumprimento.

Atividade 5 O grupo de GQS deve executar auditorias nos artefatos designados, verificando sua conformidade com os padrões e procedimentos e requisitos estabelecidos.

Atividade 6 O grupo de GQS deve reportar periodicamente os resultados de suas atividades para o grupo de engenharia de software.

Atividade 7 Os desvios identificados nas atividades e artefatos de software devem ser documentados e tratados de acordo com um procedimento documentado.

Atividade 8 O grupo de GQS deve conduzir revisões periódicas de suas atividades e dos seus laudos das revisões e auditorias realizadas com o grupo de GQS do cliente, quando existir.

GCS – Gerência de Configuração de Software

A finalidade da GCS é estabelecer e manter a integridade dos produtos do projeto de software ao longo do ciclo de vida de software. Os artefatos que estão sob a gerência de configuração são aqueles que necessitam ter seu histórico armazenado e são denominados itens de configuração.

As metas básicas que precisam ser alcançadas para satisfazer a GCS são:

Meta 1 Planejar as atividades de gerência de configuração de software.

Meta 2 Identificar, controlar e tornar disponíveis os artefatos de software selecionados.

Meta 3 Controlar as alterações nos artefatos de software identificados.

Meta 4 Informar pessoas e grupos envolvidos acerca do estado e do conteúdo das *baselines* de software.

As atividades a realizar para alcançar as metas supracitadas são:

Atividade 1 Um plano de GCS deve ser preparado para cada projeto, de acordo com um procedimento documentado.

Atividade 2 O plano de GCS documentado e aprovado deve ser usado como base para a execução das atividades de GCS.

Atividade 3 Um repositório para *baselines* deve ser estabelecido.

As atividades relacionadas com a gerência de configuração são de cunho técnico e operacional, podendo ser realizadas por engenheiros de software e/ou analistas, desde que designados pelo gerente do projeto e devidamente treinados para assumirem o papel de gerente de configuração. Todo o planejamento das atividades de GCS, itens de configuração e *baselines* devem ser definidos em um plano de gerência de configuração.

É importante salientar que, para o alcance das metas definidas para cada KPA do CMM Nível 2, além de realizar as atividades descritas acima, é preciso que sejam definidas e utilizadas medições que determinam o estado das atividades de cada KPA, bem como realizar periodicamente revisões e auditorias nas atividades e artefatos das KPAs, com a participação da gerência sênior, gerente de projeto e grupo de qualidade.

2.2 RUP – The Rational Unified Process

O RUP é um processo de engenharia de software definido pela empresa *Rational* [5] que cobre todo o ciclo de desenvolvimento de software e provê uma abordagem disciplinada para atribuir tarefas e responsabilidades durante o desenvolvimento de software dentro de uma organização. Seu principal objetivo é assegurar a produção de software de alta qualidade, que atenda às expectativas do seu usuário final dentro do cronograma e do orçamento previsto.

O RUP possui um *framework* de processo que pode ser adaptado e estendido para melhor atender às necessidades da organização. Como características mais marcantes desse processo, podemos citar:

- Processo iterativo: Essa abordagem iterativa propicia um entendimento gradativo do problema, através de sucessivos refinamentos e incremento da solução efetiva durante os múltiplos ciclos. Além disso, permite uma maior flexibilidade para acomodar novos requisitos ou solicitações de mudanças dos objetivos do projeto,

bem como identificar e resolver riscos do projeto mais cedo, contribuindo para o gerenciamento mais efetivo do projeto.

- Centrado na arquitetura: O processo foca no desenvolvimento da arquitetura de software o quanto antes. Pois tendo uma arquitetura robusta definida já nas fases iniciais do projeto, facilita o desenvolvimento paralelo, minimiza retrabalho, e aumenta a probabilidade do reuso de componentes.
- Dirigido a caso de uso[1]: O RUP enfatiza o desenvolvimento de sistemas baseado no entendimento de como o sistema será utilizado. O processo segue um fluxo de ações para realização dos casos de uso. Assim, os casos de uso são especificados, projetados, implementados e utilizados como fonte para definição dos casos de teste do sistema. Eles dirigem todo o processo de desenvolvimento do software.

O RUP oferece um método disciplinado para a distribuição de tarefas e responsabilidades durante o desenvolvimento do software, explicitando o quê, quando e por quem cada tarefa deve ser executada.

Este processo vem sendo muito bem aceito nas organizações de desenvolvimento de software, por possuir características que o tornam um processo prático. Por exemplo, ele é um processo customizável e pode ser adaptado à forma de trabalho das organizações, é disponibilizado em páginas da Web permitindo fácil acesso aos membros da organização, além de prover *guidelines*, modelos e exemplos que facilitam sua utilização.

O RUP utiliza UML – *Unified Modeling Language* - como linguagem padrão para elaboração da modelagem de software orientado a objetos. Sua estrutura de representação utiliza quatro elementos básicos:
Trabalhadores (*Worker*): O termo trabalhador se refere aos papéis que definem como o indivíduo deve realizar seu trabalho.

[1] Sequência de ações que incorpora uma determinada funcionalidade ao sistema, fornecendo a quem iniciou sua execução um resultado que pode ser quantificado.
.

Atividades: Representa uma unidade de trabalho com um propósito claro, que deve ser realizada por um trabalhador. Toda atividade deve estar associada a um trabalhador e possuir uma série de passos que descrevem o que deve ser feito para sua realização.
Artefatos: São os produtos tangíveis de um projeto que são criados, modificados e utilizados durante o processo. Todo artefato deve estar sob a responsabilidade de um trabalhador.
Fluxos (*Workflow*): É uma seqüência de atividades que produz um resultado de valor para o processo. No RUP, encontramos os seguintes fluxos:

- Modelagem de Negócio
- Requisitos
- Análise e Projeto
- Implementação
- Testes
- Implantação
- Configuração e Gerência de Mudanças
- Gerenciamento de Projetos
- Ambiente

Além disso, o RUP trabalha com o conceito de fases de desenvolvimento, onde cada fase tem um objetivo bem definido, e podem possuir uma ou mais iterações. As quatro fases do RUP são:

- Concepção: Tem como objetivo estabelecer o escopo e viabilidade econômica do projeto.
- Elaboração: Tem como objetivo eliminar os principais riscos do projeto e definir uma arquitetura de desenvolvimento estável.
- Construção: Tem como objetivo desenvolver o produto até que ele esteja pronto para beta testes.
- Transição: Tem como objetivo entrar no ambiente do usuário para implantação do sistema.

Kruchten [18] mostra que o RUP captura muitas das melhores práticas definidas para o desenvolvimento de software. Destacamos algumas dessas práticas mapeando-as a características do RUP.

Desenvolvimento iterativo de software

Como já citado, o RUP é um processo iterativo, e na sua definição, aborda as iterações do projeto de forma bastante controlada. As iterações são planejadas, considerando o número de iterações para o projeto, a duração de cada iteração e seus objetivos. As tarefas e responsabilidades de cada papel envolvido são definidas e medidas objetivas do progresso de cada iteração são capturadas e avaliadas. Ao final de cada iteração, os trabalhos são avaliados e trabalhos que fiquem para próximas iterações, ou precisem ser refeitos, são controlados.

Gerenciamento de Requisitos

O gerenciamento de requisitos é uma abordagem sistemática para levantar, organizar, comunicar e gerenciar mudanças nos requisitos de um software.

O RUP possui o fluxo de Requisitos que tem como um dos objetivos definir as funcionalidades do sistema, em comum acordo entre o desenvolvedor e o usuário, identificando e documentando os requisitos do sistema.

Além disso, através do seu fluxo de Configuração e Gerência de Mudanças, o RUP provê uma maneira de acompanhar a evolução do produto de software, capturando e gerenciando as requisições de mudanças e implementando-as de maneira consistente com os artefatos já produzidos.

Uso de Componentes e Centrado na Arquitetura

O RUP suporta o desenvolvimento baseado em componentes de diversas formas:

- A abordagem iterativa permite que o desenvolvedor progressivamente identifique componentes e decida que componentes irá desenvolver, reusar ou comprar.

- Conceitos como pacotes, subsistemas e camadas são usados durante a análise e o projeto do sistema para organizar os componentes e especificar suas interfaces.
- A arquitetura permite a enumeração dos componentes e da forma como estes se integram, bem como dos mecanismos e padrões (*patterns*) pelos quais eles interagem.

O RUP é um processo com foco na arquitetura, conforme citado anteriormente. Já nas iterações iniciais da fase de Elaboração do projeto, ele propõe que a arquitetura do software seja definida e validada, eliminado a maioria dos riscos técnicos do projeto.

Ele oferece *templates* para descrever a arquitetura nas suas diversas visões e possui atividades específicas que têm como objetivo identificar as restrições e os elementos significativos da arquitetura.

Verificação da qualidade do Software

No RUP não existe um trabalhador (*worker*) especificamente encarregado da verificação da qualidade no projeto, pois a qualidade é tratada como responsabilidade de todos os desenvolvedores da organização.

Através do seu fluxo de Testes e de outros processos e produtos de medições, como os encontrados no fluxo de Gerenciamento de Projetos, o RUP foca na validação dos produtos, verificando objetivamente se estes atingiram ou não às expectativas de qualidade definidas no projeto. Além disso, ao final de cada iteração o RUP propõe a realização da avaliação da iteração com o objetivo de verificar os resultados alcançados com relação aos critérios de avaliação estabelecidos no plano da iteração, analisar o grau de sucesso alcançado e identificar os problemas que precisam ser corrigidos nas próximas iterações.

2.3 Relacionando o RUP com o Nível 2 do CMM

A partir do que foi apresentado nas seções anteriores sobre o CMM e o RUP, podemos mapear de que forma cada KPA do CMM 2 pode ser satisfeita por alguma característica e/ou princípio que fundamenta o processo RUP.

A Tabela 2.1 apresenta um resumo das metas de cada KPA do Nível 2, relacionando-as com as características do RUP que satisfazem essas metas e consequentemente satisfazem as suas respectivas KPAs [14].

KPAs	Metas	Processo Unificado
Gerenciamento de Requisitos	(Meta 1) Documentar e controlar os requisitos alocados para estabelecer uma *baseline* para uso gerencial e da Engenharia de Software; (Meta 2) Manter planos, artefatos e atividades de software consistentes com os requisitos alocados.	(Rel – Meta 1) O Modelo de Casos de Uso é o documento base para o processo, onde são especificados os requisitos do sistema na forma de casos de uso. O controle dos requisitos (casos de uso) é realizado nas m*ilestones* durante as fases bem como nas iterações do processo. A documentação dos requisitos é gradual nos *workflows* do processo. Os aspectos tais como rastreamento de requisitos e gerenciamento de mudanças são tratados nos fluxo de Configuração e Gerência de Mudanças, onde o RUP propicia o gerenciamento e rastreamento dos requisitos, bem como o controle de configuração da evolução dos artefatos. (Rel –Meta 2) O processo orientado a casos de uso possibilita manter uma maior consistência e acompanhamento dos requisitos. Artefatos e atividades são realizados tendo como base os casos de uso, que orienta o desenvolvimento do sistema até a fase de testes.

Tabela 2.1 Relacionamento do RUP com as KPAs CMM nível 2

KPAs	Metas	Processo Unificado
Planejamento de Projeto de Software	(Meta 1) Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do Projeto de Software. (Meta 2) Planejar e documentar as atividades e os compromissos do projeto de desenvolvimento de software. (Meta 3) Obter concordância das pessoas envolvidas quanto a compromissos no projeto de desenvolvimento de software.	(Rel - Meta 1) Elaboração de Plano de Projeto de Software e Planos de Iteração, contendo estimativas de esforço, e recurso para o desenvolvimento de software. (Rel - Meta 2) Plano de Projeto de Software, Planos de Iteração, Lista de Riscos, Avaliação de Iterações e Fases. (Rel - Meta 3) Gerentes de projeto e demais profissionais envolvidos participam do planejamento bem como do desenvolvimento iterativo e incremental, no qual os compromissos de projeto são revisados e validados com o cliente ao final de cada iteração e fase do processo. Só após o gerente de projeto obter a concordância dos grupos envolvidos, é que a iteração seguinte será iniciada.
Supervisão e Acompanhamento do Projeto de Software	(Meta 1) Acompanhar os resultados e desempenhos reais confrontando-os com o plano de Desenvolvimento de Software; (Meta 2) Tomar ações corretivas e gerenciá-las sempre que houver desvios significativos do plano de Projeto de Software. (Meta 3) Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e pessoas envolvidas.	(Rel – Meta 1) Todos os resultados no final das iterações e no final das fases são comparados com os Planos de Iteração e Plano de Projeto de Software, respectivamente. Uma lista de riscos conduz o processo de avaliação dos resultados. (Rel – Meta 2) Os resultados em cada marco de referência (fases) e nas iterações são comparados com os planos iniciais. Estas avaliações e alterações norteiam a próxima iteração/fase. (Rel – Meta 3) Alterações nos compromissos são acordadas nas iterações e fases por todos os *stakeholders* envolvidos.

Tabela 2.1 Relacionamento do RUP com as KPAs CMM nível 2 (Continuação)

[2] Interessados no sistema que está sendo desenvolvido.

KPAs	Metas	Processo Unificado
Garantia de Qualidade de Software	(Meta 1) Planejar as atividades de GQS; (Meta 2) Verificar objetivamente conformidade de artefatos em relação a padrões, procedimentos e requisitos; (Meta 3) Informar grupos e pessoas envolvidas quanto a resultados e atividades de GQS. (Meta 4) Encaminhar à gerência sênior todas as questões de não conformidade não resolvidas no projeto.	(Rel – Meta 1, Meta 2, Meta 3, Meta 4) *Milestones* apresentam objetivos bem definidos e critérios específicos de conclusão que servem como base para auditorias. O processo possui atividades de revisões e papéis definidos responsáveis por realizá-las. Os resultados das avaliações e revisões servem como base para o planejamento da próxima iteração. O RUP provê modelos de documentos a serem utilizados como padrão do projeto, e que permitem que sejam documentadas políticas para tratar questões de não conformidade.
Gerência de Contrato de Software	(Meta 1) A contratante seleciona contratadas de software qualificadas; (Meta 2) A contratante e a contratada concordam com a reciprocidade de seus compromissos; (Meta 3) A contratante e a contratada mantêm comunicação ao longo de todo o contrato; (Meta 4) A contratante acompanha o desempenho e resultados reais da contratada, comparando-os com os compromissos assumidos.	Não é diretamente atendido no RUP, porém, as ferramentas, técnicas e mecanismos existentes no RUP podem ser utilizados para acompanhar o contrato.
Gerência de Configuração de Software	(Meta 1) Planejar as atividades de gerência de configuração de Software; (Meta 2) Identificar, controlar e tornar disponível os artefatos de software selecionados; (Meta 3) Controlar as alterações nos artefatos de software identificados; (Meta 4) Informar pessoas e grupos envolvidos acerca do estado do conteúdo das *baselines* de software.	(Rel – Metas 1 e 2) O fato do processo ser Iterativo e Incremental é um dos principais motivos da necessidade da gerência de configuração de software. O processo estabelece um Plano de Integração de *builds*[5], o qual necessita gerenciamento de configuração para um controle do incremento de funcionalidade em cada iteração. O processo também atenta para a necessidade de um Plano de Gerenciamento de Configuração que envolve todas atividades de gerência de configuração necessárias no processo. (Rel – Metas 3 e 4) É indicado como necessário realizar estas metas, mas a forma como isto deve ser feito não é detalhada.

Tabela 2.1 Relacionamento do RUP com as KPAs CMM nível 2 (Continuação)

[5] Versões executáveis de uma parte do sistema sendo desenvolvido.

2.3.1 KPA Gerência de Requisitos

Uma das características chave do RUP é ser dirigido a casos de uso. Casos de uso representam uma abordagem sistemática para elicitar, documentar, organizar e comunicar os requisitos de um sistema. Esta é uma das características que alinha o RUP aos objetivos da KPA Gerência de Requisitos do CMM Nível 2. Além disso, através do seu fluxo de Configuração e Gerência de Mudanças, o RUP propicia o gerenciamento e rastreamento dos requisitos, bem como o controle de configuração da evolução dos artefatos.

A meta 2 da KPA em questão está associada com a consistência dos planos, artefatos e atividades em relação aos requisitos do software. Como os requisitos são inicialmente documentados através de um modelo de casos de uso e este modelo serve de base para o desenvolvimento dos modelos posteriores, tanto os planos quanto os artefatos e atividades são realizados de forma mais consistente com os requisitos. Além disso, por o processo evoluir iterativamente, com avaliações sendo realizadas ao final das iterações, e promover a participação direta de clientes e desenvolvedores nas mudanças que freqüentemente alteram os requisitos, o RUP termina por promover um maior controle e consistência dos requisitos.

Dessa forma, a abordagem orientada a casos de uso do RUP contribui no sentido de capturar e entender os requisitos do usuário, e assegurar sua consistência e a aderência do desenvolvimento do projeto aos mesmos.

2.3.2 KPA Planejamento de Projeto de Software

Para KPA Planejamento de Projeto de Software (PPS), o RUP oferece uma série de documentos que capturam o plano e objetivos do projeto, bem como registram avaliações do seu andamento.

Um dos objetivos do RUP é assegurar que as expectativas das partes envolvidas no projeto estejam sincronizadas. Para isso, propõe a realização de estimativas de esforço e

recursos, e documenta-as no Plano do Projeto. Esse procedimento contribui para o alcance da meta 1 desta KPA que está associada a documentação das estimativas de software para serem usadas no planejamento e acompanhamento do projeto. Outros documentos do RUP, como o Plano de Desenvolvimento de Software, Plano de Métricas, Lista de Riscos, Plano de Projeto, Plano de Iteração e Avaliação da Iteração, contribuem para o alcance da meta 2, que diz respeito ao planejamento e documentação das atividades e dos compromissos do projeto de desenvolvimento de software.

A meta 3 desta KPA, obter a concordância dos grupos e das pessoas envolvidas quanto aos respectivos compromissos relacionados ao projeto de desenvolvimento de software, é auxiliada com as revisões do Plano do Projeto e Plano de Iteração, onde todos os *stakeholders* do projeto tomam conhecimento do planejamento e chegam a um consenso do que deverá ser desenvolvido. Além disso, antes do início de cada iteração, os envolvidos no projeto avaliam e aprovam a iteração. Só após o gerente de projeto obter a concordância dos grupos envolvidos, é que a iteração poderá ser realizada.

2.3.3 KPA Supervisão e Acompanhamento de Projeto de Software

Para atingir as metas da KPA Supervisão e Acompanhamento de Projeto de Software (SAPS), o RUP contribui, entre outras formas, através das atividades do seu fluxo de Gerenciamento de Projetos. Por exemplo, para alcançar a meta 1, que está relacionada ao acompanhamento dos resultados e desempenhos reais do desenvolvimento de software, o RUP sugere revisões ao final de cada iteração, que servem como ponto para avaliações de decisões e registro de lições aprendidas para direcionamentos futuros. Além disso, o RUP tem vários níveis de Planos de Projetos e Relatórios de Avaliação da Situação do Projeto que são utilizados na avaliação e registro das atividades planejadas e executadas.

O desenvolvimento iterativo sugerido pelo RUP é orientado a riscos. A avaliação desses riscos conduz todo o processo de desenvolvimento do software, gerenciando ações corretivas e as solicitações de mudanças para as iterações seguintes. As avaliações realizadas ao final das iterações permitem o acompanhamento dos desvios do

planejamento e dos problemas de mudança de escopo, que são tratados através do fluxo de Gerência de Configuração e Mudanças. Com isso, a meta 2, que refere-se às ações corretivas para resultados ou desempenhos reais que desviarem significativamente do plano de desenvolvimento de software, pode ser satisfeita.

Para a meta 3, que trata das alterações nos compromissos de software serem realizadas em acordo entre os grupos e as pessoas envolvidos, o desenvolvimento iterativo proposto pelo RUP propicia aos *stakeholders* visibilidade do progresso do projeto, permitindo que sejam identificadas mudança necessárias para garantir o andamento do projeto dentro do planejado.

2.3.4 KPA Gerência de Contrato de Software

Quanto ao Gerenciamento de Contrato de Software, este não é diretamente atendido pelo RUP, porém as ferramentas, técnicas e mecanismos existentes no RUP podem ser utilizados para conduzir o contrato, tornando o processo de contratação gerenciável. Pois, desde que os contratos com terceiros sigam o mesmo plano de desenvolvimento de projeto da organização, este poderá ser envolvido nos principais *milestones* e avaliações de situação do projeto. Da mesma forma, as decisões referentes aos contratos com terceiros devem ser documentadas no projeto.

2.3.5 KPA Garantia da Qualidade de Software

Para Garantia da Qualidade de Software, cada *milestone* do RUP apresenta objetivos bem definidos e critérios específicos de conclusão que servem como base para auditorias. Além disso, cada atividade do processo possui atividades de revisões e papéis definidos responsáveis por realizá-las. Associada a cada revisão existe um conjunto de *ckeckpoints* que devem ser avaliados antes de ser iniciada a atividade seguinte. O RUP também provê modelos de documentos a serem utilizados como padrão do projeto. Com esses mecanismos, as metas da Garantia da Qualidade podem ser mais facilmente alcançadas, mas precisam sempre contar com o apoio da organização.

2.3.6 KPA Gerência de Configuração de Software

Por fim, para atender às metas da KPA Gerência de Configuração de Software, o RUP possui dois instrumentos principais: o Plano de Gerência de Configuração e o Plano de Integração de *builds*. O primeiro descreve o controle de configuração e gerenciamento do processo para garantir que os produtos gerados sejam facilmente identificados, controlados e avaliados. Também é definido no Plano de Gerência de Configuração as formas de aprovações das solicitações de mudanças, níveis de controle e formas de comunicação entre os interessados. Já o Plano de Integração de *builds* provê detalhes dos itens de configuração a serem construídos e a ordem que devem ser integrados numa dada iteração. O RUP também defende a necessidade de um sistema de gerenciamento de mudanças para adequadamente gerenciar custos, acompanhar e implementar solicitações de mudanças.

2.4 Considerações Gerais

Neste capítulo foi apresentado o Capability Maturity Model (CMM), modelo para avaliação de processo de desenvolvimento de software proposto pelo Software Engineering Institute (SEI). Este modelo fornece às organizações diretrizes para controlar seus processos de desenvolvimento de software, desenvolvendo uma cultura de excelência em engenharia e gerenciamento de projetos de software. Dentre os cinco níveis de maturidade propostos pelo CMM, focamos no Nível 2. Isto porque, como mostra Fiorino [1], a grande maioria das empresas ainda possuem processos de desenvolvimento de software imaturos, e o primeiro passo a caminho da maturidade seria uma mudança cultural com ênfase em processos gerenciais para atingir o Nível 2 de maturidade.

Apresentamos também o processo de engenharia de software definido pela empresa *Rational* [5], o RUP, que cobre todo o ciclo de desenvolvimento de software e provê uma abordagem disciplinada para atribuir tarefas e responsabilidades durante o desenvolvimento de software dentro de uma organização. Seu principal objetivo é

assegurar a produção de software de alta qualidade, que atenda às expectativas do seu usuário final dentro do cronograma e do orçamento previsto.

Por fim, fizemos um relacionamento das características e princípios do RUP com o CMM 2, mostrando como este processo satisfaz às metas das KPAs do CMM 2.

No próximo capítulo estaremos apresentando o conceito de métricas, suas características e a importância da escolha adequada de métricas para o planejamento e acompanhamento de projetos, foco das KPAs Planejamento de Projeto de Software e Supervisão e Acompanhamento de Projeto de Software.

Capítulo 3

Uso de Métricas no Gerenciamento de Projetos de Software

As métricas e estimativas de software vêm se tornando um dos principais tópicos na Engenharia de Software [2]. Com a crescente exigência pela melhoria da qualidade de software, através do desenvolvimento dentro dos prazos e custos estimados e atendendo às expectativas do cliente, é impossível não enxergar tais técnicas como alavanca para melhoria da qualidade do desenvolvimento de software.

Neste capítulo, apresentamos o conceito de métricas de software e sua importância para o planejamento e acompanhamento de projetos de desenvolvimento de software.

3.1 Métricas de Software

As métricas de software tornaram-se uma ferramenta fundamental para as atividades de planejamento e acompanhamento de projetos, consideradas umas das atividades mais importantes do processo de desenvolvimento de software.

As medições realizadas em um projeto ajudam-nos a [11]:

- Ter uma melhor visão do processo de desenvolvimento,
- Identificar e gerenciar os riscos,
- Identificar e resolver problemas antes que se tornem críticos,
- Melhorar a comunicação da equipe com a organização,
- Avaliar o desempenho do projeto,
- Tomar decisões.

Métricas de software são utilizadas para medir atributos específicos de produtos de software ou de processos de desenvolvimento de software. São utilizadas para derivar uma base para estimativas, acompanhar progresso do projeto, determinar complexidade, ajudar a determinar quando o estado aceitável de qualidade foi atingido, analisar defeitos, e validar experimentalmente as melhores práticas utilizadas no processo. Em resumo, as métricas nos ajudam a tomar decisões mais conscientes.

Distinguem-se dois tipos de métricas:

- A métrica derivada, ou simplesmente métricas, é um atributo mensurável de uma entidade. Po exemplo, esforço de um projeto é uma métrica de tamanho de projeto. Para calcular esta métrica, será preciso somar os tempos para realizar as atividades do cronograma do projeto.
- A métrica primitiva é o dado que é usado para calcular a métrica. No exemplo anterior, os tempos das atividades do cronograma são as métricas primitivas. As métricas primitivas são basicamente os dados que estão armazenados na base, sem uma interpretação específica. Cada métrica é construída a partir de uma ou mais métricas primitivas coletadas.

As métricas devem possuir um objetivo bem definido e características específicas, tais como:

- Definição
- Questões a responder
- Dados primitivos ou fórmulas
- Visibilidade (quem pode ter acesso)
- Procedimentos de coleta
- Interpretação envolvendo limites inferiores e superiores para seus valores
- Responsável

Como exemplo de métricas podemos citar: Número de solicitações de mudanças de requisitos. Esta métrica tem como objetivo determinar a estabilidade dos requisitos de um projeto e pode ser caracterizada da seguinte forma:

- Definição: Número de requisições de alterações nos requisitos, totalizadas por fase do projeto.
- Questões a serem respondidas: Qual é a freqüência de mudanças nos requisitos? Em que fase do projeto ocorre mais solicitação de mudanças?
- Dados primitivos e fórmulas: número de requisições de mudanças registradas.
- Visibilidade: toda equipe do projeto.
- Procedimentos de coleta: Totalizar as solicitações de mudanças registradas na ferramenta de gerência de mudanças para uma dada fase do projeto.
- Responsável: Engenheiro de Requisitos.

As métricas podem ser utilizadas para estimar esforço, cronograma e recursos baseados em fatores como tamanho e produtividade; predizer, que diferente de estimar é utilizado para calcular valores futuros baseados em valores atuais e outros fatores de influência, como por exemplo predizer como o sistema irá se comportar com determinada carga baseando-se em amostras de dados de performance ou usar a taxa de erros para predizer quando o sistema deve atingir certo nível de estabilidade; e avaliar, que é utilizado para estabelecer comparações, avaliar alternativas e posição atual do projeto.

Medir é importante primeiramente para controlarmos os projetos e assim, podermos gerenciá-los. Nós medimos para avaliar o quão próximo estamos dos objetivos que definimos para o projeto quanto a qualidade, esforço, custo, requisitos. Medimos para sermos capazes de melhor estimar esforços, custo e qualidade dos projetos baseando-se nas experiências anteriores. Finalmente, medimos para avaliar como poderemos melhorar aspectos chave ao longo do projeto e avaliar suas mudanças.

Medir aspectos do projeto gera um custo adicional, portanto, devemos estabelecer objetivos precisos a serem medidos e apenas coletar métricas que irão nos ajudar a atingir esses objetivos. Os três objetivos primordiais de um gerente de projetos são [17]:

- Acompanhar o progresso do projeto
- Prover visibilidade do projeto para revisões de gerenciamento
- Assegurar o sucesso do projeto

Reconhecendo esses objetivos gerais, devemos definir as métricas necessárias ao seu alcance. Com o tempo, após medirmos vários projetos, poderemos aplicar técnicas estatísticas para aperfeiçoar as métricas inicialmente definidas.

Segundo RUP 2000 [32], podemos categorizar os objetivos da medição da seguinte forma:

- Objetivos da organização: Uma organização precisa conhecer seus custos por produto, tempo de desenvolvimento e qualidade dos seus produtos. Precisa reduzir seus riscos e aumentar a performance para manter-se competitiva. Uma organização precisa estar apta a introduzir novas tecnologias e medir o custo x benefício de suas mudanças. Como exemplos de métricas relevantes à organização, podemos citar: custo por ponto de função, custo por caso de uso, custo por linha de código, esforço por linha de código ou por ponto de função, defeitos por linha de código.
- Objetivos dos Projetos: Projetos devem ser entregues com os requisitos funcionais e não-funcionais dos seus clientes atendidos, no prazo e orçamento definido e atendendo às restrições do cliente. Para tal, é preciso o auxílio de métricas para responder a perguntas como: O projeto está atendendo aos *milestones* definidos? Como estão os esforços e custos atuais do projeto comparados ao planejado? Os requisitos funcionais estão completos? O sistema está atendendo a performance desejada? Além de definir as métricas para auxiliar nessas questões, é preciso definir interpretações e ações a serem tomadas com base nos resultados das

métricas, mas isso só poderá ser possível a partir de avaliações dos dados históricos de projetos passados.

- Objetivos Técnicos: As métricas para as necessidades técnicas são de características estruturais e comportamentais. Alguns exemplos dessas métricas são: freqüência de mudança de requisitos (volatilidade), requisitos corretos e completos (validade, completude), projeto realizando todos os requisitos (completude), estabilidade da arquitetura, extensão dos testes para cobrir todo o sistema, incidência de defeitos por atividade e fases.

3.2 Gerenciamento de Projetos de Software

O papel do gerente de projeto consiste em alcançar os objetivos do projeto, distribuindo o produto com alta qualidade, no tempo previsto e com o preço justo. É responsabilidade do gerente de projeto medir o progresso do projeto, provendo visibilidade para as revisões gerenciais. Para isso ele precisa estimar, planejar, definir cronograma e acompanhar o progresso do projeto. A seguir, descrevemos algumas das atividades da gerência de projeto, segundo diversos autores [11, 13, 29, 31]:

- Definição do Escopo: Antes que um projeto possa ser planejado, os objetivos e o escopo do projeto devem ser estabelecidos, soluções alternativas devem ser consideradas e as restrições administrativas e técnicas identificadas. Sem essas informações é impossível definir estimativas de custo razoáveis e precisas, uma divisão realística das tarefas de projeto ou uma programação de projeto administrável que ofereça indícios significativos de progresso.
- Planejamento: Uma das principais características para uma organização obter sucesso no gerenciamento de projetos é a existência de planejamento. O planejamento focaliza nas atividades e objetivos do projeto, preocupando-se com a funcionalidade, o custo, o cronograma e a qualidade, junto com os recursos e marcos de referência relacionados.
- Análise de Riscos: A implementação de uma solução é baseada em planos (de projeto e de período, como por exemplo iterações) que contêm estimativas, aproximações, incertezas e consequentemente, envolve riscos. A identificação

dos riscos existentes para realização do projeto, bem como o estudo de maneiras para evitá-los e armazená-los são tarefas essenciais para o gerenciamento de qualquer projeto de software.

- Estimativa de Recursos: O gerente deve avaliar o escopo do projeto e identificar os recursos pessoais e de ferramentas necessários à execução do projeto, verificando as habilidades exigidas para concluir o desenvolvimento, disponibilidade dos recursos, duração das atividades, especificação de hardware e software e períodos em que serão requeridas cada ferramenta.
- Monitoramento do Desenvolvimento: A atividade de monitoração e controle inicia-se logo que o planejamento do desenvolvimento for concluído. Cada tarefa previamente planejada e identificada no plano de desenvolvimento é rastreada pelo gerente de projeto. O gerente pode usar uma ferramenta de planejamento e controle de projetos para determinar o impacto do não cumprimento dos prazos sobre os marcos de referência intermediários do projeto e da data de entrega global. Recursos podem se redirecionados, tarefas reordenadas e compromissos de entrega modificados para acomodar o problema que foi descoberto, resultando em um desenvolvimento de software mais controlado. O desenvolvimento de um projeto de software é monitorado através da identificação e coleta das métricas que auxiliam a avaliar se os aspectos de cronograma, custo e qualidade estão sendo desenvolvidos de acordo com o planejado. Desse modo, é necessária a existência de uma disciplina de utilização de métricas que permita identificar, coletar e avaliar informações relevantes, que identifiquem o andamento do projeto.

A gestão do software consiste nas ações necessárias para gestão do projeto, do ambiente de desenvolvimento e do produto, visando atingir objetivos previamente estabelecidos no que concerne à produtividade e qualidade, bem como ao atendimento das necessidades dos usuários.

As ações necessárias para a gestão do projeto podem ser categorizadas em:

- Planejamento do projeto, que entre outras coisas compreende elaborar o escopo preliminar do produto a ser desenvolvido, elaborar estimativas de prazos, recursos, esforço, custos e tamanho do produto, e definir pontos de controle do projeto.
- Controle do projeto, que compreende controlar a alocação de recursos, verificar e validar produtos intermediários, controlar mudança de escopo, refinar o planejamento inicial, acompanhar as realização das tarefas conforme planejado e acompanhar a realização orçamentária.

Nesse contexto, o uso de medições de software é essencial para fornecer aos gerentes de projeto informações tangíveis para planejar o projeto, realizar estimativas, gerenciar e controlar os projetos com maior precisão. A realidade aponta para a necessidade de medições visto que, na maioria dos projetos de software:

- As estimativas de prazos, recursos, esforço e custo são realizadas com base no julgamento pessoal do gerente de projeto;
- Estimativa de tamanho de software não é realizada;
- Produtividade da equipe de desenvolvimento não é mensurada;
- A qualidade dos produtos (internos e finais) não é medida;
- Fatores que impactam a produtividade e a qualidade não são determinados;
- Os custos de não conformidade ou de má qualidade não são medidos.

Dessa forma, não se aproveitam as experiências passadas para aperfeiçoar o processo de desenvolvimento e a capacidade de estimativa para projetos futuros.

Valores coletados pelo Hewlett-Packard [8] em 125 projetos revelam que os projetos investem cerca de 18% do esforço total na fase de especificação dos requisitos, 19% na fase de projeto, 34% na codificação e 29% nos testes. Várias outras organizações coletaram essas informações e o resultado variou em torno de 10% a mais ou a menos desses percentuais. Cada organização deve coletar essas informações referentes a seus projetos, e estabelecer seus próprios percentuais. Essas dados são fáceis de serem

coletados, e uma vez coletados, representam uma informação interessante de como a organização utiliza seu tempo e esforço, ajudando ao gerente de projeto na estimativa de esforço do seu projeto, bem como no acompanhamento do mesmo.

3.3 Métricas no Gerenciamento de Projetos de Software

As métricas são ferramentas essenciais ao gerenciamento de projetos de software. A escolha das métricas está intimamente associada às estratégias e objetivos da organização, e vai depender do estágio de maturidade em que a mesma se encontra. As métricas coletadas devem prover informações que ajudem na tomada de decisões de acordo com os objetivos e estratégias da organização. Alguns desses objetivos podem ser: melhorar a qualidade do planejamento do projeto, reduzir os custos de retrabalho no processo, melhorar a qualidade do processo de desenvolvimento, melhorar a qualidade do produto resultante, reduzir os custos de falha, aumentar a produtividade do desenvolvimento, aperfeiçoar continuamente os métodos de gestão do projeto. Por essa razão, o uso de métricas tem se tornado uma grande vantagem estratégica [8] .

A gestão de projetos de software somente atinge determinado nível de eficácia e exatidão se houver medidas que possibilitem gerenciar através de fatos e, o que é mais importante, gerenciar os aspectos econômicos do software, que geralmente são negligenciados em organizações de desenvolvimento.

Para implementação de um programa de métricas é preciso identificar os tipos e categorias de métricas (tais como métricas de tamanho de software, métricas de qualidade, métricas de produtividade e outras), os usuários das métricas e audiências (destinatários dos resultados). É preciso definir métricas simples de entender e objetivas, visando reduzir a influência de julgamento pessoal na coleta, cálculo e na análise dos resultados. As métricas devem ser informativas, ou seja, devem propiciar informações que possibilitem avaliar acertos (ou não) de decisões e ações realizadas no passado, evidenciar a ocorrência de eventos presentes que subsidiem decisões tempestivas, bem como prever a possibilidade de ocorrências de eventos futuros. Além disso, as métricas

devem ser efetivas no custo, ou seja, o valor da informação obtido como resultado das medições deve compensar o custo de coletar, armazenar e calcular as métricas.

Brito e Abreu em [27] relacionam algumas características que as métricas devem apresentar:

- Métricas devem ser bem definidas.
- Métricas devem ser dimensionáveis ou expressas em alguma unidade.
- Métricas devem ser obtidas o mais cedo possível no ciclo de vida do sistema.
- Métricas devem ser facilmente calculadas.
- Métricas devem estar em uma escala que aumente sua precisão.
- Métricas devem ser vistas como probabilidades de forma a permitir a aplicação de teorias estatísticas sobre as mesmas.

Já Gilb [28] ressalta outras características que as métricas de um processo de desenvolvimento de software devem possuir:

- As métricas devem ser robustas, ou seja, devem ser precisas e relativamente insensíveis a pequenas mudanças em ferramentas, métodos ou características do produto.
- As métricas devem sugerir uma norma, indicando os valores ótimos e os valores a serem evitados.
- As métricas devem relacionar propriedades do produto ou processo.
- As métricas devem sugerir uma estratégia de melhoria.
- As métricas devem ser resultados naturais de um processo, ou seja, as atividades necessárias para coleta e avaliação das métricas não devem prejudicar o desenvolvimento do produto.
- As métricas devem ser simples.
- As métricas devem ser intuitivas e rastreáveis.

Ainda no processo de definição de métricas de software, é importante definir quais métricas serão públicas e quais métricas serão privadas. Essa classificação ajuda a tratar

as reações das pessoas às medições e a propiciar um clima de maior confiança na coleta de informações.

Métricas sobre defeitos são um exemplo comum de métricas que as pessoas preferem que sejam privadas. As pessoas não gostam de ter seus erros apontados, muito menos de ter seus erros divulgados. A questão é saber analisar esses dados com o objetivo de melhorar o processo e não avaliar as pessoas.

No que diz respeito à privacidade das métricas, podemos classificá-las da seguinte forma:

- Privadas para o indivíduo, como por exemplo, taxa de defeitos por indivíduo.
- Privadas para o grupo do projeto, como por exemplo, taxa de defeitos por módulo.
- Pública para organização, como por exemplo, taxa de defeito de um dado projeto.

Pensar em termos de dados públicos e privados ajuda tanto ao gerente quando ao desenvolvedor a entender melhor quem deve ter acesso a que métricas e como o conhecimento sobre as métricas deve ser aplicado. Dessa forma, fica claro o entendimento de como e quando cada métrica deve ser utilizada.

O sucesso das decisões tomadas baseadas em métricas será dependente da qualidade das métricas utilizadas. Manter a integridade dos dados e honrar a privacidade dos mesmos quando apropriado é a forma de criar um ambiente confiável, onde as pessoas irão trabalhar com maior eficiência. Nesse contexto, vale lembrar que métricas não devem ser utilizadas para medir e avaliar pessoas. Uma vez utilizadas com esse objetivo, as reportagens das métricas podem ser manipuladas e não retratar a realidade.

De acordo com o objetivo da medição, as métricas podem servir à gestão estratégica, tática e operacional de uma organização. As medições estratégicas são obtidas a partir de agregações das medições operacionais e táticas, e devem possibilitar a realização de *benchmarking*; melhorias contínuas no tocante à qualidade dos métodos de planejamento de projetos, gestão do processo de desenvolvimento e gestão do produto; e avaliação

econômica do ativo de software. Como exemplo dessas medições podemos citar: nível de satisfação do cliente, custo médio de desenvolvimento, melhoria da produtividade do desenvolvimento, benchmarking da produtividade do desenvolvimento.

As medições táticas dizem respeito à gestão do ambiente de software em termos de impacto da introdução de novas ferramentas, mudanças no processo de desenvolvimento, análise de tendências da produtividade. Alguns exemplos dessas medições são: tendência da densidade de defeitos de software, tendência da exatidão das estimativas relativas a projeto, tendência da produtividade do desenvolvimento por tipo de processo.

Já as medições operacionais, ocorrem em nível de cada projeto, visando subsidiar o planejamento de projetos, a gestão do processo de desenvolvimento, o planejamento do atendimento ao produto de software e à própria gestão do produto. Será neste tipo de medição que estaremos nos concentrando neste trabalho.

Dentro das medições operacionais, ressaltamos as medições realizadas no planejamento de projetos que vão subsidiar a gestão do processo de desenvolvimento de software. Como exemplos dessas métricas podemos citar:

- Estimativa de tamanho do software
- Estimativa do prazo do projeto
- Estimativa do esforço do projeto
- Custo estimado para o projeto
- Estimativa da distribuição de esforços por fase do projeto
- Estimativa da distribuição de prazo por fase do projeto
- Estimativa do número de defeitos pré-release
- Estimativa do número de defeitos pós-release
- Estimativa de esforço de retrabalho
- Estimativa de custo de retrabalho
- Estimativa do número de profissionais por fase do projeto

Apesar das contribuições, até aqui apresentadas, que as métricas propiciam para uma gerência de projetos efetiva, Howard Rubin tem acompanhado a implantação de métricas em organizações de sistemas de informações, e nos mostra que apenas cerca de 20% das organizações que iniciam a implantação de um programa de métricas são bem sucedidas. Rubin considera uma implantação bem sucedida quando [8]:

- Os resultados da mensuração são ativamente utilizados nas tomadas de decisões da organização.
- Os resultados são comunicados e aceitos fora da organização.
- O programa durar mais de 2 anos.

Para o sucesso é preciso um bom planejamento, paciência, liderança, foco nos resultados e muito trabalho. É preciso convencer as pessoas sobre a importância da medição, e criar um ambiente de credibilidade com o uso correto e consistente dos dados.

O desenvolvimento de software ainda é uma atividade muito dependente das pessoas e de sua criatividade. Em se tratando de métricas de software, é necessário definir algumas regras comportamentais para evitar má interpretação e falta de clareza nas métricas coletadas. Muitas das métricas de software coletadas são resultado das atividades desempenhadas pelos desenvolvedores de software e coletadas por eles próprios. Dessa forma, é preciso construir uma atmosfera confiável, que respeite as habilidades de cada pessoa para se medir e identificar as mudanças necessárias ao processo.

A Tabela 3.1 apresenta algumas recomendações para aplicar métricas de software em um projeto, considerando o ponto de vista da organização, do gerente de projeto e da equipe técnica [8].

Organização	- Não permitir que métricas sejam utilizadas para medir individualmente as pessoas. - Definir objetivos claros para as métricas e envolver seu staff na definição das métricas. - Não enfatizar uma métrica, excluindo outras.

	- Garantir que informações levantadas sobre a organização não vão prejudicar a pessoa que as reportou.
Gerente de Projeto	- Alcançar a concordância de toda equipe quanto as métricas a serem acompanhadas, e definir essas métricas em um plano de projeto. - Fornecer *feedback* regularmente à equipe sobre os dados que eles estão coletando. - Conhecer o foco estratégico da organização e, nos relatórios de métricas, enfatizar as métricas que suportam essa estratégia.
Equipe técnica	- Procurar reportar os dados o mais apurado possível e nos períodos acordados. - Ajudar o gerente a focar nos dados do projeto que irão levar à melhoria do processo - Não utilizar os dados levantados em benefício próprio.

Tabela 3.1 Papel da organização em um plano de métricas

Ainda quanto à implantação de métricas, é preciso definir os usuários das várias métricas, como eles as utilizarão e os intervalos de tempo favoráveis para avaliá-las. A Tabela 3.2 apresenta um exemplo do relacionamento entre os vários usuários das métricas e suas necessidades quanto ao uso das mesmas [8].

Usuário primário	Desenvolvedor	Gerente de projeto	Gerente de projeto	Gerente de projeto/Desenvolvedor
Uso dos dados	Entender e alterar os produtos de software produzidos	Identificar tendências e potenciais problemas na área	Ajustar cronograma	Ajustar planejamento, revisar estimativas
Tempo de avaliação	Segundos-minutos	Horas	Semanas	Meses

Tabela 3.2 Usuários das métricas

Para a realização das medições precisamos de três condições técnicas básicas [17]:

- Processo de software definido
- Conceito de *work-product (WP)*
- Banco de dados de métricas

Estas três condições estão intimamente relacionadas, visto que as medições são feitas no contexto de um processo de software, que gera produtos intermediários e finais (*work-product*), e os dados sobre essas medições vão alimentar um banco de dados, cuja finalidade é permitir o monitoramento e aperfeiçoamento do processo, bem como realimentar e subsidiar novos projetos de software. Podemos considerar este relacionamento um ciclo evolutivo, como mostra a Figura 3.1.

No banco de dados de métricas, as métricas devem estar associadas ao tipo de processo de desenvolvimento e ao produto específico para que sua interpretação seja mais precisa, considerando as características do contexto em que foram coletadas. É esse banco de dados que proverá informações para estimativas futuras mais exatas e realísticas. Por isso, é importante também que durante o processo seja medida a exatidão das estimativas.

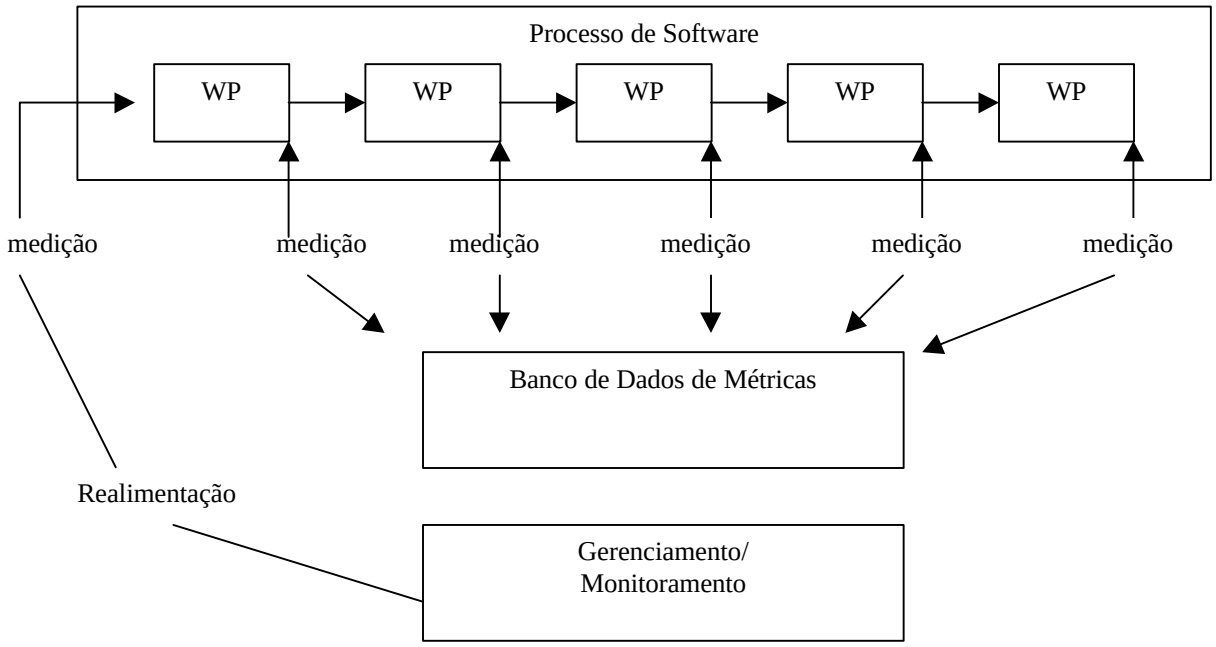

Figura 3.1 – Ciclo evolutivo de medição

No contexto deste trabalho, estaremos utilizando o RUP como processo de desenvolvimento de software e seus artefatos gerados como w*ork-product*.

3.4 Considerações Gerais

As métricas de software são essenciais para controlarmos os projetos e assim podermos gerenciá-los. Elas são utilizadas para derivar uma base para estimativas, acompanhar o progresso do projeto, determinar quando o estado aceitável de qualidade foi atingido, analisar defeitos e validar experimentalmente melhores práticas utilizadas no processo.

Um gerente de projeto deve acompanhar o progresso do projeto, prover visibilidade do projeto para revisões e tomadas de decisões quando o projeto estiver sofrendo desvios do planejamento, e assegurar o sucesso do projeto e a satisfação do cliente. Dessa forma, as métricas tornam-se ferramentas fundamentais para o gerenciamento de projetos de software.

No entanto, a escolha das métricas adequadas e a implantação das métricas na organização não são tarefas fáceis. A escolha das métricas deve estar intimamente associada às estratégias e objetivos da organização, e vai depender do grau de maturidade da mesma. As métricas escolhidas devem ter objetivos bem definidos, alinhados aos objetivos da organização, e devem possuir um procedimento bem especificado para sua coleta, incluindo os responsáveis pelo mesmo, as fórmulas para o cálculo das métricas e interpretação de seu resultado.

Por fim, para garantirmos a qualidade das métricas e o sucesso do programa de implantação das mesmas na organização, é essencial que seja criado um ambiente confiável, onde as pessoas conheçam os objetivos das métricas, e os dados coletados tenham sua integridade mantida, bem como a privacidade necessária para garantir a credibilidade do programa.

No capítulo seguinte apresentaremos um conjunto de métricas para auxiliar o planejamento e gerenciamento de projetos, bem como a abordagem utilizada para a escolha do conjunto, tomando como base as diretrizes do CMM Nível 2.

Capítulo 4

Métricas para o CMM Nível 2 e o Fluxo de Gerenciamento de Projetos do RUP

A seleção e definição de um conjunto de métricas não é uma tarefa fácil. Na verdade, esta é uma tarefa custosa que demanda grande conhecimento para evitar que o seu uso não aumente ainda mais os problemas enfrentados durante o desenvolvimento de software. Esta dificuldade ocorre devido ao grande número de atributos que podem ser medidos durante um projeto. Uma escolha incorreta das métricas pode levar ao aumento desnecessário de esforço e a uma visão distorcida do processo dificultando a sua análise e, muitas vezes, terminando por orientar decisões equivocadas [19].

Neste capítulo iremos apresentar a abordagem utilizada para seleção das métricas propostas nesta dissertação, bem como a caracterização das métricas selecionadas. Na Seção 4.1 detalharemos as etapas e atividades da abordagem na qual nos baseamos para seleção das métricas. Na Seção 4.2 descrevemos o processo realizado para seleção das métricas, e na Seção 4.3 definimos as métricas selecionadas. Na Seção 4.4 apresentamos algumas recomendações para implantação de um programa de métricas, considerando a importância da escolha correta das métricas a serem utilizadas, e por fim, na Seção 4.5 apresentamos um framework para acompanhamento e estimativas de projetos de software através de métricas.

4.1 Abordagem para Seleção de Métricas

Experiências com medições orientadas a objetivos mostram a importância da definição prévia de metas e objetivos para facilitar a seleção de métricas e a correta interpretação dos resultados [19]. Acima de tudo, a experiência tem mostrado que a escolha das métricas deve estar associada aos objetivos da organização, ou seja, as métricas devem

contribuir para o alcance das metas da organização. Neste contexto, iremos citar algumas abordagens que vêm sendo bastante utilizadas.

O Balanced Scorecard (BSC) é uma abordagem desenvolvida por Robert Kaplan e David Norton que visa provê um direcionamento sobre o que a organização deve medir. O BSC traduz a missão e estratégia da organização em objetivos e medidas tangíveis, sugerindo que a organização seja vista sob quatro perspectivas [41]:

- Aprendizagem e Crescimento: direciona a atenção para as pessoas e para as infra-estruturas de Recursos Humanos necessárias ao sucesso da organização.
- Financeira: permite medir e avaliar resultados que o negócio proporciona e necessita para o seu crescimento e desenvolvimento.
- Processos Internos: representam os processos-chave do negócio.
- Cliente: direciona todo o negócio e atividade da empresa para as necessidades e satisfação dos seus clientes.

O BSC sugere que as métricas sejam definidas, coletadas e analisadas sob cada uma dessas perspectivas.

Fatores Críticos de Sucesso (FCS) é uma abordagem mais antiga, sugerida por Rockart (1979: 85) e utilizada por diferentes tipos de organizações, entre elas organizações comercias, industriais e universidades. FCS sugere que sejam identificadas as áreas-chave da organização nas quais os resultados sendo satisfatórios, estarão alcançando os objetivos da organização e assegurando sua vantagem competitiva. Uma vez se concentrando nos fatores essenciais para os objetivos da organização, poderemos identificar as informações que contribuirão e ajudarão a acompanhar o alcance desses objetivos [42].

O método GQM (Goal/Question/Metric), por sua vez, consiste em primeiramente identificar as metas da medição (Goal), depois definir as questões que se deseja responder (Question), para então definir que métricas ou indicadores poderão ajudar (Metric). O GQM se baseia na convicção de que para uma organização medir de forma eficiente, é necessário primeiro especificar os objetivos a serem alcançados, relacionar esses

objetivos com dados reais obtidos através de medições e, finalmente, prover um *framework* para a interpretação desses dados de acordo com os objetivos propostos [19].

Para este trabalho, consideramos o GQM a abordagem mais adequada, uma vez que ela tem seu foco principal na estrutura para escolha, definição e interpretação das métricas, considerando os objetivos da organização sob qualquer aspecto.

As metas da medição podem ser estabelecidas para monitorar e direcionar as atividades de melhoria do processo de desenvolvimento bem como do produto gerado. No entanto, quando forem ser estabelecidas essas metas é importante considerar as seguintes recomendações:

- As metas devem ser possíveis de serem alcançadas, ou serão ignoradas pela equipe.
- As metas devem estar associadas a tempo, isto é, deve ser estabelecida uma expectativa de quando a meta deverá estar sendo alcançada.
- A metas devem ser suportadas por ações, ou seja, não é suficiente estabelecer uma meta, é preciso definir estratégias e prover meios para alcançá-la.
- Para metas a longo prazo, é preciso definir metas intermediárias para acompanhar sua execução.

Para definirmos o conjunto de métricas apresentado neste trabalho, adotamos uma abordagem baseada no trabalho de Donald McAndrews [33]. No seu trabalho, Donald define uma arquitetura para projetar um processo de mensuração de software, baseada no método GQM, de forma que este seja parte integrante do processo de desenvolvimento de software da organização. A arquitetura proposta por Donald consiste de três fases:

- Planejamento: Onde é realizado o planejamento e documentação do processo para organização, e realizada suas devidas adaptações para projetos específicos.
- Implementação: Implementação do processo nos projetos da organização através da execução do planejamento e procedimentos definidos.

- Melhoria: Melhoria do processo através de avaliações dos planos e procedimentos, decorrentes da evolução da maturidade dos projetos e necessidade de mudança das métricas.

Especificamente para a definição do conjunto de métricas deste trabalho, seguimos apenas as etapas da fase de planejamento, que estão descritas a seguir.

Etapa 1: Identificar o escopo da atividade de medição.

Esta etapa consiste em estabelecer o propósito do programa de métricas que inclui as seguintes atividades:

- Identificar as necessidades para medição.
- Identificar os objetivos da organização que serão atendidos, verificando se as necessidades de medição levantadas estão alinhadas aos objetivos da organização.
- Identificar os objetivos da medição.
- Identificar questões para cada objetivo, que quando respondidas, irão determinar se os objetivos estão sendo alcançados.
- Para cada questão, identificar o que deverá ser medido.

Etapa 2: Definir os procedimentos para medição.

Uma vez identificado o conjunto de métricas, a partir da definição do escopo realizada na etapa anterior, serão definidos os procedimentos para medição da seguinte forma:

- Definir as métricas em termos operacionais, descrevendo o que consiste cada métrica, os dados que as compõem, a unidade de medida a ser adotada e outras informações necessárias para que exista um entendimento único e uniforme das medidas.
- Definir procedimentos para coleta das métricas incluindo a origem dos dados, responsáveis pela coleta, freqüência da coleta e ferramentas especiais requeridas.
- Definir formato e mecanismo de armazenamento.

Durante a escolha do conjunto de métricas, além de seguirmos a abordagem proposta por McAndrews, tivemos a preocupação de definir um conjunto balanceado de métricas, que monitorasse a performance do grupo de desenvolvimento em vários aspectos complementares. Essa preocupação visa evitar disfunções no comportamento da equipe, onde um membro da equipe pode alterar seu desempenho para otimizar o aspecto que estará sendo medido, e não focar no real objetivo da medição. Por exemplo, uma vez que esteja sendo medida a produtividade da equipe, mas não a qualidade do produto gerado, pode-se esperar que a equipe que estiver sendo medida modifique seu estilo de programação para gerar maior volume de código, sem se preocupar com a qualidade do código gerado. Um conjunto balanceado de métricas que monitore diferentes aspectos do desenvolvimento de software identificará tal comportamento.

Todas as métricas utilizadas neste trabalho foram extraídas ou adaptadas da literatura [8, 11, 13, 17]. Entretanto, é importante ressaltar que este conjunto de métricas aqui apresentado não se propõe a ser o melhor, mesmo porque não foi este o objetivo deste trabalho, mas é adequado e suficiente para permitir o planejamento e acompanhamento de projetos de software, de acordo com os objetivos estabelecidos nesta abordagem. Consideramos que é melhor ter um conjunto inicial para poder realizar as medições, mesmo que este não seja perfeito, e depois melhorá-lo a partir do aprendizado obtido com a experiência, a permanecer longo tempo em discussões buscando as métricas perfeitas.

4.2 Seleção das Métricas

Nesta seção, iremos descrever como selecionamos o conjunto de métricas proposto nesta dissertação, utilizando a abordagem de McAndrews, descrita na Seção 4.1.

A Etapa 1 da abordagem de McAndrews consiste em identificar o escopo da atividade de medição. Nesta etapa, a primeira atividade é “identificar as necessidades de medição”. Escolhemos um conjunto de métricas que pudesse prover ao gerente de projeto visibilidade do progresso dos projetos de desenvolvimento de software, permitindo o

acompanhamento da execução do projeto segundo o planejamento, identificando problemas no projeto e no processo utilizado e criando uma base histórica para estimativas. Esta seria então, a necessidade primária para medição: medir para acompanhar e controlar os projetos, e melhorar o processo de desenvolvimentos de software.

Prosseguindo com as demais atividades da Etapa 1, os passos seguintes para escolha do conjunto de métricas são:

- Identificar os objetivos da organização. Neste trabalho, estamos considerando uma organização que busca melhorar seu processo de desenvolvimento de software, buscando CMM Nível 2, e que utiliza um processo de desenvolvimento de software baseado no RUP.
- Identificar os objetivos da medição. No escopo deste trabalho, os objetivos da medição estão atrelados às metas das KPAs do CMM Nível 2.
- Identificar questões para cada objetivo e para cada questão, o que deverá ser medido. Dessa forma, para cada KPA do CMM 2 selecionamos algumas questões que irão contribuir para o alcance de suas metas, e com base nessas questões, selecionamos um conjunto de métricas que irá contribuir com as respostas a essas questões.

Esses passos estão demonstrados a seguir, para cada KPA do CMM Nível 2 abordada neste trabalho.

A KPA Gerência de Requisitos tem como propósito atender às necessidades e expectativas do cliente, garantindo um entendimento comum entre as partes interessadas com relação aos requisitos a serem atendidos no sistema. [1]. Para isso, é preciso acompanhar e controlar os requisitos do sistema, garantindo sua qualidade e entendimento, e garantindo que os mesmos estão sendo atendidos pelo sistema. Além disso, é preciso controlar as alterações de mudanças nos requisitos, acompanhando as mudanças pendentes e as realizadas.

Diante deste propósito, a KPA Gerência de Requisitos tem as seguintes metas:

- Documentar e controlar os requisitos alocados para estabelecer uma base para todo o processo de desenvolvimento.
- Manter planos, artefatos e atividades de software consistentes com os requisitos alocados.

Para essas metas identificamos as seguintes questões a serem respondidas, seguidas do que deverá ser medido para respondê-las:

Questões	**Métricas**
• Em que fase do desenvolvimento as solicitações de mudanças ocorrem com maior freqüência? • O número de solicitações de mudança está diminuindo com o tempo?	M-1. Número de mudanças de requisitos solicitadas por fase de desenvolvimento
• Os requisitos planejados para serem atendidos no *release* estão sendo implementados?	M-2. Número de mudanças de requisitos realizadas x Número de mudanças solicitadas

A KPA Planejamento de Projeto tem como objetivo estabelecer planos de projeto consistentes e realísticos para a execução das atividades de engenharia de software e o gerenciamento do projeto de software, através da realização de estimativas para o trabalho a ser realizado, identificação de riscos e definição do plano para realização do trabalho especificado [1].

Para esse objetivo, o CMM coloca as seguintes metas:

- Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do projeto.
- Planejar e documentar as atividades e os compromissos do projeto de desenvolvimento de software.
- Obter a concordância dos grupos e das pessoas envolvidas quanto aos respectivos compromissos relacionados ao projeto de desenvolvimento de software.

Para essas metas identificamos a seguir as questões a serem respondidas bem como o que deverá ser medido para respondê-las :

Questões	Métricas
• Qual o esforço gasto em cada fluxo de atividades do projeto?	M-3. Distribuição do esforço por fluxo de atividades
• Qual o esforço gasto em cada fase de desenvolvimento do projeto?	M-4. Distribuição do esforço por fase de desenvolvimento
• Qual a produtividade média da equipe?	M-5. Produtividade da equipe por fase de desenvolvimento
• Qual o tamanho estimado do software?	M-6. Tamanho do software

A KPA Supervisão e Acompanhamento de Projeto de Software tem o objetivo de estabelecer uma visibilidade adequada do progresso do projeto, identificando desvios significativos do seu planejamento [1].

As metas desta KPA são:

- Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software.
- Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software.
- Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e as pessoas envolvidas.

Para essas metas identificamos a seguir as questões a serem respondidas bem como o que deverá ser medido para respondê-las :

Questões	Métricas
• Qual o progresso do projeto em relação ao planejado?	M-7. Dias em atraso para alcance dos *milestones* M-8. Número de casos de uso implementados x planejados por iteração
• Qual o desvio entre o esforço real utilizado e o planejado para cada atividade?	M-9. Esforço planejado x realizado por iteração

A KPA Garantia da Qualidade de Software (GQS) tem o objetivo de fornecer à gerência uma visibilidade apropriada do processo utilizado para o desenvolvimento do projeto de software e dos produtos gerados no projeto [1].

Para tal, a KPA Garantia da Qualidade tem as seguintes metas:

- Planejar atividades de GQS
- Verificar objetivamente a conformidade das atividades e dos produtos de software com os padrões, procedimentos e requisitos aplicáveis.
- Informar aos grupos e às pessoas envolvidas quanto às atividades e resultados de GQS.
- Encaminhar à gerência todas as questões de não-conformidade que não possam ser resolvidas no âmbito do projeto de software.

Para essas metas identificamos a seguir as questões a serem respondidas bem como o que deverá ser medido para respondê-las :

Questões	**Métricas**
• O sistema está sendo entregue com um percentual aceitável de erros?	M-10. Número de Bugs/KLOC registrados por teste da iteração M-11. Número de Bugs encontrados após *release*
• Qual a quantidade de bugs em aberto?	M-12. Número de bugs registrados x Número de bugs fechados
• Qual o número de problemas registrados nas revisões? • Quantos problemas ainda não foram resolvidos? • Qual a fase do projeto com maior número de problemas registrados?	M-13. Número de problemas registrados x Número de problemas resolvidos
• Qual o número de não conformidades encontradas nas auditorias? • Quantas não conformidades estão em aberto?	M-14. Número de não conformidades registradas x Número de não conformidades resolvidas.

4.3 Definição das Métricas

Após a seleção do conjunto de métricas, seguimos com a Etapa 2 da abordagem de McAndrews, que é definir os procedimentos para medição de cada uma das métricas selecionadas. Neste momento, para cada métrica estaremos descrevendo sua definição, os procedimentos para coleta e a interpretação do seu resultado. Iremos sugerir algumas interpretações para as métricas, contudo, só com uma base histórica dos projetos da organização é que se poderá, futuramente, definir os limites superior e inferior aceitáveis para o resultado de cada métrica, permitindo uma melhor interpretação dos resultados. É importante salientar que esses limites superior e inferior poderão variar de acordo com tipos e características dos projetos.

A melhor fonte de informação para decidir esses limites aceitáveis são os próprios dados da organização. Isto porque processos variam de organização para organização e muitas métricas não possuem critérios de contabilizarão e processos de cálculos bem definidos. Por esses motivos, comparar as métricas obtidas de organizações diferentes pode levar a interpretações errôneas [40]. Dessa forma, recomendamos que se utilize, sempre que possível, os próprios dados da organização para determinar sua *baseline*. Se dados históricos ainda não estiverem disponíveis para tal, recomenda-se esperar para que uma base histórica razoável seja estabelecida, ou nos casos em que se optar por tomar como referencial inicial dados de outras organizações, atentar para as características dos processos adotados na mesma, características dos projetos cujas métricas foram coletadas, bem como definição das métricas e seu processo de coleta e de cálculo, de forma a evitar interpretações ambíguas.

Uma vez decididos os limites aceitáveis, será possível identificar os momentos a serem tomadas ações corretivas no projeto. O gerente de projeto deve monitorar os resultados das métricas e conduzir o projeto de forma que esses resultados sejam mantidos dentro dos limites definidos. Uma vez que os resultados não estejam dentro dos limites aceitáveis, ações corretivas ou melhorias deverão ser tomadas.

A seguir apresentaremos as definições das métricas propostas neste trabalho, ressaltando que foram consideradas as métricas aplicadas a um processo de desenvolvimento iterativo, baseado no RUP, com quatro fases de desenvolvimento: Concepção, Elaboração, Construção e Transição, as quais possuem objetivos específicos e atividades bem definidas nos vários fluxos de atividades: Requisitos, Planejamento de Projeto, Análise e Projeto, Implementação, Testes, Implantação e Configuração e Gerência de Mudanças [32].

M-1. Número de mudanças de requisitos solicitadas por fase de desenvolvimento

Definição: Quantidade de solicitações de alterações nos requisitos ou inclusões de novos requisitos, totalizadas por fase de desenvolvimento.

Procedimento: Totalizar a quantidade de solicitações de alteração ou inclusão de requisitos a partir das requisições registradas na ferramenta de controle de mudanças adotada no projeto, ou caso a organização não adote ferramenta para este fim, as requisições devem ser feitas utilizando algum procedimento acordado anteriormente, tais como envio de e-mail ou preenchimento de documento de solicitação de mudança, que possam ser arquivados e posteriormente totalizados.

Quando: A coleta desta métrica deve ser consolidada ao final de cada fase do processo de desenvolvimento, e devem ser totalizadas apenas as solicitações realizadas durante a dada fase.

Interpretação: Com essa métrica, podemos identificar as fases do desenvolvimento nas quais estão ocorrendo mais solicitações de mudanças. Se o número de solicitações de mudanças aumenta nas fases finais do desenvolvimento, isto pode indicar que os requisitos não estão sendo bem entendidos ou bem especificados nas fases iniciais do projeto, ou o escopo inicial do projeto está sofrendo grandes variações. O maior número de solicitações de mudanças deve ocorrer nas fases iniciais do projeto.

M-2. Número de mudanças de requisitos realizadas x Número de mudanças solicitadas

Definição: Total das solicitações de alterações nos requisitos, ou inclusões de novos requisitos, que já foram concluídas comparado ao número de solicitações que foram registradas.

Procedimento: Totalizar a quantidade de solicitações de alteração ou inclusão de requisitos a partir das requisições registradas na ferramenta de controle de mudanças adotada, ou caso a organização não adote ferramenta para este fim, as requisições devem ser feitas utilizando algum procedimento acordado anteriormente, tais como envio de e-mail ou preenchimento de documento de solicitação de mudança, que possam ser arquivados e posteriormente totalizados. Esse mesmo procedimento deve ser realizado para contabilizar todas as solicitações registradas que já tiverem sido atendidas com sucesso.

Quando: A coleta desta métrica deve ser consolidada ao final de cada iteração do processo de desenvolvimento.

Interpretação: Com essa métrica, podemos acompanhar as solicitações de mudanças pendentes e realizadas para garantir que o sistema está atendendo a todos os requisitos solicitados. Podemos avaliar se os *releases* estão sendo liberados contemplando os requisitos solicitados. Se o número de solicitações ainda não atendidas for muito alto em relação às solicitações realizadas, isso mostra que o projeto ainda não está atendendo satisfatoriamente às expectativas do cliente.

M-3. Distribuição de esforço por fluxo de atividades

Definição: Entende-se por esforço as horas da equipe do projeto utilizadas para realizar atividades relacionadas ao projeto, sejam estas atividades previstas no cronograma ou não. Esta será a definição de esforço adotada neste trabalho. Dessa forma, a distribuição de esforço por fluxo de atividades é o somatório das horas de todos os membros da equipe gastas nas atividades de um dado fluxo de atividades, em relação ao total das horas gastas no projeto como todo.

Procedimento: As horas de cada membro do projeto deverão ser contabilizadas a partir de um sistema de *timesheet*, que deverá relacionar todas as atividades de cada fluxo de atividades do processo adotado pela organização, incluindo as atividades de gerenciamento do projeto e os fluxos de apoio. Neste sistema, cada membro da equipe estará registrando suas horas de trabalho empregadas em cada atividade. Também estarão sendo contabilizadas as horas dos membros que não são exclusivos da equipe do projeto, pois atendem a vários projetos simultaneamente, tais como pessoas do suporte, designer, e outros. Obviamente, para estes casos serão contabilizadas apenas as horas gastas para o projeto em questão.

A distribuição de esforço para um dado fluxo de atividades(ω) é dada da seguinte forma:

Calcular ω_η, total de esforço gasto para o fluxo de atividades ω na iteração η.

Calcular $E\omega = \Sigma_\eta \omega_\eta$, somatório do esforço gasto para o fluxo de atividades ω em todas as iterações do projeto .

Calcular $\Sigma_\omega E\omega$, somatório do esforço gasto em todos os fluxos de atividades durante todo projeto.

Por fim, podemos calcular a distribuição do esforço por fluxo de atividades:

$$E\omega/\Sigma_\omega E\omega$$

Quando: A contabilização da distribuição real do esforço deverá ser realizada ao final de cada iteração, somando as horas dos membros da equipe registradas no *timesheet* na dada iteração e totalizando-as por fluxo de atividades (ω_η). E ao final do projeto, calcular a distribuição do esforço para cada fluxo de atividades ($E\omega/\Sigma_\omega E\omega$). A contabilização da distribuição planejada poderá ser feita logo após a elaboração do cronograma do projeto, totalizando as horas planejadas para as atividades de cada fluxo de atividades.

Interpretação: Com essa métrica podemos conhecer melhor como estão distribuídos os esforços nos fluxos de atividades para melhor estimar o cronograma do projeto e alocar recursos. Conhecendo as atividades que demandam mais esforço, poderemos identificar o número mais adequado de recursos de cada perfil[6] necessário para o projeto. Além disso, poderemos acompanhar melhor os projetos, pois uma vez identificada uma concentração de esforço diferente do esperado em um dado fluxo de atividades durante o

projeto, poderá caracterizar algum problema na definição do cronograma ou no andamento do projeto.

M-4. Distribuição de esforço por fase de desenvolvimento

Definição: É o total do esforço gasto em cada fluxo de atividades por fase de desenvolvimento do projeto.

Procedimento: Para calcular essa métrica, iremos nos valer dos dados levantados para a métrica anterior (M-3). Iremos totalizar o esforço gasto em cada fluxo de atividades nas iterações pertencentes a uma mesma fase do projeto, da seguinte forma:

$E_F\omega = \Sigma_\eta \omega_\eta, \forall_\eta \in F$. Onde *F* é uma fase de desenvolvimento do projeto.

A distribuição de esforços por fase de desenvolvimento é calculada da seguinte forma:

$$E_F\omega/\Sigma_\omega E_F\omega,$$

Quando: A contabilização da distribuição real do esforço deve ser realizada ao final de cada fase de desenvolvimento, calculando o $E_F\omega$, e dividindo pelo esforço total do projeto ($E_F\omega,/\Sigma_\omega E_F\omega,$). A contabilização da distribuição planejada poderá ser realizada logo após o planejamento do cronograma do projeto.

Interpretação: Com essa métrica podemos conhecer melhor como estão distribuídos os esforços nas fases de desenvolvimento do projeto, para melhor estimar o cronograma do projeto e alocar recursos. Conhecendo as fases que demandam mais esforço, poderemos identificar melhor as fases do projeto que estaremos precisando de mais recursos de um dado perfil. Além disso, poderemos acompanhar melhor os projetos, pois uma vez identificada uma concentração de esforço diferente da esperada em uma dada fase do projeto, poderá caracterizar algum problema na definição do cronograma ou no andamento do projeto.

M-5. Produtividade da equipe por fase do projeto

Definição: Esforço necessário para implementar uma linha de código, em cada fase de desenvolvimento do projeto.

[6] Funções que são assumidas por papéis específicos dentro do projeto.

Procedimento: Utilizar ferramenta ou script para contabilizar as linhas de código do sistema produzidas em uma dada fase de desenvolvimento. Para tal, deveremos totalizar o número de linhas de código do sistema ao final da fase e subtrair do número de linhas de código obtido na fase anterior. Por fim, contabilizar os esforço em horas despendido na dada fase. Esse esforço será contabilizado a partir dos registros efetuados no *timesheet* . A métrica é obtida dividindo o esforço em horas pelo número de linhas de código.

Quando: Esse procedimento será realizado ao final de cada fase de desenvolvimento.

Interpretação: Com essa métrica poderemos conhecer a produtividade média da equipe para melhor estimar o cronograma dos projetos, bem como identificar problemas de produtividade da equipe durante o andamento de um projeto. Ressaltamos a importância da coleta dessa métrica por fase de desenvolvimento, pois claramente a produtividade calculada em esforço/KLOC varia de uma fase para outra, uma vez que a maior parte da construção de linhas de código concentra-se na fase de Construção, quase não existindo na fase de Concepção, por exemplo.

M-6. Tamanho do software

Definição: O tamanho do software poderá ser medido por linhas de código ou número de casos de uso.

Procedimento: Totalizar o número de casos de uso desenvolvidos no projeto e o número de linhas de código do software.

Quando: Recomendamos realizar esse procedimento para cada *release* gerado, ou ao final das iterações.

Interpretação: O tamanho do software é um dado importante para o processo de planejamento do projeto. A partir da estimativa de tamanho do software, o gerente de projeto poderá estimar melhor o custo, esforço e cronograma do projeto. Dessa forma, se for identificado que o crescimento do software a cada *release* estiver sendo maior que o esperado, será preciso avaliar o impacto que os custos e o cronograma do projeto poderão sofrer, bem como avaliar o motivo deste aumento (mudanças de requisitos, problemas na estimativa, má entendimento da complexidade do sistema, pouco reuso de código).

M-7. Dias de atraso para alcance dos *milestones*

Definição: Totalizar os dias entre a data provável para serem completados os *milestones* do projeto e a data prevista no planejamento inicial do mesmo.

Procedimento: Verificar no cronograma do projeto, as datas previstas para os *milestones* no planejamento inicial do projeto e as datas em que eles serão realmente entregues, contabilizando os dias de diferença entre essas datas.

Quando: Recomendamos que seja realizada pelo menos ao final de cada iteração. Para iterações longas (com mais de um mês) poderá ser realizada quinzenalmente ou até semanalmente, a depender do grau de controle no acompanhamento desejado pelo gerente do projeto.

Interpretação: Essa métrica nos permite acompanhar o progresso do projeto, identificando se os atrasos estão dentro de uma margem aceitável, passível de renegociação.

M-8. Número de casos de uso implementados x planejados por iteração

Definição: Verificar o número de casos de uso implementados com sucesso e comparar com o estimado no cronograma da iteração.

Procedimento: Contar os casos de uso testados com sucesso ao final de cada iteração, e comparar com o número de casos de uso previstos para serem implementados nesta iteração segundo o cronograma do projeto.

Quando: Esse procedimento deverá ser realizado ao final de cada iteração.

Interpretação: Essa métrica nos permite acompanhar o progresso do projeto, identificando quanto do projeto já foi realizado em termos de casos de uso.

M-9. Esforço planejado x realizado por iteração

Definição: Totalizar o esforço real gasto nas atividades de um fluxo de desenvolvimento realizadas em uma dada iteração e compará-lo ao esforço estimado no cronograma para essas atividades.

Procedimento: Totalizar as horas de todos os membros da equipe, registradas no *timesheet*, referentes às atividades de um fluxo de atividades, realizadas na iteração

corrente. O somatório dessas horas para cada fluxo de atividades representa o esforço real gasto na iteração para o dado fluxo. O esforço planejado será levantado a partir das horas estimadas no cronograma desta iteração.

Quando: Esse procedimento deverá ser realizado ao final de cada iteração.

Interpretação: Essa métrica permite acompanhar o desvio do esforço planejado em relação ao realizado. Para melhor interpretação dessa métrica é importante também avaliar o progresso do projeto, pois em um determinado momento pode ter sido despendido mais esforço que o planejado, porém o projeto pode ter progredido também mais que o esperado.

M-10. Número de bugs/KLOC registrados por teste da iteração

Definição: Essa métrica representar a densidade de bugs encontrados nos testes da iteração.

Procedimento: Contabilizar os bugs/KLOC registrados no teste de cada iteração.

Quando: Esse procedimento deverá ser realizado após os testes de cada iteração.

Interpretação: Essa métrica permite avaliar a efetividade dos testes, e prover visibilidade ao gerente de projeto quanto a qualidade e confiabilidade do produto gerado. Se a densidade dos bugs, ou seja, o número de bus/KLOC, estiver aumentando a cada teste de iteração, esse dado nos mostra que o número de bugs inseridos durante as iterações está crescendo, o que ocasionará um aumento de esforço para ajustes do código. Com dados históricos dessa métrica, poderemos estimar o custo e esforço necessário para correção de bugs, e já considerá-los no planejamento do cronograma, bem como identificar as fases do projetos onde estão sendo gerados maior número de bugs.

M-11. Número de bugs encontrados após release

Definição: Essa métrica será expressa em bugs/KLOC, e representará os bugs encontrados no sistema após o *release*.

Procedimento: Utilizar alguma ferramenta ou script para contar as linhas de código da versão do sistema que sofrerá *release*, incluindo linhas de comentários. Contabilizar os bugs registrados pelo usuário ou pelos testadores após o *release* do sistema.

Quando: Após cada release do sistema.

Interpretação: Com essa métricas pretendemos melhorar o nível de qualidade dos produtos entregues ao cliente, definindo metas realísticas e aceitáveis para o número de bugs/KLOC existentes nos *releases* liberados.

M-12. Número de bugs registrados x Número de bugs fechados

Definição: Total de bugs que foram registrados, comparados aos bugs já resolvidos.

Procedimento: Totalizar os bugs registrados em um dado período e totalizar os bugs resolvidos neste período.

Quando: Ao final de cada iteração.

Interpretação: O número de bugs registrados retrata a quantidade de esforço e custo necessários para avaliação e correção dos mesmos, bem como a qualidade de desenvolvimento da equipe. O número de bugs fechados retrata a capacidade da equipe de tratar os erros encontrados. Se o número de bugs em aberto não estiver diminuindo com o tempo, pode estar havendo sérios problemas com a qualidade de produto ou na estratégia de testes.

M-13. Número de problemas registrados x Número de problemas resolvidos

Definição: Essa métrica irá totalizar os problemas registrados durante as revisões formais realizadas, e os problemas já resolvidos. Entende-se por revisão formal aquelas realizadas com o cliente, ou outra parte interessada no projeto, para comentários e aprovação de um produto ou atividades técnicas e gerenciais, ou para acompanhamento do progresso do projeto. Essa revisão caracteriza o final de uma fase e aprovação para início da fase seguinte [35]. Os problemas aqui mencionados, são problemas quanto a aceitação do produto por parte do cliente, detectados nessas revisões.

Procedimento: Para calcular esta métrica iremos totalizar os problemas encontrados durante as revisões realizadas ao término de cada iteração do projeto, bem como os problemas que foram solucionados na dada iteração.

Quando: Esse procedimento deverá ser realizado ao final de cada iteração do projeto.

Interpretação: Essa métrica permite verificar a eficácia das revisões realizadas durante o processo de desenvolvimento, bem como a velocidade com que as ações para resolução dos problemas são tomadas. A diminuição de problemas encontrados a cada revisão indica que a equipe está melhor preparada para o desenvolvimento do projeto. Com essa métrica também poderemos detectar as fases que estão gerando mais problemas para treinarmos a equipe e tratarmos as causas dos problemas.

M-14. Número de não conformidades registradas x Número de não conformidades resolvidas

Definição: Essa métrica irá totalizar as não conformidades registradas durante as auditorias e as não conformidades já resolvidas. Auditoria é o exame de um produto ou processo, realizado por uma entidade independente, que avalia a conformidade com especificações, padrões, acordos contratuais ou outros critérios acordados no projeto, com o objetivo de melhorar a qualidade do software [35].

Procedimento: Para calcular esta métrica iremos totalizar não conformidades encontradas durante as auditorias, bem como não conformidades que já foram solucionadas.

Quando: Esse procedimento deverá ser realizado ao final de cada fase de desenvolvimento do projeto.

Interpretação: Essa métrica permite verificar a eficácia das auditorias realizadas durante o processo de desenvolvimento, bem como a velocidade com que as ações para resolução dos problemas são tomadas. A diminuição de não conformidades encontradas nos projetos a cada auditoria indica que a equipe está melhor preparada no processo de desenvolvimento de software. Com essa métrica também poderemos detectar as fases que estão gerando mais problemas para assim revisarmos os processos dessas fases e tratarmos as causas dos problemas.

Para essas métricas aqui apresentadas, tais como produtividade da equipe em esforço/KLOC, tamanho do software em KLOC, número de bugs/KLOC, cujas medidas envolvem linhas de código (KLOC), a escolha de tal medida deu-se pelo fato de, neste

primeiro momento do trabalho, optarmos pela simplicidade da coleta e do cálculo das métricas.

Dessa forma, por estarmos propondo um conjunto inicial de métricas, e levando em consideração que a grande maioria das empresas ainda não possuem uma cultura de medição, a adoção de métricas orientada a tamanho, tal como linhas de código, torna os processos de coleta e avaliação mais simples e direto. Consideramos que a adoção de métricas orientadas à função, por exemplo pontos-por-função, neste momento envolveria um maior grau de subjetividade no processo, uma vez que são medidas indiretas do software e do processo por meio do qual ele é desenvolvido, pois as métricas orientadas à função concentram-se na complexidade das funcionalidades do sistema. Sendo assim, as organizações que usam esse tipo de métrica precisam desenvolver critérios para determinar se sua entrada particular é simples, média ou complexa, de forma a minimizar ao máximo a subjetividade da medição.

4.4 Implantação de um Programa de Métricas

Conforme apresentado na Seção 4.3, as métricas de software podem ajudar bastante no entendimento e melhoria do desenvolvimento de software, no entanto, implementá-las em uma organização tem sido um grande desafio. Wiegers [22] apresenta algumas recomendações a serem observadas durante a implantação de um programa de métricas:

- O programa de métricas deve estar atrelado aos objetivos gerenciais da organização, para assim contar com o apoio da alta gerência da organização.
- Iniciar com um conjunto pequeno e balanceado de métricas, que permita o melhor entendimento da forma de trabalho da organização e dos objetivos do programa de métricas. A medida que a cultura de mensuração amadureça na organização, este conjunto de métricas irá sendo modificado e expandido. A escolha desse conjunto balanceado de métricas é essencial para o sucesso do programa de métricas.

- Medir aspectos corretos, que irão ajudar nas estratégia da organização. Se os dados coletados não estiverem sendo utilizados ou não estiverem provendo informações que contribuam para uma gerência mais eficiente de projetos e de pessoas, é hora de reavaliar as métricas definidas.
- As métricas devem ser definidas de forma precisa, evitando que estas sejam interpretadas de formas diferentes.
- As métricas não devem ser utilizadas para avaliar indivíduos, nem para motivá-los. O objetivo principal das métricas é obter um entendimento do processo, identificando seus problemas e onde este pode ser melhorado. Utilizar métricas para avaliar ou motivar pode conduzir a comportamentos anormais, que trazem resultados inconsistentes.
- Todos os envolvidos no programa de métricas devem ter um entendimento único sobre os objetivos e expectativas do programa, bem como dos objetivos e definições das métricas, e do seu papel no programa.

Ainda para o sucesso do programa de métricas, recomendamos o uso de ferramentas que auxiliem na coleta, armazenamento e consulta das métricas, minimizando o *overhead* dessas atividades. Para tal, sugerimos que no mínimo a organização disponha de um banco de dados para armazenar e consultar os dados coletados, e que as atividades de coleta e armazenamento dos dados sejam inseridas no processo de desenvolvimento de software, com atividades, procedimentos, responsáveis e *templates* bem definidos para guiarem sua execução como parte inerente ao processo de desenvolvimento de software.

4.5 Framework para Acompanhamento e Estimativas de Projeto de Software através de Métricas

Nesta seção iremos propor algumas modificações no fluxo de Gerenciamento de Projetos do RUP, para que este possa prover um *framework* para acompanhamento e estimativas de projeto de software através de métricas.

O fluxo de Gerenciamento de Projetos do RUP foi influenciado pelo PMBOK – *Project Management Body of Knowledge*, um guia que descreve um subconjunto de conhecimentos e práticas geralmente utilizadas pelos profissionais de gerência de projetos de diversas áreas [26]. O fluxo do RUP utiliza a abordagem de desenvolvimento iterativo e orientado a casos de uso, estando focado nos seguintes aspectos: planejamento do projeto, gerência de riscos, monitoramento do progresso do projeto, utilização de métricas.

Os objetivos deste fluxo de Gerenciamento de Projetos são:

- Prover um *framework* para gerenciar projetos de software.
- Prover um guia prático para planejar, executar e monitorar projetos de software.
- Prover um *framework* para gerenciamento de riscos.

No entanto, esse fluxo não cobre todos os aspectos do gerenciamento de projetos, como por exemplo:

- Gerenciamento de pessoas: contratação/demissão, treinamento, capacitação;
- Gerenciamento de orçamento;
- Gerenciamento de contratos.

O RUP 2000 [32] sugere as atividades apresentadas na Tabela 4.1 para serem realizadas no seu fluxo de Gerenciamento de Projetos:

Atividades de iniciação do projeto	Desenvolver Estudo de Viabilidade do Projeto Identificar Riscos Iniciar Projeto
Atividades de planejamento do desenvolvimento	Definir monitoramento e controle do processo Planejar fases e iterações Desenvolver Plano de desenvolvimento de software Definir organização e equipe do projeto
Atividades de início e final de iteração	Desenvolver plano de iteração Adquirir equipe Iniciar iteração Avaliar Iteração Preparar para finalizar fase

	Preparar para finalizar projeto
Atividades de gerenciamento	Monitorar status do projeto Elaborar cronograma e avaliar trabalho Reportar status Tratar exceções e problemas
Desenvolver planos	Plano de mensuração Plano de gerenciamento de riscos Plano de qualidade Plano de resolução de problemas Plano de aceitação de riscos
Atividades de revisão	Revisar aprovação do projeto Revisar Planejamento do Projeto Revisar Plano de Iteração Revisar critérios de aceitação da iteração Revisar aceite da iteração Revisar ciclos de vida dos *milestones* Revisar aceitação do projeto

Tabela 4.1 Atividades do fluxo de Gerenciamento de Projetos do RUP

Observando o RUP, percebe-se que o mesmo tem um conjunto bem definido de atividades para o planejamento do projeto e tratamento de riscos, no entanto o acompanhamento de projetos e a utilização sistemática de técnicas e métricas para monitorar o desenvolvimento deixam a desejar. O RUP não define um conjunto de tarefas que devem ser realizadas para um acompanhamento preciso do projeto, ele apenas indica a importância de ser realizado o acompanhamento. Ele apresenta um conjunto de diretrizes, que citam alguns indicadores a serem utilizados durante o desenvolvimento, tais como indicadores de tamanho, progresso e da qualidade do produto que está sendo gerado, mas não apresenta um conjunto definido de métricas que possa ser utilizado eficientemente na monitoração do progresso, e visualização do desempenho das equipes durante o desenvolvimento do projeto. O RUP apenas oferece o Plano de Mensuração onde deverão ser documentadas as métricas a serem coletadas, bem como seus procedimentos para coleta. Sentimos falta de uma atividade específica no RUP para realizar a coleta de dados para as métricas de progresso, qualidade, acompanhamento, produtividade e outras métricas que venham a ser definidas pela organização, bem como uma forma para registro e armazenamento desses dados.

Em algumas atividades do RUP, tais como Avaliar Iteração e Monitorar e Controlar Projeto**,** existem passos que sugerem a coleta de métricas primitivas para avaliação do progresso do projeto. Porém, pelo fato da atividade de coleta de métricas ser de extrema importância para as atividades de planejamento e gerenciamento de projetos de software, propomos incluir explicitamente esta atividade no fluxo de Gerenciamento de Projetos do RUP, conforme mostrado na Figura 4.1.

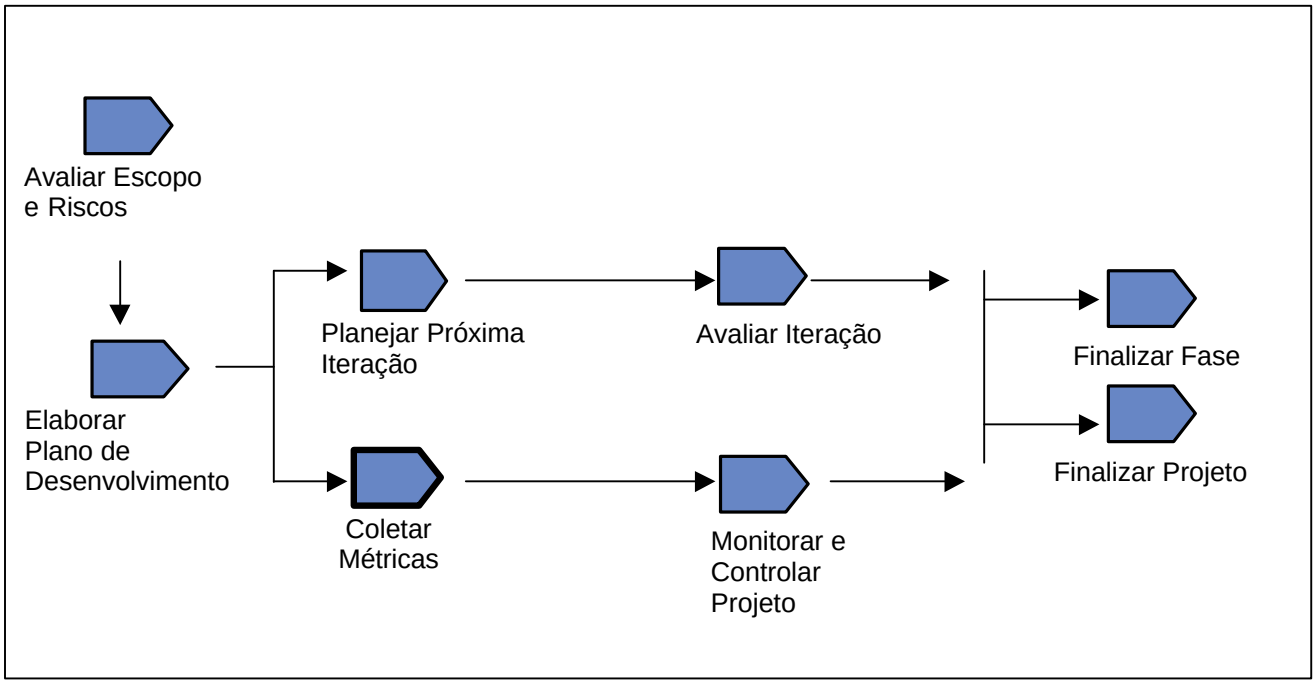

Figura 4.1 – Adaptação do Fluxo de Gerenciamento de Projetos do RUP 2000

O fluxo original do RUP 2000 foca nas atividades de planejamento do projeto, através da atividade Avaliar Escopo e Riscos, seguida da atividade Elaborar Plano de Desenvolvimento. Uma vez elaborado o plano de desenvolvimento do projeto, o RUP sugere que antes do início de cada iteração seja feito o planejamento da mesma através da atividade Planejar Próxima Iteração. Dessa forma, para cada iteração é realizado seu planejamento inicial e sua avaliação final, através da atividade Avaliar Iteração. As iterações são realizadas até que seja finalizada uma fase do desenvolvimento ou o projeto, momento onde serão realizadas as atividades Finalizar Fase ou Finalizar Projeto, respectivamente. Durante toda execução das iterações, até o término das fases do

desenvolvimento, o projeto é acompanhado através da atividade Monitorar e Controlar Projeto, que irá fornecer informações para o planejamento das iterações seguintes e para avaliação da situação atual do projeto. A nova atividade Coletar Métricas, incluída neste fluxo, irá prover informações concretas para o monitoramento do projeto, planejamento e avaliação das interações.

Com esta atividade, propomos a inclusão de mais um artefato no RUP, o Formulário de Métricas, que é saída desta atividade. Este artefato pode ser uma planilha, um formulário HTML ou um banco de dados, com a finalidade de armazenar as métricas primitivas coletadas, bem como as fórmulas para cálculo automático das demais métricas especificadas no Plano de Mensuração, permitindo fácil visualização pelo Gerente de Projeto. Esse artefato será entrada para as atividades que necessitam de coleta de métricas, tais como Avaliar Iteração e Monitorar e Controlar Projeto**,** e estas não terão mais o passo sugerindo a coleta de métricas, apenas consultarão o artefato Formulário de Métricas atualizado. No RUP não há indicação de artefatos (*templates*, ferramentas, planilhas ou formulários) com esse objetivo. No Apêndice A apresentamos um modelo para este formulário. A atividade Coletar Métricas é apresentada a seguir, na Figura 4.2.

<table>
<tr><td colspan="2">Objetivos
- Coletar informações de qualidade, esforço, tamanho, estabilidade e progresso do projeto para prover uma melhor visibilidade do projeto ao Gerente de Projeto.</td></tr>
<tr><td colspan="2">Passos
- Preparar Formulário de Métricas com dados iniciais do projeto (passo realizado uma vez, no início do projeto)
- Coletar dados das iterações
- Coletar dados do Release
- Registrar dados coletados no Formulário de Métricas
- Preencher informações de finalização do projeto no Formulário de Métricas (passo realizado uma única vez, no final do projeto)</td></tr>
<tr><td>Artefatos de entrada</td><td>Artefatos de Saída</td></tr>
</table>

- Plano de Mensuração	- Formulário de Métricas
Responsável: Gerente de Projeto	

Figura 4.2 – Atividade Coletar Métricas

Passo 1: Preparar Formulário de Métricas com dados iniciais do projeto

Após o planejamento inicial do projeto, deverão ser preenchidas as seções 1 e 2 do Formulário de Métricas com as informações de identificação do projeto e do planejamento inicial realizado quanto ao número de casos de uso a ser implementado no projeto e a quantidade de iterações por fase do projeto. Essas informações poderão ser encontradas no Plano de Projeto.

Passo 2: Coletar dados das iterações

Ao início de cada iteração, devem ser coletados os dados do planejamento da iteração quanto ao esforço estimado para cada fluxo de atividades a ser executado na iteração, as datas previstas para os *milestones* e quantidade de casos de uso a serem implementados. Essas informações poderão ser obtidas a partir do artefato Plano da Iteração e do cronograma da iteração.

Durante a iteração e ao seu final, essas mesmas informações deverão ser levantadas a partir do cronograma atualizado com as datas de entrega dos *milestones*, os casos de uso implementados e o esforço real gasto nas atividades de cada fluxo. Além disso, deverão ser levantadas as informações dos bugs encontrados nos testes da iteração, solicitações de mudanças registradas e mudanças realizadas na iteração, problemas encontrados nas revisões e auditorias, bem como número de linhas de código produzidas na iteração.

Passo 3: Coletar dados do *Release*

Sempre que liberado um *release*, deverão ser registradas as solicitações de mudanças e os bugs encontrados após sua liberação.

Passo 4: Registrar dados coletados no Formulário de Métricas

As informações coletadas deverão ser registradas na Seção 3 do Formulário de Métricas (em forma de planilha, ou formulário HTML, ou banco de dados). Esse formulário deverá ser o mais automatizado possível para calcular os desvios e as métricas por iterações e por fases definidas para o projeto, apresentando esses resultados em gráficos e tabelas facilitando a visualização e acompanhamento do projeto.

Passo 5: Preencher informações de finalização do projeto no Formulário de Métricas

Ao final projeto, deverão ser registrados os dados finais do projeto, quanto à quantidade de casos de uso efetivamente implementados, quantidade de iterações realizadas por fase do processo, e o total de linhas e código produzidas. Esses dados serão registrados na Seção 4 do Formulário de Métricas.

4.6 Considerações Gerais

Neste capítulo ressaltamos a importância da definição prévia de metas para facilitar a escolha de métricas e a correta interpretação dos resultados. Neste contexto, adotamos uma abordagem baseada na experiência de Donald McAndrews [33] para definição do conjunto de métricas apresentado neste trabalho. Essa abordagem consiste em identificar o escopo da atividade de medição, ou seja, estabelecer o propósito do programa de métricas, e definir os procedimentos para medição, descrevendo em que consiste cada métrica, os dados que as compõem, a unidade de medida a ser adotada, procedimentos de coleta, formato e mecanismo de armazenamento.

Escolhemos um conjunto de métricas que pudesse prover ao gerente de projeto visibilidade do progresso dos projetos de desenvolvimento de software, permitindo o acompanhamento da execução do projeto segundo o planejamento, identificando problemas no projeto e no processo utilizado e criando uma base histórica para estimativas. Para tal escolha, consideramos uma organização que busca melhorar seu processo de desenvolvimento de software, buscando CMM Nível 2, e que utiliza um

processo baseado no RUP. Dessa forma, os objetivos das métricas escolhidas foram atrelados às metas das KPAs do CMM Nível 2: Gerência de Requisitos, Planejamento de Projeto, Supervisão e Acompanhamento de Projeto de Software e Garantia da Qualidade.

Para o sucesso do programa de métricas devemos, entre outras coisas, criar a cultura de métricas na organização, inserindo as atividades de coleta e armazenamento dos dados como parte do processo de desenvolvimento de software. Sugerimos a definição de atividade, procedimentos, responsáveis e *templates* para guiarem a execução do programa de métricas, inserindo-os como parte inerente ao processo de desenvolvimento de software da organização.

Dessa forma, propusemos uma modificação no fluxo de Gerenciamento de Projetos do RUP, inserindo a atividade Coletar Métricas para que este possa prover um *framework* para acompanhamento e estimativas de projeto de software através de métricas.

Capítulo 5

Aplicação das Métricas em Projetos Reais

Neste capítulo iremos relatar a experiência de uma aplicação das métricas propostas no Capítulo 4 desta dissertação em projetos reais de desenvolvimento de software. As métricas coletadas e avaliadas nesta experiência foram: número de bugs registrados x número de bugs fechados, distribuição de esforço por fase de desenvolvimento, distribuição de esforço por fluxo de atividades e esforço panejado x esforço realizado por iteração. As demais métricas não foram coletadas uma vez que os projetos avaliados não tinham registros de dados que permitissem sua avaliação.

Essa experiência foi realizada em quatro projetos desenvolvidos no Centro de Estudos e Sistemas Avançados do Recife (CESAR), uma organização de natureza não governamental criada no Centro de Informática da Universidade Federal de Pernambuco.
A experiência aqui relatada compreende a coleta de dados, a partir de ferramentas já utilizadas pelos projetos, e a análise das métricas e dos seus resultados dentro do contexto de cada projeto e da própria organização. Para tal, iremos descrever as características do ambiente onde foi realizada a experiência, incluindo a organização e os projetos envolvidos, bem como os processos utilizados para a coleta e interpretação das métricas.
O objetivo desta aplicação é mostrar como essas métricas podem prover informações para uma efetiva gerência de projetos, considerando as diretrizes do CMM Nível 2.

5.1 Ambiente de Trabalho

O CESAR (www.cesar.org.br) é uma organização de natureza não governamental, criada por professores do Centro de Informática da Universidade Federal de Pernambuco para realizar o desenvolvimento de projetos, sistemas e produtos, pesquisa, treinamento e consultoria na área de Tecnologia da Informação, visando o desenvolvimento e inovação tecnológica da sociedade. Em particular, é uma empresa direcionada a impulsionar a interação entre os meios empresarial e acadêmico.

Os projetos desenvolvidos no CESAR adotam como linha-mestra para seu desenvolvimento o ProSCes, processo de desenvolvimento de software definido pelo CESAR, e fortemente baseado no RUP, cujas principais características são um desenvolvimento iterativo e incremental, orientado a objetos, com foco na criação de uma arquitetura robusta, análise de riscos e utilização de casos de uso para o desenvolvimento.

O ProSCes foi fundamentado no modelo CMM e nas normas ISO9000 (em especial a ISO9000-3), e representa uma das iniciativas do projeto de qualidade do CESAR, o TQM21 [37], que busca conduzir o CESAR a se tornar uma organização Nível 2 do CMM. Esta é uma meta do CESAR a ser alcançada até maio do ano de 2003 e foi o fato que motivou a escolha do CESAR para aplicação deste estudo.

Durante a aplicação deste estudo, o CESAR estava estruturando sua Gerência de Qualidade, que tem como responsabilidades manter e evoluir o ProSCes, coletar métricas de projeto e métricas organizacionais, além de realizar auditorias nos projetos. Essa Gerência de Qualidade atuará como o grupo de GQS, definido pelo CMM [1].

Foram escolhidos quatro projetos em desenvolvimento pelo CESAR para aplicação e avaliação das métricas definidas no Capítulo 4 dessa dissertação. A Tabela 5.1 apresenta uma breve descrição desses projetos.

Projeto	Descrição	Iterações	Início	Duração esperada	Casos de Uso
Controle de Acesso	Sistema de controle de Segurança de acesso a sistemas corporativos.	4	07/2001	10 meses	60
Controle de Serviços	Sistema para controle de serviços financeiros oferecidos por cartões de crédito.	4	10/2001	15 meses	143
Call Center	Sistema de Call Center.	4	08/2001	5 meses	15
PEP	Sistema de Prontuário Eletrônico do Paciente.	3	10/2001	8 meses	30

Tabela 5.1 Descrição dos Projetos

Todos esses projetos ainda estavam em desenvolvimento durante a coleta dos dados, que foi realizada no mês de abril de 2002. Especificamente o Controle de Acesso encontrava-se na última iteração do projeto e o PEP estava iniciando sua fase de Construção. O sistema de Controle de Serviços estava na sua fase de Elaboração e o Call Center estava na fase de Construção, após ter sofrido algumas renegociações de escopo. Esses quatro projetos avaliados tiveram seu desenvolvimento em Java, sob o ambiente Web, e possuíam equipes com perfis e experiências semelhantes. A Tabela 5.2 apresenta uma visão das equipes inicialmente planejadas para esses projetos.

Projeto	Equipe	Qtd
Controle de Acesso	Gerente de Projeto Analista de Negócio[7] Engenheiro de Software[8]	1 2 4
Controle de Serviços	Gerente de Projeto Analista de Negócio Engenheiro de Software	1 3 7
Call Center	Gerente de Projeto Analista de Negócio Engenheiro de Software	1 1 1
PEP	Gerente de Projeto Analista de Negócio Engenheiro de Software	1 2 4

Tabela 5.2 Equipe dos Projetos

É importante ressaltar que os projetos Controle de Acesso, Controle de Serviços e Call Center tinham o Gerente de Projeto compartilhado.

Além disso, esses projetos utilizaram o Timesheet como ferramenta para registro das horas despendidas nas suas atividades e o Bugzilla [37] como ferramenta para registro e acompanhamento dos bugs encontrados nos testes.

O Timesheet é uma ferramenta desenvolvida pelo próprio CESAR, onde estão cadastradas todas as atividades de desenvolvimento do projeto, segundo o ProSCes [37], para que sejam reportados os esforços despendidos pela equipe em cada uma dessas atividades.

O Bugzilla é uma ferramenta para gerenciamento de mudanças [37], e foi desenvolvido para a registro e acompanhamento de bugs do projeto Mozilla, um browser web de código aberto, desenvolvido pela organização Mozilla (mozilla.org). As mudanças aqui

[7] Responsável pelo levantamento e especificação dos requisitos do sistema, bem como definição do modelo de análise.
[8] Responsável pela codificação do sistema.

referenciadas podem ser novas funcionalidades, defeitos encontrados, pedidos de melhoria ou pedidos de mudança, solicitadas por qualquer pessoa envolvida no projeto. Ao registrar uma solicitação de mudanças podemos definir sua prioridade, e manter seu status atualizado (aberta, fechada, reaberta, validada, e outros).

5.2 Coleta e Interpretação das Métricas

Para aplicação desse estudo foram coletadas métricas de qualidade, esforço e estabilidade. Os dados foram coletados a partir dos registros dos projetos realizados nas ferramentas Timesheet e Bugzilla. Analisaremos a seguir os resultados obtidos.

5.2.1 Métricas de Qualidade

Para cada projeto em estudo, foram totalizados os bugs em aberto e os bugs já resolvidos, com o intuito de obter a métrica de bugs registrados x bugs fechados (M-12). Os projetos analisados foram planejados em iterações muito longas, com mais de dois meses, dessa forma a totalização dos bugs foi realizada em períodos variando de duas a quatro semanas, e não ao final de cada iteração como proposta na definição dessa métrica.

Analisando os gráficos das Figuras 5.1, 5.2, 5.3 e 5.4, o gerente de projeto terá as seguintes visões:

- Total de bugs reportados
- Total de bugs em aberto
- Taxa em que os bugs são resolvidos

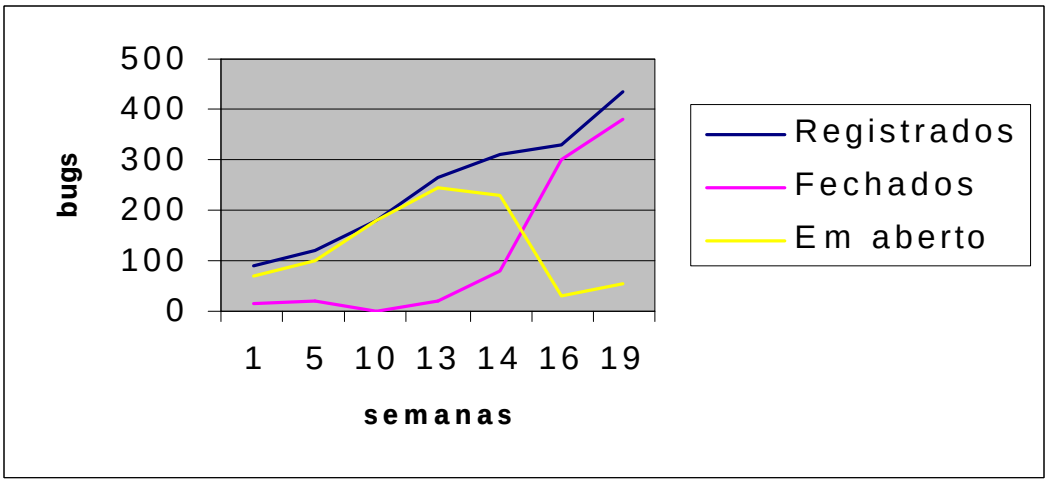

Figura 5.1 – Bugs x Semana (Controle de Acesso)

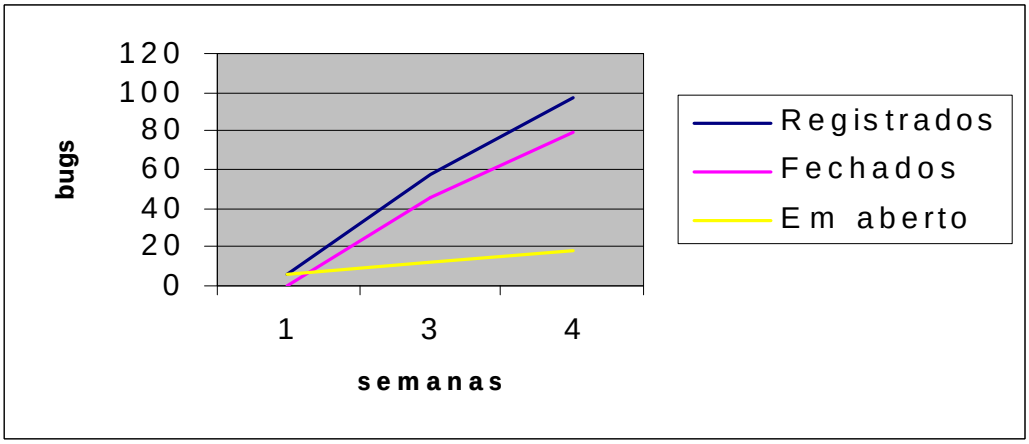

Figura 5.2 - Bugs x Semana (PEP)

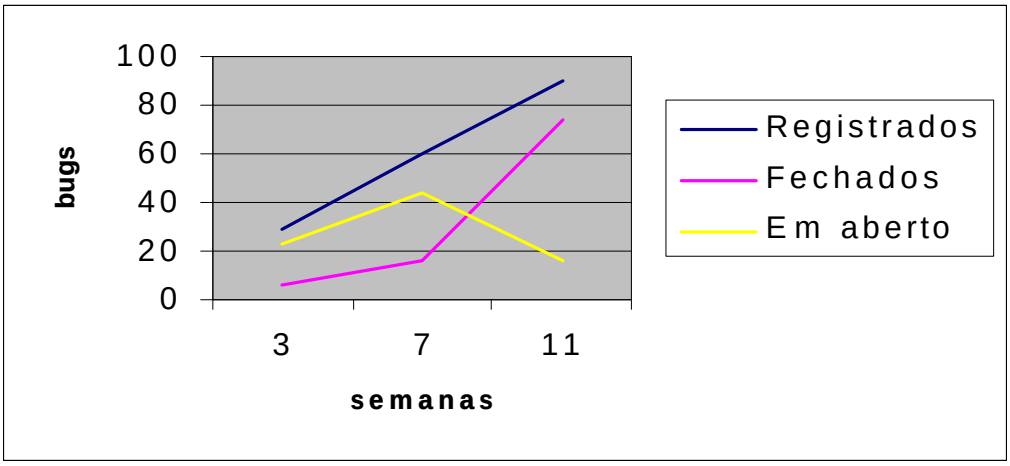

Figura 5.3 – Bugs x Semana (Call Center)

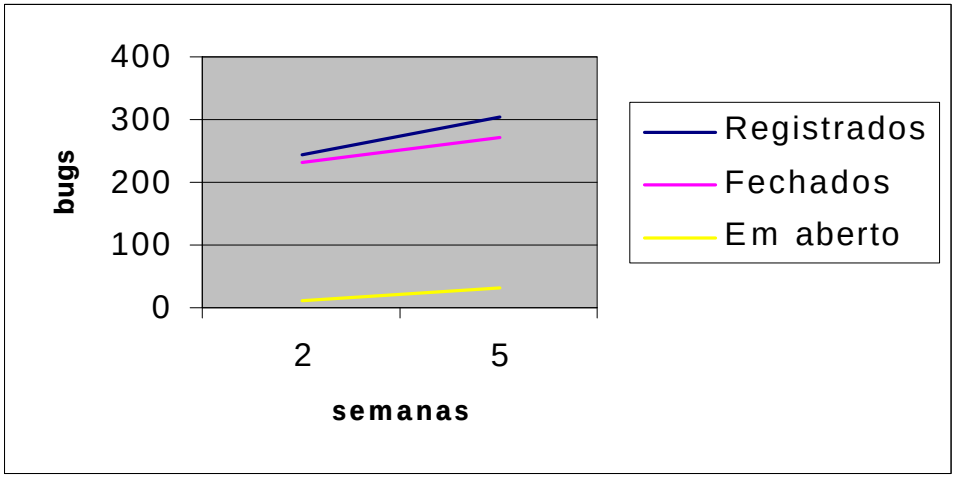

Figura 5.4 - Bugs x Semana (Serviços)

Os gráficos mostram uma tendência de aumento linear do número de bugs registrados na linha do tempo. Considerando este dado, durante o planejamento do projeto o gerente poderá estimar o número de bugs a ser registrado, baseado em dados históricos, e estimar custos e cronograma de forma mais consistente. Se o número bugs registrados no projeto exceder o estimado, outras estimativas que consideraram este dado, tais como custo e cronogrma, deverão ser revistas.

O número de bugs registrado reflete o esforço de retrabalho que deverá ser despendido para sua correção antes do produto ser liberado para o cliente. Se este esforço de retrabalho não for previsto no planejamento do projeto, ocasionará impacto no custo e cronograma do projeto.

Idealmente, o número de bugs em aberto deve permanecer próximo de zero. Se este número desviar consideravelmente de zero, ou continuar crescendo na linha do tempo, o gerente de projeto precisará reavaliar a alocação dos seus recursos, priorizando as atividades de resolução dos bugs.

Nos gráficos das figuras 5.2 e 5.4, identificamos uma estabilidade do número de bugs em aberto, permanecendo próximo a zero. Já nos gráficos das figuras 5.1 e 5.3, visualizamos

uma tendência ao crescimento do número de bugs em aberto no início do projeto, e só após um dado período o gráfico apresenta uma redução desse número.

Um outro aspecto importante que podemos avaliar nesses gráficos é a rapidez com que os bugs estão sendo resolvidos. Nos gráficos das figuras 5.2 e 5.4, verificamos que a taxa de resolução de bugs evolui quase linearmente, acompanhando o crescimento do número de bugs registrados. Dessa forma, consegue-se manter a taxa de bugs em aberto sempre próximo a zero. Nos gráficos das figuras 5.1 e 5.3, a taxa de resolução de bugs apresenta-se inicialmente muito baixa, o que leva ao crescimento do número de bugs em aberto. Após algumas semanas o crescimento do número de bugs fechados foi maior que o número de bugs registrados, levando então, a redução do número de bugs em aberto. Para o projeto da Figura 5.1, esse fato se deu próximo a liberação do *release* para o cliente, pois foi quando os analistas do projeto voltaram a testar o sistema, verificando que os bugs que haviam sido registrados já estavam fechados, e então realizaram o registro de fechamento de bugs na ferramenta. No entanto, a equipe de desenvolvimento já havia trabalhado no fechamento desses bugs durante o desenvolvimento do *release*, apenas os analistas optaram por realizar os testes novamente perto da entrega do *release*.

No caso do projeto da Figura 5.3, tivemos uma situação um pouco diferente. O cliente solicitou fazer parte dos testes do sistema, ainda que este não estivesse finalizado. Dessa forma, após a realização dos testes de final da iteração, o sistema foi instalado para o cliente, e este passou a registrar na ferramenta os bugs encotrados. No entanto, a equipe prosseguiu normalmente com o desenvolvimento, e apenas na fase de testes da iteração seguinte é que foram corrigidos os bugs registrados pelo cliente.

Idealmente, a taxa de registro de novos bugs deve decrescer com o progresso dos testes, isto é, a curva que representa o número de bugs registrados deve diminuir seu crescimento com o passar do tempo. Isso porque, com o progresso dos testes a descobertas de novos deveria ocorrer com menor frequência, uma vez que os bugs mais frequentes devem ser descobertos nos testes iniciais. Se o número de bugs registrados não

diminuir na linha do tempo, pode estar comprometendo a qualidade do produto ou pode estar havendo problema na estratégia de testes. É importante identificar, através de dados históricos, em que fase do projeto esta diminuição tende a ocorrer para assim podermos avaliar melhor a situação do projeto e sua tendência.

5.2.2 Métricas de Esforço

Determinar o esforço requerido de cada perfil necessário ao projeto, e em que momento exato do projeto cada perfil precisará estar disponível é uma importante função do gerente de projeto, e garante um melhor planejamento dos custos e cronograma do projeto. Os indicadores de esforço permitem ao gerente de projeto ter essa visão, bem como acompanhar os esforços despendidos pela equipe, comparando-os ao planejado.

A seguir, comentaremos algumas métricas que oferecem tal visão ao gerente de projeto.

5.2.2.1 Distribuição do esforço por fase de desenvolvimento (M-4)

Para essa métrica, foram totalizados os esforços planejados para cada fase de desenvolvimento, bem como a distribuição desses esforços por fluxo de atividades, em cada fase de desenvolvimento.

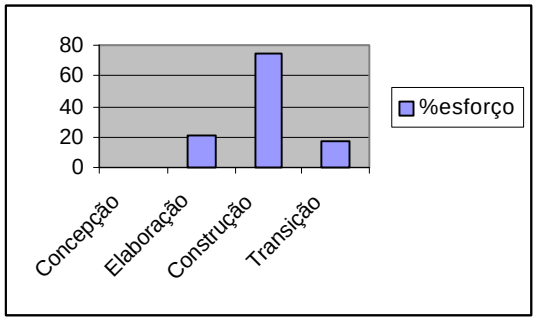

Figura 5.5 – Esforço x Fase (Call Center)

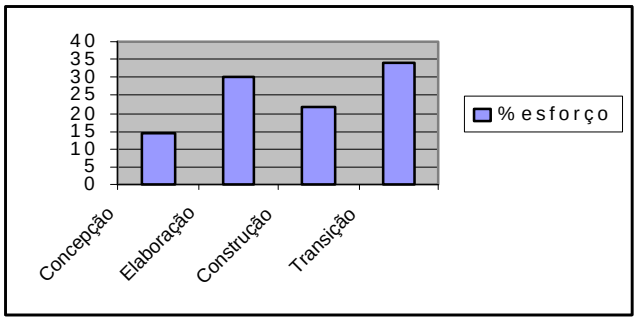

Figura 5.6 – Esforço x Fase (Controle de Acesso)

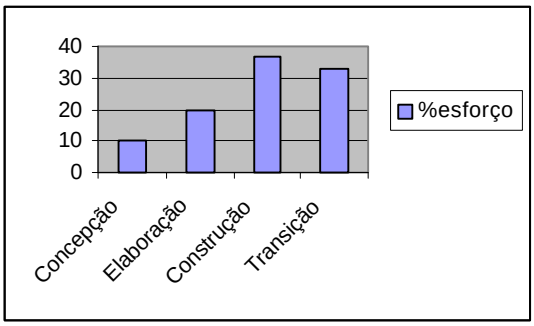

Figura 5.7 – Esforço x Fase (Serviços)

O projeto PEP não foi planejado dividido em fases, uma vez que tinha apenas três iterações, por isso não foi analisado quanto essa métrica. Analisando os demais projetos nos gráficos das Figuras 5.5, 5.6 e 5.7, identificamos que não existe uma tendência clara desses projetos quanto a distribuição de esforços entre as fases. O RUP 2000 sugere uma distribuição para projetos de médio porte segundo a Tabela 5.3 [32].

	Concepção	Elaboração	Construção	Transição
Esforço	5 %	20 %	65 %	10%

Tabela 5.3 Distribuição de esforço por fase [32]

Como a organização está no início do seu programa de métricas, não possuindo ainda dados históricos para avaliar suas tendências, tomamos como base os dados sugeridos pelo RUP.

No projeto Call Center (Figura 5.5), não foi realizada a fase de Concepção, uma vez que os requisitos já haviam sido levantados e especificados antes da contratação do projeto. As demais fases apresentam uma distribuição de esforço semelhante à sugerida pelo RUP. Já nos projetos de Controle de Acesso (Figura 5.6) e Serviços (Figura 5.7) existe uma grande concentração de esforços na fase de Transição. Este fato é justificado devido a forma como foram planejados esses projetos. A maior parte do esforço de implementação do sistema, bem como dos testes do *release* final foram despendidos nesta fase do desenvolvimento.

Analisando as iterações da fase de Transição desses dois sistemas, onde cada iteração está com uma duração média de 4 meses, identificamos que parte do esforço despendido na

finalização da construção do sistema durante esta fase poderia ter sido distribuída nas iterações da fase de Construção, deixando o planejamento dos sistemas mais consistente com a proposta do processo de desenvolvimento utilizado, evitando que fosse despendido muito esforço ainda em atividades de construção na fase em que o sistema deve está sendo preparado para implantação.

Para termos um melhor conhecimento quanto a necessidade de recursos ao longo do projeto, além da distribuição do esforço por fase do projeto, precisamos conhecer a distribuição do esforço nas atividades de cada fase, para assim identificarmos os perfis necessários.

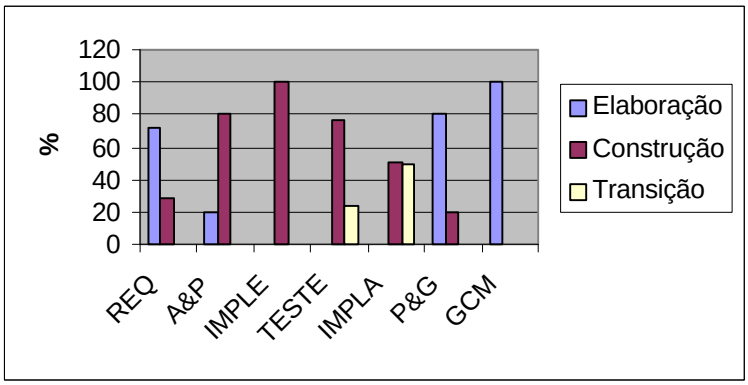

Figura 5.8 – Fluxo x Fase (Call Center)

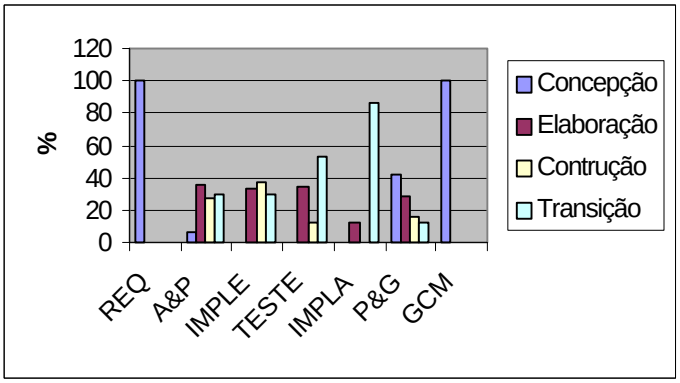

Figura 5.9 – Fluxo x Fase (Controle de Acesso)

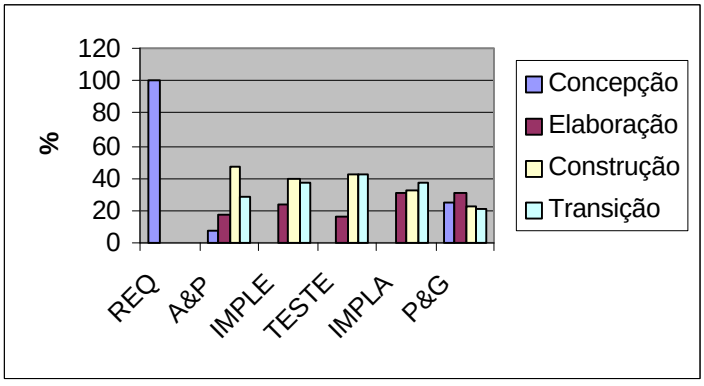

Figura 5.10 – Fluxo x Fase (Serviços)

Analisando os gráficos das Figuras 5.8, 5.9 e 5.10, identificamos uma concentração maior das atividades de planejamento e gerenciamento de projeto (P&G) e requisitos (REQ) na fase inicial dos projetos, ou seja, na Concepção. No caso do projeto Call Center (Figura 5.8), essa concentração deu-se na fase de Elaboração, considerando o fato do projeto não possuir fase de Concepção.

Nos gráficos das Figuras 5.9 e 5.10, identificamos uma distribuição relativamente homogênea das outras atividades nas demais fases do projeto. Esse fato é justificado pela forma como foram planejados esses projetos. Os requisitos foram levantados e especificados na fase inicial do projeto, e a partir daí, foram agrupados em módulos a serem desenvolvidos e implantados em cada uma das demais fases do projeto. Dessa forma, nas fases seguintes (Elaboração, Construção e Transição), foram realizadas atividades de análise e projeto, implementação, testes e implantação, para cada um dos módulos definidos. Apenas na fase de Construção do projeto de Controle de Acesso não foram realizadas atividades de implantação.

No gráfico da Figura 5.8, identificamos uma tendência ao desenvolvimento em cascata, onde a maior parte das atividades de requisitos e planejamento são realizadas na fase inicial do projeto (neste projeto específico, na fase de Elaboração), na fase de Construção são realizadas as atividades de análise e projeto, implementação e testes, sendo realizada toda implementação do projeto durante esta fase, e por fim, na fase de Transição são finalizados os testes e realizadas as atividades de implantação.

Com essa análise constatamos que os projetos avaliados não foram planejados considerando os conceitos de fases e iterações do RUP. Dessa forma, não foi possível identificar a tendência de distribuição de esforço das atividades em cada fase de desenvolvimento, para assim podermos identificar as reais necessidades dos perfis por fases. Essa constatação fica ainda mais clara quando comparamos os gráficos das Figuras 5.8, 5.9 e 5.10 com o gráfico de Baleias do RUP, da Figura 5.11.

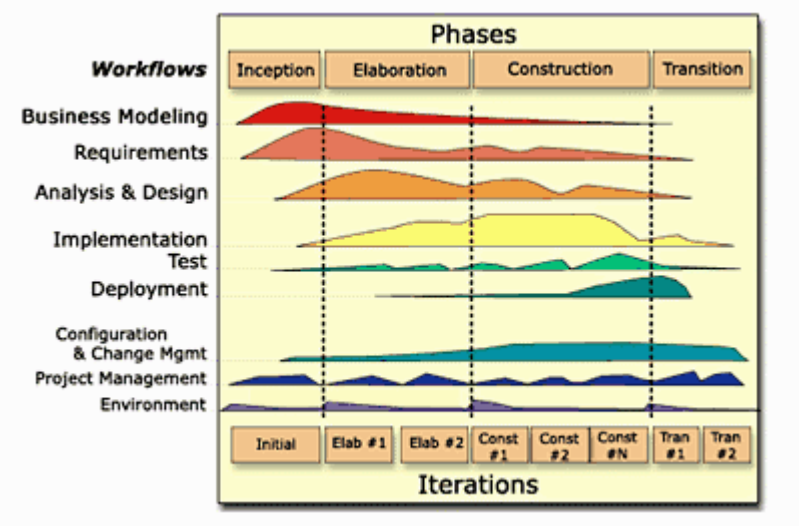

Figura 5.11 – Fluxos e Fases do RUP

O gráfico do RUP, Figura 5.11, apresenta uma tendência da distribuição de esforço dos fluxos de atividades nas diversas fase de desenvolvimento, de acordo com os objetivos de cada fase. As atividades de requisitos, por exemplo, tendem a uma maior concentração na fase de Concepção, enquanto que na fase de Elaboração apresenta uma maior concentração de atividades de análise e projeto. Já na fase de Construção as atividades de implementação e testes são as que aparecem com maior intensidade, da mesma forma que as atividades de implantação são mais intensas na fase de Transição. As atividades de planejamento e gerenciamento de projeto apresentam-se uniformemente distribuídas em todas as fases do desenvolvimento.

Nos gráficos das Figuras 5.9 e 5.10 claramente observamos uma concentração de esforço das atividades de requisitos na fase de Concepção. No entanto, nas demais fases a distribuição de esforço dos fluxos apresentam-se uniformes, não havendo uma tendência clara a uma maior concentração de atividades específicas para uma dada fase do desenvolvimento, conforme sugerido pelo RUP.

Analisando a Figura 5.8 verificamos que todas as atividades referentes aos fluxos de planejamento e gerenciamento, requisitos e análise e projeto foram realizadas quase que inteiramente na fase de Elaboração, finalizando na fase de Construção. Ainda na fase de

Construção foi realizada toda implementação do sistema e inicializadas as atividades de testes, que foram concluídas na fase de Transição. Com essa análise, mais uma vez caracteriza-se o desenvolvimento em cascata na execução deste projeto.

5.2.2.2 Distribuição do esforço por fluxo de atividades (M-3)

Foram totalizados os esforços planejados para cada fluxo de atividades do projeto, independente das fases de desenvolvimento.

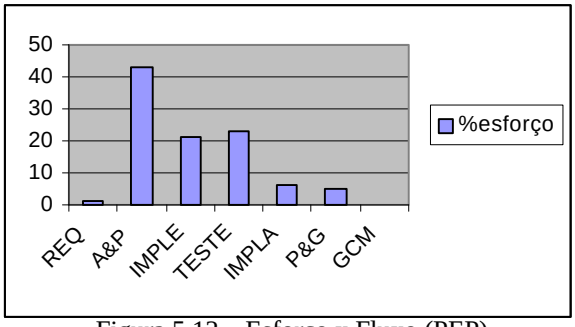

Figura 5.12 – Esforço x Fluxo (PEP)

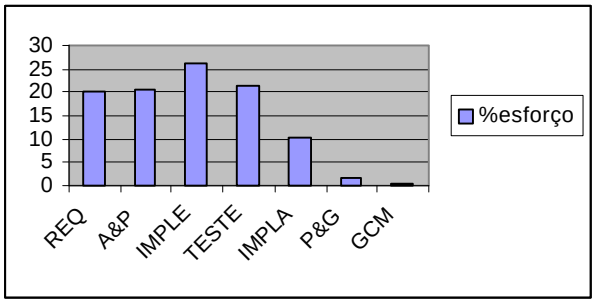

Figura 5.13 – Esforço x Fluxo (Call Center)

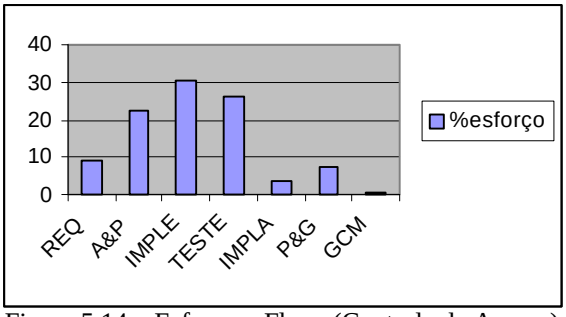

Figura 5.14 – Esforço x Fluxo (Controle de Acesso)

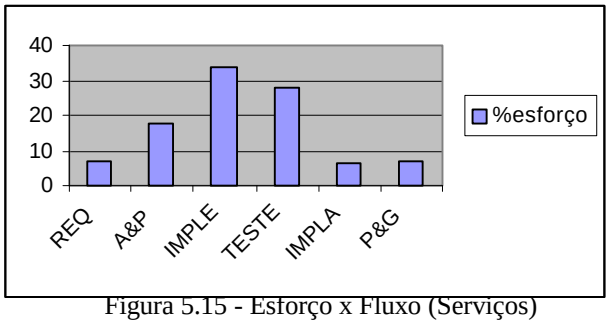

Figura 5.15 - Esforço x Fluxo (Serviços)

Analisando os gráficos das Figuras 5.12. 5.13, 5.14 e 5.15, verificamos que as atividades que demandam mais esforço são as atividades de implementação, testes e análise e projeto, em ordem respectivamente decrescente de esforço.

Identificamos que foi estimado pouco esforço para as atividades de planejamento e gerenciamento de projeto. Durante o desenvolvimento desses projetos avaliamos se o esforço estimado para essas atividades foi subestimado, ou realmente as atividades de

gerenciamento e acompanhamento não demandam muito esforço em relação as demais atividades.

A identificação dos percentuais de esforço comumente gastos nas atividades do projeto é um importante dado para o planejamento e acompanhamento das estimativas de esforço. Uma vez conhecendo o percentual de esforço gasto em cada atividade, e tendo uma estimativa inicial do esforço gasto em pelo menos uma dessas atividades, como por exemplo o esforço gasto para implementação, poderemos facilmente derivar os esforços necessários para as demais atividades. Da mesma forma, sabendo o esforço já gasto em uma atividade, poderemos controlar melhor os esforços que ainda poderemos gastar.

Para alocação dos recursos considerando os perfis necessários para cada atividade, precisaríamos ter a distribuição dos esforços dessas atividades ao longo das fases do projeto. Conseguiríamos obter essa informação através da distribuição do esforço dos fluxos por fase de desenvolvimento, representada nos gráficos das Figuras 5.8, 5.9 e 5.10.

5.2.2.3 Esforço planejado x realizado por iteração (M-9)

Totalizamos os esforços (em homem/hora) despendidos nas atividades de cada fluxo, comparando-os com o esforço planejado para todo projeto.

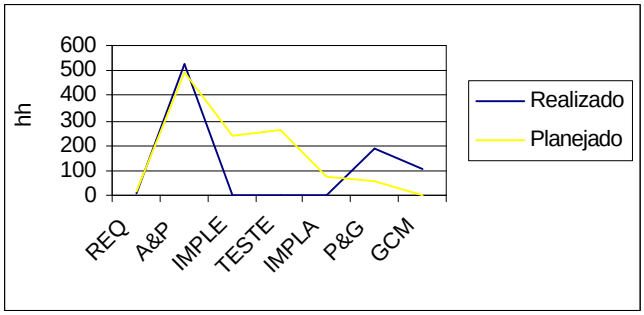

Figura 5.16 – Esforço Realizado (PEP)

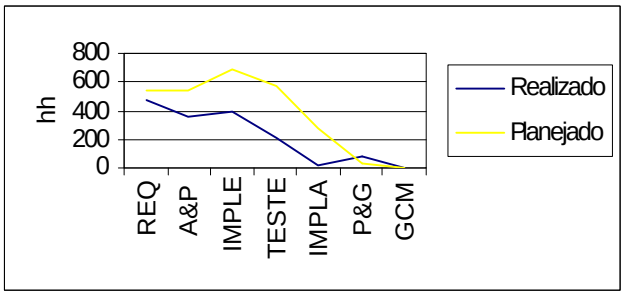

Figura 5.17 – Esforço Realizado (Call Center)

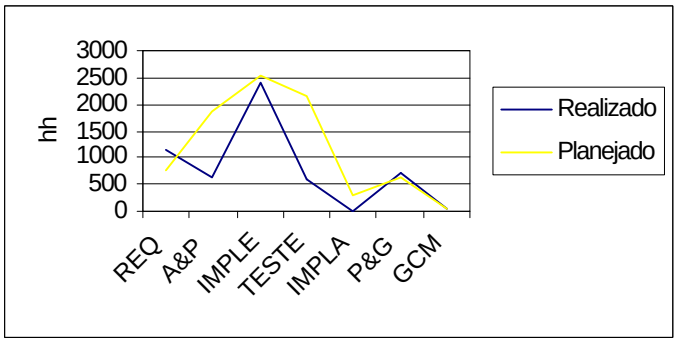

Figura 5.18 – Esforço Realizado (Controle de Acesso)

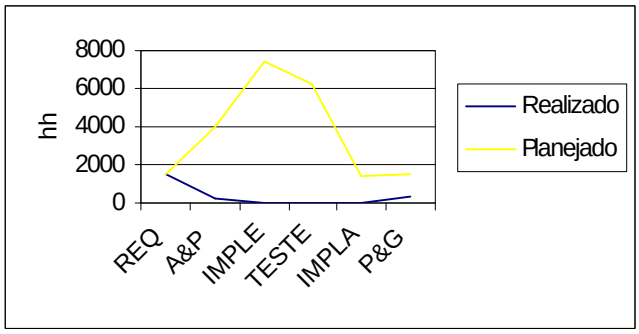

Figura 5.19 – Esforço Realizado (Serviços)

Os gráficos das Figuras 5.16, 5.17, 5.18 e 5.19 mostram o esforço planejado para todo o projeto, e permitem o acompanhamento da utilização desse esforço para cada atividade, até o momento da coleta dos dados.

O gerente pode esperar desvios entre o esforço planejado e o realizado durante a execução do projeto, no entanto, grandes desvios ou períodos extensos onde a equipe esteja sub ou super alocada podem requerer uma análise do problema.
Longos períodos onde a equipe alocada esteja ociosa ou sub-utilizada pode ser resultado de alguns fatores tais como [35]:

- Tamanho do software ter sido super estimado, e neste caso a equipe está sendo muito grande para o trabalho.
- Falta de entendimento dos requisitos, onde a equipe pode está esperando por definições dos requisitos, ou está trabalhando sem todas as informações necessárias, ou ainda pode está trabalhando com um entendimento errado dos requisitos, mais simplificado.
- Time muito produtivo.
- Produto desenvolvido com baixa qualidade.

Já uma situação onde a equipe encontra-se sobrecarregada pode ser resultado dos seguintes fatores [35]:

- Problema mais complexo do que o esperado.

- Requisitos instáveis, que causam retrabalho extensivo.
- Equipe com perfil inapropriado para o trabalho.
- Número de registros de bugs maior que o esperado, causando mais trabalho para equipe.
- Tamanho do software sub-estimado, e neste caso a equipe está sendo pequena para o trabalho.

Uma vez planejado o esforço das atividades do projeto, o gerente do projeto poderá determinar a partir dos dados históricos, se a alocação planejada está apropriada. Existindo grandes desvios, o gerente do projeto deverá decidir se a alocação planejada está incorreta ou se o projeto tem características específicas que justificam os desvios.

Quando avaliamos os esforços já despendidos no projeto, para termos uma visão mais concreta da situação do projeto, é importante avaliarmos também os indicadores de progresso. Com isso poderemos identificar algumas situações tais como:

1. O esforço despendido é maior que o planejado, e o progresso do desenvolvimento do produto também é superior ao planejado. Nesse caso, uma vez seguindo esta tendênca, o projeto irá terminar antes do esperado, dentro dos esforço planejado.
2. O esforço despendido é menor que o planejado, mas o progresso do desenvolvimento do produto é superior ao planejado. Nesse caso, o esforço requerido para as atividades é inferior ao planejado, e o projeto irá terminar antes do previsto, com menos esforço que o planejado.
3. O esforço despendido é maior que o planejado, mas o progresso do desenvolvimento do produto é inferior ao planejado. Nesse caso, mais esforço é requerido para realizar as atividades, e o projeto irá terminar depois do prazo previsto, com mais esforço gasto que o planejado.
4. O esforço despendido é menor que o planejado e o progresso do desenvolvimento do produto é inferior ao planejado. Uma vez seguindo esta tendência, o projeto irá

terminar depois do prazo previsto, porém poderá utilizar a mesma quantidade de esforço prevista no planejamento.

Analisando os gráficos das Figuras 5.16, 5.17 , 5.18 e 5.19 podemos verificar que os esforços já realizados para a maioria das atividades estam abaixo do planejado, porém, como não temos os indicadores de progresso dos projetos, não podemo avaliar qual a tendência do projeto, segundo as quatro situações citadas acima. No entanto, já pode ser identificado que os esforços previstos para as atividades de planejamento e gerenciamento de projetos foram sub-estimados.

5.2.3 Estabilidade – Tamanho (M-6)

Essa métrica foi coletada inicialmente em termos de número de casos de uso, sendo levantado no início de cada projeto o número de casos de uso a ser implementado e o esforço estimado para sua implementação.

	Casos de Uso	**Esforço (hh)**
Controle de Acesso	60	8304
Serviço	143	22210
Call Center	15	2486
PEP	32	1140

Tabela 5.4 Casos de Uso x Esforço

Verificamos que apenas com a quantidade de casos de uso a ser implementados não conseguimos ter uma noção real do tamanho do software, nem conseguimos dimensionar o esforço necessário para seu desenvolvimento. Esse fato é demonstrado quando analisamos os projetos Call Center e PEP. Ambos estão sendo desenvolvidos utilizando a mesma tecnologia, no entanto o Call Center precisará o dobro de esforço para desenvolver 15 casos de uso, enquanto que foi estimado a metade do esforço para o PEP desenvolver 32 casos de uso. Isso se dá devido às diferentes complexidades de implementação de cada caso de uso, bem como às maneiras distintas de especificar cada caso de uso, variando a granularidade e o nível de detalhamento da especificação. Dessa

forma, apenas a quantidade de casos de uso não fornece uma idéia representativa de tamanho do software, nem permite o acompanhamento da variação de tamanho do mesmo em relação a estimativa inicial quando surgir novos casos uso durante o desenvolvimento do projeto.

Vem sendo realizado um trabalho no CESAR para definir as complexidades dos casos de uso: simples, médio, complexo e muito complexo, e a relação entre estes, de forma que possamos dimensionar o software em termos de pontos de complexidade, seguindo os princípios do método de pontos de casos de uso. Pontos de casos de uso é um método de estimativa influenciado pelo método de pontos-por-função, e baseia-se no número de transações existentes em cada caso de uso do sistema para categorizá-los como casos de uso simples, médio ou complexo [36]. As transações aqui referidas são eventos que ocorrem entre os atores e o sistema. Esse método também leva em consideração a complexidade de implementação da interface entre o ator e o sistema, categorizando também os atores em simples, médio ou complexo.

Atualmente, os estudos realizados no CESAR chegaram a seguinte relação entre as categorias dos casos de uso:
1 caso de uso médio = 1,8 casos de uso simples
1 caso de uso complexo = 2,8 casos de uso simples
1 caso de uso muito complexo = 4,1 casos de uso simples

Isto significa que um sistema com 10 casos de uso simples e 5 complexos, terá 30,5 pontos de complexidade e um outro sistema com 5 casos de uso médios e 5 complexos terá 29,5 pontos de complexidade, ou seja, ambos os sistemas terão praticamente o mesmo ponto de complexidade e com isso precisarão do mesmo esforço para serem desenvolvidos.

Esse estudo é realizado considerando as caracterísiticas dos projetos, tecnologias utilizadas e o processo de desenvolvimento de software do CESAR.

Com a conclusão desse estudo, recomendamos que o tamanho do software passe a ser estimado em pontos de complexidade de caso de uso, e não em número de casos de uso.

A Tabela 5.5 apresenta um resumo das interpretações aqui realizadas, relacionando as métricas coletadas às KPAs do CMM 2.

Métricas	**Interpretações**	**KPAs**
(M-12) Bugs registrados x Bugs fechados	• Total de bugs reportados no período. • Total de bugs em aberto. • Taxa em que os bugs são resolvidos. • Taxa em que os bugs são abertos ao longo do projeto.	Supervisão e Acompanhamento de Projeto de Software Gerência de Configuração de Software Garantia da Qualidade de Software
(M-4) Distribuição do esforço por fase de desenvolvimento	• Percentual de esforço despendido por fase de desenvolvimento. • Percentual de esforço despendidos por atividades, em cada fase de desenvolvimento. • Perfis requeridos ao longo das fases de desenvolvimento.	Planejamento de Projeto Supervisão e Acompanhamento de Projeto de Software Gerência de Contrato de Software
(M-3) Distribuição do esforço por fluxo de atividades	• Percentual de esforço despendidos por atividade. • Perfis requeridos no projeto.	Planejamento de Projeto Supervisão e Acompanhamento de Projeto de Software Gerência de Contrato de Software
(M-9) Esforço planejado x realizado por iteração	• Situações de super-alocaçaõ de recursos. • Situações de sub-alocação de recursos. • Situação do cronograma do projeto (esforço requerido x progresso do projeto)	Planejamento de Projeto Supervisão e Acompanhamento de Projeto de Software Gerência de Contrato de Software

(M-6) Tamanho do Software	• Variação do tamanho estimado do software e tamanho atual. • Variação do tamanho estimado do release e tamanho atual. • Tendência da variação do tamanho do software.	Planejamento de Projeto Supervisão e Acompanhamento de Projeto de Software Gerência de Configuração de Software Gerência de Contrato de Software

Tabela 5.5 Interpretação das métricas

5.3 Considerações sobre os Resultados da Experiência

Durante a experiência relatada neste capítulo foram coletadas apenas métricas de qualidade, esforço e tamanho. Das métricas de qualidade, coletamos apenas as métricas de testes (bugs), uma vez que os processos de revisão e auditoria ainda não haviam sido implantados na organização.

5.3.1 Ferramentas Utilizadas

O uso das ferramentas Timesheet e Bugzilla nos projetos se mostraram essenciais para permitir a coleta das métricas. A partir dessas ferramentas, podemos extrair relatórios que facilitam bastante a coleta dos dados.

Com o Timesheet foi possível extrair os esforços gastos em cada atividade do ProSCes, bem como os esforços gastos por cada perfil do projeto nessas atividades. Os esforços planejados para cada atividade e para cada perfil foram extraídos dos cronogramas dos projetos, elaborados na ferramenta Project. Além disso, podemos realizar nessa ferramenta o acompanhamento dos *milestones* e das atividades previstas e realizadas.

A partir do Bugzilla foi possível extrair relatórios dos bugs registrados em um período, bugs em aberto e bugs fechados em um dado período. Também é possível controlar os bugs abertos e fechados de acordo com sua severidade, de forma que o gerente de projeto poderá priorizar a resolução dos bugs mais críticos para o projeto, bem como controlar

em que fase esses bugs aparecem com maior freqüência, coletando outras métricas de acordo com suas necessidades.

No Bugzilla também são registradas as solicitações de mudanças de requisitos, bem como os bugs encontrados nos *releases* que estão nos clientes, de forma que podem ser extraídas informações sobre as requisições realizadas, em aberto e as já concluídas.

As ferramentas utilizadas para coleta dos dados se mostraram satisfatórias. No entanto, o uso de ferramentas de testes, podendo ser integradas a ferramentas de gerência de configuração e controle de mudanças, poderá automatizar ainda mais a coleta de dados, bem como deixar a corretude dos dados mais independentes dos técnicos que realizam os testes, uma vez que as próprias ferramentas estariam registrando os erros e as mudanças realizadas.

5.3.2 Registro e Coleta dos Dados

As informações do Timesheet e Bugzilla são cadastradas pelos próprios membros da equipe do projeto. Dessa forma, ressaltamos a importância de toda equipe do projeto estar envolvida e consciente da importância do programa de métricas para a organização, pois todas essas informações extraídas das ferramentas só serão realmente válidas se a equipe estiver cadastrando os dados de forma correta.

O sucesso dos planos de métricas torna-se muito mais difícil de ser alcançado quando são requeridos trabalhos e computações manuais. Este fato é particularmente mais preocupante nos processos de desenvolvimento de software iterativo e incremental, que exigem a coleta das métricas em intervalos regulares. Além disso, devido ao fato de utilizar-se as métricas como forma de prevenção, e não apenas como forma de reação aos problemas detectados, o ideal é que a qualquer momento do projeto, o gerente possa obter a situação atual do mesmo a partir da consulta às métricas, de forma simples e direta.

Para tal, já existem ferramentas e componentes que podem ser integrados às ferramentas de modelagem e plataformas de desenvolvimento para coletarem dados do projeto automaticamente e apresentarem os resultados para análise dos gerentes [5]. Também encontramos ferramentas que permitem a análise e revisão dos dados coletados do projeto, incluindo alertas para dados suspeitos ou questionáveis, bem como comparações entre o atual comportamento dos dados reportados e do planejado [39]. No entanto, na ausência de ferramentas pró-ativas como essas, é importante que o gerente esteja frequentemente acompanhando as métricas do projeto, em intervalos tão curtos quanto necessite o projeto. Sob esta visão, alertamos para o fato que as métricas a serem coletadas por iteração devem considerar a duração da mesma. A coleta de métricas por iterações, quando estas são muito longas, tais como as iterações dos projetos avaliados neste Capítulo, podem revelar tardiamente os problemas do projeto. Dessa forma, o mais adequado é que seja avaliado o nível de controle que se deseja ter sobre o projeto, e assim definir o intervalo de coleta e avaliação das métricas dentro de cada iteração.

Recomendamos fortemente o uso de formulários em banco de dados ou planilhas, tal como o indicado no Apêndice A, para facilitar o registro, visualização e análise das métricas, principalmente nos casos em que a organização não dispuser de ferramentas para coleta e análise das métricas. Especificamente neste trabalho, os dados foram cadastrados em formulários no banco de dados Access e os gráficos das métricas gerados a partir de planilhas Excel.

Durante o trabalho de coleta de dados, foi identificado que os conceitos de fases do desenvolvimento de software segundo o RUP e iterações não estavam muito claros para as equipes dos projetos. Por esse motivo, os projetos não foram bem planejados de acordo com esses conceitos e não pudemos realizar as coletas dos dados por iterações, conforme indicado nas descrições das métricas.

5.3.3 Contribuições para os Projetos

As métricas aplicadas aos projetos utilizados como experimento para esse trabalho contribuíram para o acompanhamento dos projetos de software, dando uma ampla visão

ao gerente sobre como encontra-se o projeto no âmbito de garantia da qualidade, acompanhamento do progresso, custo, utilização dos recursos e esforços planejados. Com essa visão, o gerente do projeto poderá monitorar melhor os projetos, antecipando-se aos problemas que poderão ocorrer, ou identificando-os mais cedo, o que possibilita a realização de um tratamento que não cause grandes impactos ao andamento do projeto.

A métrica M-12 (Bugs registrados x Bugs fechados) por exemplo, apresentou o crescimento do número de bugs registrados ao longo do projeto comparado com a taxa de resolução dos mesmos, permitindo ao gerente de projeto a qualquer instante ter uma visão dos bugs que ainda demandarão esforços para serem fechados, bem como identificar as fases do projeto que estão gerando maior número de bugs. Dessa forma, o gerente de projeto poderá organizar sua equipe para dedicar-se mais a resolução de bugs, evitando que próximo a liberação do *release* do sistema precise-se ainda de um grande esforço para finalizar o produto.

Com as métricas de esforço foi possível identificar os problemas quanto ao planejamento das atividades do projeto nas diversas fases, uma vez que não existia uma coerência na distribuição dos esforços das atividades entre as fases de Concepção, Elaboração, Construção e Transição, bem como acompanhar se os esforços despendidos durante o desenvolvimento do projeto estavam compatíveis com as estimativas realizadas no planejamento. Uma vez encontrados grandes desvios entre o esforço realizado e o planejado, é possível avaliar se as estimativas foram incorretas ou se existiu algum fator específic do projeto que levou o uma demanda maior de esforço. Até o momento em que foram coletadas as métricas, pôde-se identificar para os quatro projetos que o esforço estimado para as atividades de planejamento do projeto e gerência de configuração foram sub-estimados.

Por fim, seguindo as diretrizes do CMM 2, essas métricas servirão como base para estimativas futuras mais precisas, auxiliando nos acordos dos compromissos entre as pessoas interessadas no projeto, bem como para o acompanhamento dos resultados e

desempenhos reais confrontando-os com o planejado, possibilitando tomadas de ações corretivas diante dos desvios, e dando subsídio para assegurar que as alterações nos compromissos se dêem através de acordo entre as pessoas envolvidas.

Capítulo 6

Conclusões e Trabalhos Relacionados

O trabalho realizado nesta dissertação focalizou aspectos de gerenciamento que facilitam o acompanhamento de projetos de software segundo as diretrizes do CMM Nível 2, utilizando a manipulação de métricas como técnica para realizar estimativas, encontrar problemas no desenvolvimento e auxiliar o gerente na tomada de decisões.

As métricas aqui apresentadas foram selecionadas considerando as metas das KPAs do CMM Nível 2 e voltadas para um processo de desenvolvimento de software baseado no RUP, orientado a casos de uso e que trabalha com conceitos de iterações e fases de desenvolvimento. Durante a escolha destas métricas tivemos a preocupação de considerar tipos de métricas que pudessem mostrar diferentes visões do projeto tais como progresso, qualidade, esforço e estabilidade de requisitos.

As seções seguintes apresentam algumas limitações observadas e perspectivas de trabalhos futuros que visam dar continuidade ao projeto realizado; trabalhos similares realizados na área, servindo como fonte de consulta para pesquisa na área de processos para utilização de métricas de software; e por fim, algumas considerações finais sobre o trabalho realizado.

6.1 Principais Contribuições

Como principal contribuição deste trabalho destacamos a definição de um conjunto de métricas como passo inicial para implantação de um programa de métricas em uma organização. Conforme ressaltado nesta dissertação, a definição das métricas adequadas não é uma tarefa fácil, e é essencial para o sucesso da implantação de um programa de métricas em uma organização. O conjunto de métricas que foi proposto propicia ao gerente de projeto uma ampla visão do projeto, focando diversos aspectos tais como: garantia da qualidade, progresso, produtividade, acompanhamento de esforço e variação de tamanho do software. Dessa forma, a aplicação dessas métricas estará promovendo a melhoria do gerenciamento de projetos, e consequentemente do desenvolvimento do software.

As métricas propostas demonstraram ser simples de serem coletadas e analisadas, além de contribuírem para o alcance das metas das KPAs do CMM nível 2.

6.2 Trabalhos Relacionados

Existem vários trabalhos relacionados a essa área de pesquisa que, por ser uma necessidade real das grandes organizações no cenário atual, onde os sistemas estão cada vez maiores, mais complexos, envolvendo uma quantidade maior de pessoas e dificultando ainda mais as atividades de gerenciamento, representa uma área de grande interesse tanto científico quanto comercial. Essa seção apresenta alguns exemplos de trabalhos desenvolvidos como forma de melhorar o gerenciamento de projetos.

Silva no seu trabalho de dissertação [15] propõe um modelo de gestão de projetos de software guiado por artefatos, baseado no RUP e que aborda o desenvolvimento iterativo. O modelo oferece *templates* de artefatos a serem produzidos pelo gerente de projeto, ressaltando os aspectos que atendem as práticas de gerência de projetos do Nível 2 do CMM e as diretrizes de gestão recomendadas na Norma ISO9000-3. Assim como Silva, neste trabalho também focamos na gerência de projetos, considerando seus aspectos que

atendem ao CMM Nível 2. No entanto, focalizamos na definição, utilização e interpretação das métricas na gerência de projetos para atender às metas das KPAs CMM Nível 2, enquanto Silva apenas menciona as métricas como ferramenta dentro do seu modelo de gestão, para realização de estimativas com base nos históricos de aferições passadas, e para acompanhamento do progresso do projeto. Além disso, modificamos o fluxo de Gerenciamento de Projetos do RUP, para que este enfatize as atividades de coleta de métricas.

O Inspector, nome dado ao processo definido por Meneses [25], define um conjunto de atividades que visam tornar o acompanhamento de projetos uma tarefa sistemática, onde os problemas no desenvolvimento são identificados antes que atinjam proporções maiores. O Inspector oferece ao gerente de projeto duas visões de progresso complementares que fornecem suporte para um bom gerenciamento do processo de desenvolvimento. São elas: visão de desempenho e visão de funcionalidade. A primeira visão verifica o desempenho das equipes de desenvolvimento a partir da análise das atividades planejadas para as mesmas, definindo três métricas que mostram a qualidade do planejamento e a produtividade da equipe de desenvolvimento. Já a visão de funcionalidade define uma métrica, dirigida a casos de uso, que verifica o progresso funcional do sistema. De posse das duas visões, o gerente de projeto é capaz de identificar equipes com dificuldades e casos de uso com problemas no desenvolvimento. No trabalho apresentado nesta dissertação, as métricas propostas fornecem ao gerente de projeto, além da visão de progresso do projeto, uma visão do esforço estimado e realizado, qualidade do processo e do produto, e estabilidade do requisitos. Dessa forma, permite um melhor acompanhamento e controle da execução do projeto, bem como provê insumos para planejamentos mais realistas de projetos futuros.

Ainda na linha deste trabalho aqui apresentado, de definição de métricas para serem aplicadas em organizações no nível 2 de maturidade do CMM, podemos citar o trabalho de Baumert [35]. No seu trabalho, Baumert sugere seis indicadores para serem utilizados nas organizações no nível 2 do CMM: Progresso, Esforço, Custo, Qualidade, Estabilidade e Utilização de Recursos.

Na Tabela 6.1 relacionamos as métricas aqui apresentadas a alguns dos indicadores de Baumert, e às metas das KPAs do Nível 2 do CMM.

Progresso

Métricas	KPAs/Metas
(M-7) Dias de atraso para alcance dos *milestones* (M-8) Número de casos de uso implementados x planejados por iteração (M-9) Esforço planejado x realizado por iteração	**Planejamento de Projeto** • Planejar e documentar as atividades e os compromissos do projeto de desenvolvimento de software. • Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do projeto de software. **Supervisão e Acompanhamento de Projeto de Software** • Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software. **Gerência de Contrato de Software** ▪ A contratante acompanha o desempenho e resultados reais da contratada, comparando-os com os compromissos assumidos.

Esforço

Métricas	KPAs/Metas
(M-3) Distribuição do esforço por fluxo de atividades (M-4) Distribuição do esforço por fase de desenvolvimento (M-9) Esforço planejado x realizado por iteração (M-5) Produtividade da equipe por fase de desenvolvimento	**Planejamento de Projeto** • Planejar e documentar as atividades e os compromissos do projeto de desenvolvimento de software. • Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do projeto de software. **Supervisão e Acompanhamento de Projeto de Software** • Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software. • Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software. **Gerência de Contrato de Software**

	▪ A contratante acompanha o desempenho e resultados reais da contratada, comparando-os com os compromissos assumidos.

Qualidade – Auditorias

Métricas	KPAs/Metas
(M-14) Número de não conformidades registradas x número de não conformidades registradas	**Garantia de Qualidade de Software** • Verificar a conformidade das atividades e dos artefatos de software com os padrões, procedimentos e requisitos aplicáveis. • Encaminhar à gerência sênior todas as questões de não-conformidade que não possam ser resolvidas no âmbito do projeto de software.

Qualidade – Revisões

Métricas	KPAs/Metas
(M-13) Número de problemas registrados x número de problemas resolvidos.	**Supervisão e Acompanhamento de Projeto de Software** • Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software. • Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software. • Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e as pessoas envolvidas. **Garantia de Qualidade de Software** • Verificar a conformidade das atividades e dos artefatos de software com os padrões, procedimentos e requisitos aplicáveis. • Encaminhar à gerência sênior todas as questões de não-conformidade que não possam ser resolvidas no âmbito do projeto de software **Gerência de Configuração de Software** • Identificar, controlar e tornar disponíveis os artefatos selecionados.

Qualidade – Testes

Métricas	KPAs/Metas
(M-10) Número de bugs/Klock registrados por teste de iteração (M-12) Número de bugs registrados x Número de bugs fechados (M-11) Número de bugs encontrados após *release*	**Supervisão e Acompanhamento de Projeto de Software** • Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software. **Gerência de Configuração de Software** • Identificar, controlar e tornar disponíveis os artefatos selecionados. • Controlar as alterações nos artefatos de software identificados. • Informar as pessoas e grupos envolvidos acerca do estado e do conteúdo das *baselines* de software. **Garantia da Qualidade de Software** • Verificar a conformidade das atividades e dos artefatos de software com os padrões, procedimentos e requisitos aplicáveis.

Estabilidade - Requisitos

Métricas	KPAs/Metas
(M-1) Número de mudanças de requisitos solicitadas por fase de desenvolvimento (M-2) Número de mudanças de requisitos realizadas x número de mudanças solicitadas	**Gerência de Requisitos** • Documentar e controlar os requisitos alocados para estabelecer uma *baseline* para uso gerencial e da engenharia de software. • Manter planos, artefatos e atividades de software consistentes com os requisitos alocados. **Supervisão e Acompanhamento de Projeto de Software** • Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e as pessoas envolvidas. **Gerência de Configuração de Software** • Identificar, controlar e tornar disponíveis os artefatos selecionados. • Controlar as alterações nos artefatos de software identificados. • Informar as pessoas e grupos envolvidos acerca do estado e do conteúdo das *baselines* de software.

Estabilidade – Tamanho

Métricas	KPAs/Metas
(M-6) Tamanho do software	**Planejamento de Projeto** • Os grupos e as pessoa afetadas estão de acordo com os compromissos relacionados ao projeto de software. • Documentar as estimativas de software a serem usadas no planejamento e acompanhamento do projeto de software. **Supervisão e Acompanhamento de Projeto de Software** • Acompanhar os resultados e desempenhos reais confrontando-os com o plano de desenvolvimento de software. • Tomar ações corretivas e gerenciá-las até sua conclusão, sempre que resultados ou desempenhos reais desviarem significativamente do plano de desenvolvimento de software. • Assegurar que as alterações nos compromissos de software se dêem através de acordo entre os grupos e as pessoas envolvidas. **Gerência de Contrato de Software** • A contratante e a contratada concordam com os compromissos assumidos entre eles. • A contratante acompanha o desempenho e resultados reais da contratada, comparando-os com os compromissos assumidos. **Gerência de Configuração de Software** • Identificar, controlar e tornar disponíveis os artefatos selecionados. • Controlar as alterações nos artefatos de software identificados. • Informar as pessoas e grupos envolvidos acerca do estado e do conteúdo das *baselines* de software.

Tabela 6.1 Indicadores, Métricas e as KPAs do CMM nível 2

Além destes trabalhos citados, outros trabalhos relacionados podem ser encontrados em [19, 20, 21, 22, 27, 30, 36]. Todos esses trabalhos ressaltam o uso das métricas como

ferramenta para auxiliar o gerente de projeto no controle da qualidade e acompanhamento do processo de desenvolvimento e do produto em construção.

6.3 Trabalhos Futuros

As métricas propostas por este trabalho irão apoiar fortemente o processo de qualdiade do CESAR, em especial no contexto do projeto CMM 2, cujas metas se concentram no atendimento das KPAs definidas neste nível. A idéia é que as métricas sejam coletadas a partir das ferramentas adotadas pelo CESAR para registro de horas alocadas nos projetos, registros de bugs, e auditorias realizadas nos projetos. Com esse trabalho, poderemos coletar métricas de um conjunto mais amplo de projetos, considerando projetos de diferentes portes e características e teremos a oportunidae de acompanhar efetivamente a realização das atividades propostas pelas KPAs do CMM 2 através das métricas coletadas.

A partir da coleta das métricas em uma organização deverá ser criada uma base histórica dos projetos e assim poderão ser definidos os limites aceitáveis para a organização como resultados das métricas, bem como estruturar as interpretações dos resultados e avaliar as tendências dos projetos. A definição desses limites aceitáveis e das interpretações dos resultados das métricas a partir desses limites é um próximo trabalho a ser realizado, com base nos dados dos projetos coletados no CESAR. Para tal, iremos nos basear nas fases seguintes da abordagem de Donald McAndrews [33], que trata da efetiva implementação do processo de coleta e análise de métricas nos projetos da organização e da melhoria deste processo através de avaliações dos planos e procedimentos, decorrentes da evolução da maturidade dos projetos e necessidade de mudança das métricas.

Outro aspecto a ser dado continuidade com este trabalho é o estudo sobre ponto de complexidade de casos de uso, para avaliarmos sua utilização como unidade das métricas de tamanho e progresso. O objetivo é avaliar os benefícios de passar a medir tamanho do software, progresso e produtividade em termos de pontos de complexidade de casos de uso, ao invés de linhas de código ou número de casos de uso, por acreditarmos que essa

unidade fornece uma visão mais concreta dos projetos que são desenvolvidos baseados em casos de uso, uma vez que os casos de uso guiam os projetos desde os requisitos até os testes.

Ainda como um maior aprofundamento da relação entre as métricas aqui propostas e o CMM nível 2, poderemos relacionar essas métricas à Característica Comum de Medição e Análise das KPAs do nível 2 do modelo CMM, de forma a identificar novas métricas que também possam contribuir para o alcance deste nível de maturidade. Essa Característica Comum sugere medições para determinar o estado das atividades executadas, propostas em cada KPA do CMM.

6.4 Considerações Finais

Ainda que existam organizações preparadas, que apresentem um bom nível de maturidade e utilizem um processo de desenvolvimento de software de qualidade, essa não é a realidade para maioria das empresas de software. O que prevalece são empresas que não mantêm um padrão de qualidade na produção para controle de seus produtos, satisfazendo os requisitos do cliente no tempo previsto e com um custo aceitável [38].

O sucesso de um projeto depende da eficiência da equipe de desenvolvimento e de um gerenciamento eficaz, que planeje as atividades, monitore os riscos e identifique dificuldades enfrentadas no desenvolvimento [11]. A competitividade e a existência de projetos de software cada vez maiores e mais complexos tornou a utilização de técnicas de gerenciamento uma prática fundamental para obtenção do sucesso de projetos de desenvolvimento de software. Nesse contexto, a mensuração de projetos de software representa um importante auxílio ao gerente de projeto, quantificando propriedades que permitem controlar o processo de desenvolvimento e a qualidade do produto.

Por outro lado, a normalização e certificação de empresas de software no Brasil já é uma realidade, recente, porém crescente. A busca pela qualidade do processo de desenvolvimento de software tem despertado a necessidade nas organizações de software

de adotarem as normas e padrões específicos para software que têm sido propostos. Entre eles, podemos destacar o modelo SW-CMM (*Capability Maturity Model for Software*), que tem sido bastante difundido entre as empresas.

O trabalho realizado buscou o aperfeiçoamento do gerenciamento de projetos em organizações que realizam o desenvolvimento de software, introduzindo na organização uma cultura de utilização de métricas para realização de acompanhamento dos projetos, gerenciamento da qualidade do produto gerado e construção de base histórica para estimativas de projetos futuros.

Sem a utilização de métricas, o planejamento e acompanhamento de projetos tornam-se atividades empíricas, realizadas com base apenas no sentimento e experiência dos gerentes de projetos. O conjunto de métricas proposto neste trabalho demostrou atender às metas das KPAs do CMM nível 2, fornecendo ao gerente uma ampla visão do projeto, considerando aspectos de qualidade, progresso, utilização de esforço e recursos.

As métricas aplicadas aos projetos durante este trabalho permitiram que os gerentes tivessem uma melhor visão do projeto quanto ao estado dos bugs abertos, esforços despendidos comparados ao planejado, distribuição de esforços nas fases do projeto, status das atividades planejadas, além de detectar problemas no planejamento dos projetos quanto ao entendimento dos conceitos de fases e iterações, segundo o RUP.
Enfim, as métricas ajudaram no gerenciamento do projeto e proveram insumos para planejamentos futuros.

O fato dos projetos avaliados não terem sido planejados seguindo o conceito de fases do RUP, além de possuírem iterações muito longas, limitou a aplicação das métricas tal qual foram propostas neste trabalho. A coleta e avaliação das métricas ao final de cada iteração pode apresentar tardiamente os desvios na execução do projeto, uma vez que as iterações tenham sido planejadas com durações muito longas (mais de um mês).

Com o resultado deste trabalho, o CESAR irá coletar as métricas aqui apresentadas em todos os seus projetos, seguindo seu programa de qualidade que busca alcançar o CMM Nível 2 para a organização. Com a estruturação da Gerência de Qualidade, os projetos do CESAR passarão a ter Engenheiros de Processo, os SQAs, trabalhando nos projetos para realizarem as auditorias e revisões, bem como coletarem as métricas de projeto. A partir desta estruturação, e com o uso efetivo das métricas, poderemos avaliar mais detalhadamente a contribuição dessas métricas para o alcance das metas do CMM Nível 2, bem como identificar novas métricas que deverão ser coletadas para atingir esse objetivo.

Este trabalho abre espaço para realização de novos trabalhos em uma área ainda bastante nova e que necessita de pesquisa e incentivo, com o intuito de aperfeiçoar o gerenciamento de projetos de software. Um processo que formalize a utilização de um conjunto de métricas representa grande interesse para empresas, que necessitam de um maior controle sobre seus projetos.

Através da experiência realizada nos quatro projetos do CESAR, verificamos que o investimento para implantação de um programa de métricas não é tão alto, podendo contar com a utilização de ferramentas gratuitas para registro e coleta de dados, que facilitam a extração e avaliação das métricas. Por outro lado, o benefício trazido aos projetos, através das visões apresentadas ao gerente pelos gráficos obtidos, permitindo comparações com o planejamento e com outros projetos, é bastante compensador. Dessa forma, essa experiência será estendida para todos os projetos do CESAR, visando melhorar o planejamento e gerenciamento dos projetos e como conseqüência, caminhar mais facilmente para obtenção do Nível 2 de maturidade do CMM.

Referências

[1] Fiorino S.T., Staa A.V., Baptista R.M., *Engenharia de Software com CMM*, Brasport, 1998, ISBN 85-85840-84-6.

[2] "Métricas e Estimativas de Software", http://apinfo.com/artigo44.htm, último acesso em mar. 2002.

[3] Rational Software Corporation, informações institucionais disponíveis em http://www.rational.com, último acesso em nov. 2001.

[4] SEI – Software Engineering Institute, informações institucionais disponíveis em http://www.sei.cmu.edu/cmm, último acesso em nov. 2001.

[5] Rational Software Corporation, "Reaching CMM Levels 2 and 3 with the Rational Unified Process", disponível em http://www.rational.com/products/whitepapers/100416.jsp, set. 1998. Último acesso em nov. 2001.

[6] "Métricas de Software", http://metricas.tw.eng.br/, último acesso em mar. 2002.

[7] Florac W.A., Carleton A.D., *Measuring the Software Process: Statistical Process Control for Software Process Improvement,* SEI Series in Software Engineering, Addison-Wesley, jun. 1999.

[8] Grady R.B., *Pratical Software Metrics for Project Management and Process Improvement*, Prentice Hall, 1992, ISBN 0-13-720384-5.

[9] Ministério da Ciência e Tecnologia, *Qualidade e Produtividade no Setor de Software Brasileiro 1999*, Qual. Set. software bras., Brasília, nº3, 2000.

[10] “Discussion Estimating Software Cost, T. Capers Jones”, http://www.highq.be/quality/estimat.htm, ultimo acesso em mar. 2002.

[11] Pressman R.S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 5ª ed., 1999.

[12] The Capability Maturity Model Guidelines for improving the software process, SEI Series in Software Engineering.

[13] Humphrey W. S., *Managing the Software Process*, Addison-Wesley, 1990.

[14] Santander V.F.A. e Vasconcelos A.M.L., *Mapeando o Processo Unificado em Realação ao CMM – Nível 2*, XI CITS – Qualidade de Software, Curitiba, 2000.

[15] Silva E.M.M., *Um Modelo de Gestão de Projetos de Software*, dissertação de mestrado do Centro de Informática – UFPE, abr. 2001.

[16] DeMarco T., *Controlling Software Project*, Prentice Hall, 1982, ISBN 0-13-171711-1.

[17] Fernandes A.A., *Gerência de Software Através de Métricas*, Atlas S.A, 1995, ISBN 85-224-1264-2.

[18] Kruchten P., *The Rational Unified Process – An Introduction*, Addison-Wesley, 1998, ISBN 0201604590.

[19] Gomes A., Oliveira K. e Rocha A.R., *Avaliação de Processos de Software Baseada em Medições*, XV SBES – Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 2001.

[20] Wiegers K.E., “A Software Metrics Primer”, paper publicado no site www.processimpact.com, último acesso em jan 2002.

[21] Wiegers K.E., “Lessons from Software Work Effort Metrics”, paper publicado no site www.processimpact.com, último acesso em jan 2002.

[22] Wiegers K.E., “Software Metrics: Ten Traps to Avoid”, paper publicado no site www.processimpact.com, último acesso em jan 2002.

[23] Hughes B. e Cotterell M., *Software Project Management*, McGrawHill, 1999, ISBN 0077095057.

[24] Maciel T.M., *Garantindo a Qualidade na Especificação do Plano de Projeto Iterativo de Software*, XIV Simpósio Brasileiro de Engenharia de Software, Anais do Workshop de Qualidade, João Pessoa, out. 2000.

[25] Meneses J.B., *Inspector: Um Processo de Avaliação de Progresso para Projetos de Software*, dissertação de mestrado do Centro de Informática - UFPE, fev. 2001.

[26] The Project Management Institute Inc., “A Guide to the Project Management Body of Knowledge – PMBOK Guide 2000 Edition”, disponível em http://www.pmi.org/publictm/download/2000welcome.htm, último acesso em jan. 2002.

[27] Brito, Abreu F., “MOOD – Metrics for Object-Oriented Design”, OOPSLA’94 Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, 1994.

[28] Gilb T., *Principles of Software Engineering Management*, Addison-Wesley, 1990.

[29] Henderson-Sellers B., Younessi H., Graham I.S., *The Open Process Specification*, Addison Wesley, Open Series, 1997.

[30] “Using MetricCenter Workstation for Meeting the Requirements of the SEI´s Software CMM level 2”, Technical Paper, Distributive Software, www.distributive.com, 2000, último acesso em fev. 2002.

[31] ISO/IEC 12207 Tecnologia da Informação – Processos de ciclo de vida de software.

[32] Rational Unified Process 2000, propriedade e direitos reservados da Rational Software Corporation.

[33] McAndrews D.R., “Establishing a Software Measurement Process” Documento No CMU/SEI-93-TR-16, 1993.

[34] Waina R., “Capability Maturity Model Benefits”, Multi-Dimensional Maturity, Versão 2.0, mar. 2002.

[35] Baumert J. H., McWhinney M. S., “Software Measures and the Capability Maturity Model”, Documento No. CMU/SEI-TR-25, Carnegie Mellon University, Software Engineering Institute, 1992.

[36] Anda B. *et al*, “Estimating Software Development Effort based on Use Cases – Experiences from Industry”.

[37] Processo de Software do CESAR, “ProSCes”, 2000, propriedade e direitos reservados do CESAR.

[38] Kotonia G., Sommerville I., Requirements Engineering Processes and Techniques, Horizon Pubs & Distributors Inc, 1998.

[39] Distributive Software, “Using MetricCenter Workstation and CMM Level 2”, disponível em http://www.distributive.com. Último acesso em nov. 2001.

[40] Westfall L., “Software Metrics that Meet your Information Needs”, Software Measurement Services, Plano TX 75075, ASQC 49th Annual Quality Congress Proceedings.

[41] “What is the Balanced Scorecard?”, http://www.balancedscorecard.org/, último acesso em 15/08/2002.

[42] “The Critical Success Factors Approach”, http://informationr.net/ir/1-3/paper8.html, último acesso em 15/08/2002.

Apêndice A

Este apêndice apresenta o modelo de artefato para registro da coleta de métricas, proposto para ser utilizado no fluxo de atividades de Gerenciamento de Projetos do RUP, visando tornar mais prática e simples a atividade de coleta de métricas e o registro das mesmas.

A notação utilizada para representar esse artefato é bastante simples, sendo o artefato subdividido em seções, onde cada seção pode ser composta de subseções, e conter informações, representadas entre <>, que indicam como a seção deverá ser preenchida e fornecem dicas sobre como obter os dados necessários. A página seguinte apresenta o modelo de artefato proposto.

Formulário de Coleta de Métricas
(Versão 1.0, junho de 2002)

1. Identificação

Nome do Projeto: _____
Gerente do Projeto: _____
Data de início: _____
Tecnologias Utilizadas: _____

2. Dados de início do projeto

<Apresenta os dados do planejamento no início do projeto>

Qtd. prevista de casos de uso	
Qtd prevista de iterações por fase	
Concepção	
Elaboração	
Construção	
Transição	

3. Dados das iterações

<Apresenta os dados planejados para cada iteração e os dados reais, coletados ao final das mesmas>

3.1 Fase de Concepção

3.1.1 Iteração Preliminar

	Esforço estimado(hs)	Esforço utilizado (hs)	Desvio (%)
Requisitos			
Análise e Projeto			
Implementação			
Testes			
Implantação			
Gerência de Projetos			
Configuração e Gerência de Mudanças			

Milestones	Data Prevista	Data Realizada	Desvio em Dias

Total de linhas de código	
Casos de uso estimados para serem implementados na iteração	
Casos de uso efetivamente implementados na iteração	
Bugs encontrados	
Solicitações de mudanças abertas na iteração	
Solicitações de mudanças concluídas na iteração	
Número de Problemas registrados/revisão	
Auditorias realizadas	
Número de não conformidades encontradas/auditoria	

Release	
Versão	
Total de linhas de código	
Total de bugs	

Bugs Fechados	
Problemas resolvidos	
Não conformidades resolvidas	

3.2 Fase de Elaboração

3.2.1 Iteração 1

	Esforço estimado(hs)	Esforço utilizado (hs)	Desvio (%)
Requisitos			
Análise e Projeto			
Implementação			
Testes			
Implantação			
Gerência de Projetos			
Configuração e Gerência de Mudanças			

Milestones	Data Prevista	Data Realizada	Desvio em Dias

Total de linhas de código	
Casos de uso estimados para serem implementados na iteração	
Casos de uso efetivamente implementados na iteração	
Bugs encontrados	
Solicitações de mudanças abertas na iteração	
Solicitações de mudanças concluídas na iteração	
Número de Problemas registrados/revisão	
Auditorias realizadas	
Número de não conformidades encontradas/auditoria	

Release	
Versão	
Total de linhas de código	
Total de bugs	

Bugs Fechados	
Problemas resolvidos	
Não conformidades resolvidas	

3.2 Iteração 2

...

4. Dados do final do projeto

<Apresenta os dados reais no final do projeto>

Qtd. de casos de uso implementados	
Qtd de iterações por fase	
Concepção	
Elaboração	
Construção	
Transição	
Total de linhas de código	

5. Apresentação das Métricas

<Nesta seção deverão ser apresentados os resultados das métricas em forma de gráficos, com base nos dados registrados nas seções anteriores>