Universidade Federal de Pernambuco
Centro de Informática

RENATA ENDRISS CARNEIRO CAMPELO

XP-CMM2: Um Guia para Utilização de Extreme Programming em um Ambiente Nível 2 do CMM

ORIENTADOR

Prof. Hermano Perrelli de Moura

Recife, novembro de 2003

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RENATA ENDRISS CARNEIRO CAMPELO

XP-CMM2: Um Guia para Utilização de Extreme Programming em um Ambiente Nível 2 do CMM

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(A): HERMANO PERRELLI DE MOURA

RECIFE, NOVEMBRO/2003

Campelo, Renata Endriss Carneiro
XP-CMM : uma guia para utilização de Extreme Programming em um ambiente nível do CMM / Renata Endriss Carneiro Campelo . – Recife : O Autor, 2003.
196 folhas : il., fig., tab., quadros, gráf.

Dissertação (mestrado) – Universidade Federal de Pernambuco. Cin. Ciências da Camputação, 2003.

Inclui bibliografia e apêndices.

1. Engenharia de software – Metodologia de desenvolvimento. 2. CMM (copability Maturity Model) – Análise do nível 2 de maturidade. 3. Metodologias ágeis – XP (Extreme Programming) – Guia de uso em CMM. I. Título.

004.41	CDU (2.ed.)	UFPE
005.12	CDD (22.ed.)	BC2005-082

Dissertação de Mestrado apresentada por **Renata Endriss Carneiro Campelo** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"XP-CMM2: Um Guia para Utilização de Extreme Programming em um Ambiente Nível 2 do CMM"**, orientada pelo **Prof. Hermano Perrelli de Moura** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Prof. Jones Oliveira de Albuquerque
Departamento de Matemática e Física / UFRPE

Prof. Hermano Perrelli de Moura
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 5 de novembro de 2003

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Agradecimentos

Agradeço aos meus pais por tudo. Talvez por coisas que eu nem mesma reconheça neste momento. Mas, em especial, pelo amor, incentivo, por me darem tudo o que eu sempre precisei. Pela nossa linda família. Por serem meu porto seguro.

Ao meu amor Zé, companheiro e incentivador. Pelos finais de semana em casa comigo, para que eu pudesse estudar. Por escutar minhas angústias nos momentos em que eu achava que nunca ia chegar lá. Por me incentivar nestes momentos. Por se mostrar orgulhoso a cada vitória ou capítulo concluído ou artigo publicado. Por dividir minhas alegrias e tristezas nesta longa caminhada.

Às minhas queridíssimas irmãs, que preenchem a minha vida de alegria.

A todos os que fazem o C.E.S.A.R, local em que trabalho, que me proporciona crescimento e conhecimento. Sem meu trabalho, provavelmente não teria chegado lá. Em especial, a duas pessoas importantíssimas: Valença e Teresa. Por sempre terem me incentivado a concluir meu trabalho, por terem me proporcionado atividades relacionados à minha pesquisa, por terem sido flexíveis quando precisei de tempo para estudar, de projeto para realizar estudo de caso, de férias para descansar. Divido esta vitória com vocês também. Muito obrigada.

A Hermano, por ter contribuído neste trabalho, pelas dicas valiosas, pelos trabalhos realizados de última hora (como as revisões dos artigos perto da meia noite). Principalmente por me transmitir confiança, por diminuir minhas angústias e medos, por me colocar para cima nos momentos difíceis.

Aos professores Alexandre e Jones, por avaliarem este trabalho.

Aos amigos e a todos que contribuíram direta ou indiretamente neste trabalho. Não daria para listar todos os nomes aqui. Obrigada a todos que torcem por mim.

Resumo

Recentemente a comunidade de software vem se deparando com um grupo de novas metodologias de desenvolvimento de software, classificadas como “metodologias ágeis”. Algumas das metodologias que fazem parte deste grupo são Extreme Programming (XP) e SCRUM, sendo XP a mais conhecida e utilizada. Estas metodologias possuem em comum um conjunto de valores para o desenvolvimento de software, priorizando: indivíduos e iterações sobre processos e ferramentas; software funcionando sobre documentação compreensiva; colaboração do cliente sobre negociação de contrato; resposta à mudança sobre seguir um plano. Em paralelo à disseminação das metodologias ágeis, os investimentos em qualidade de software vêm aumentando a cada ano. Pesquisas realizadas sobre o setor de software, indicam um crescimento na adoção de modelos de qualidade como ISO 9000 e *Capability Maturity Model for Software* (CMM). Modelos de qualidade e metodologias ágeis possuem fundamentos opostos, como é possível notar nos valores definidos por essas metodologias. Autores de metodologias ágeis freqüentemente criticam modelos como o CMM. Em contra partida, alguns trabalhos indicam que é possível utilizar as duas abordagens em um mesmo ambiente. Este trabalho apresenta o Guia XP-CMM2, que tem como objetivo apoiar as organizações no uso da metodologia ágil XP em um ambiente nível 2 do CMM. Com o uso do Guia XP-CMM2, as organizações deverão se beneficiar da agilidade proposta por XP e da maturidade adquirida com o nível 2 do modelo de qualidade de software mais respeitado do mundo, o CMM. Para a elaboração do Guia XP-CMM2, foi realizado inicialmente um diagnóstico da satisfação de XP ao nível 2 do CMM e, depois, para cada problema identificado, uma solução foi proposta. Finalmente o Guia XP-CMM2 foi aplicado em dois ambientes distintos visando avaliação dos resultados obtidos.

Palavras-chave: Capability Maturity Model (CMM), Extreme Programming, XP, Metodologias Ágeis, Guia XP-CMM2.

Abstract

Software community has recently been faced to a group of new methodologies on software development, named "agile methodologies". Some of these methodologies are Extreme Programming (XP) and SCRUM, being XP the most known and used. These methodologies share some values for software development: individuals and iteractions over processes and tools; software working over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan. Along with the spread of agile methodologies, the investiments on software quality have been increasing each year. Researches on software indicates an increase on using quality models like ISO 9000 and CMM. Quality models and agile methodologies have opposite fundamentals, as one can figure out with the values defined by each methodology. Agile methodologies' authors frequently criticize models like CMM. In contrast, some papers indicate that it is possible to use both in the same environment. This study presents the XP-CMM2 Guide, wich has the objective of supporting the organizations on using XP in a CMM level 2 environment. Using the XP-CMM2 Guide, the organizations should have benefits on both the agility proposed by XP and the maturity acquired by the level 2 of the most respectful software quality model, CMM. In order to create the XP-CMM2 Guide, it was initially done a diagnosis on the satisfaction of XP through level 2, and after, for each problem recognized, one solutions was proposed. Finally, the XP-CMM2 Guide was applied in two different environments aiming the evaluation of the obtained results.

Keywords: Capability Maturity Model (CMM), Extreme Programming, XP, Agile Methodologies, XP-CMM2 Guide.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

Este capítulo introduz conceitos e apresenta a motivação para este trabalho conforme descrito abaixo:

- Seção 1.1 – Motivação: descreve a motivação para realização deste trabalho, incluindo aspectos sobre engenharia de software, qualidade de software, metodologias ágeis e o problema ao qual este trabalho se propõe a resolver;
- Seção 1.2 – Objetivos do Trabalho: descreve os objetivos do trabalho e como o mesmo será conduzido;
- Seção 1.3 – Estrutura da Dissertação: descreve como este documento está organizado.

1.1 Motivação

Um dos fatores que mais tem influenciado a mudança de vida das pessoas é a produção de software. Software está atualmente, especialmente com o advento da Internet, em todos os locais e atividades da sociedade: hospitais, empresas, controle de tráfego aéreo, educação, etc. Há muito tempo, o software deixou de apenas automatizar as tarefas realizadas pelas pessoas para também mudar a forma destas realizações.

Nos anos 60 e 70 a produção de software se dava de forma desorganizada e sem nenhuma sistematização. Isto gerou uma grande insatisfação por parte dos consumidores dos softwares produzidos. Esta insatisfação surgiu em decorrência de atrasos em cronogramas, orçamentos estourados e da baixa qualidade do produto final. Todos estes problemas ficaram conhecidos mundialmente como a "Crise do Software" [14].

O termo "Engenharia de Software" surgiu no final de 1960 motivado pela crise do software [14]. Buscava-se a elaboração de teorias, métodos e ferramentas

relacionadas à produção de software, visando obtenção de maior qualidade, custo e prazo previsíveis para o produto de software gerado [14]. A partir de então, produzir software passou a ser mais do que programar, mas também documentar, produzir modelos, realizar tarefas [14]. O IEEE [54] define Engenharia de Software da seguinte forma [15]:

"Engenharia de Software: (1) A aplicação de uma abordagem sistemática, disciplinada, confiável para desenvolvimento, operação e manutenção de software; isto é aplicação de engenharia de software. (2) O estudo das abordagens descritas em (1)".

Segundo Sommerville, "o objetivo da Engenharia de software é produzir produtos de software" [14]. A Engenharia de Software trata de processos, métodos e ferramentas **[40]**. Processos estabelecem o que deve ser feito para entrega efetiva de software; métodos descrevem informações técnicas de como deve ser feito para construir software; ferramentas automatizam parcialmente ou completamente os métodos e processos [40].

Apesar da Engenharia de Software já ter evoluído bastante ao longo dos seus 40 anos de existência, projetos de software continuam sendo cancelados ou entregues com qualidade baixa, atrasados, acima do custo acordado. O Standish Group [61] realiza pesquisas, chamadas CHAOS [42], a respeito de projetos de software e publica relatórios a respeito. A Tabela 1-1 exibe os resultados dos projetos de software segundo o CHAOS para os anos de 1994, 1996, 1998 e 2000 [20].

Resultados dos projetos	1994	1996	1998	2000
Sucesso	16%	27%	26%	28%
Falha	31%	40%	28%	23%
Finalizado	53%	33%	46%	49%

Tabela 1-1 - Resultados dos projetos de software segundo o CHAOS Report [20]

Os resultados dos projetos são classificados em três tipos [20]:

1. Sucesso: o projeto é finalizado dentro do prazo e orçamento estabelecido, com todas as características e funcionalidades originalmente especificadas;
2. Finalizado: O projeto é finalizado e operacional, mas estourou o orçamento, o prazo e possui menos características e funcionalidades que as especificadas inicialmente;

3. Falha: O projeto é cancelado antes de ser finalizado ou nunca é implementado.

Para a pesquisa do ano 2000, mais de 30.000 projetos foram pesquisados. Os resultados mostram que grande parte dos projetos é finalizada com problemas (os projetos classificados como “finalizados”), outra grande parte é considerada como falha. A quantidade de projetos que obtém sucesso vem aumentando ao longo dos anos, mas o percentual ainda é considerado baixo.

O foco na qualidade suporta a evolução e aprimoramento da Engenharia de Software. Segundo Pressman [40], “o gerenciamento da qualidade total e filosofias similares fomentam uma cultura de melhoria contínua de processos, e esta cultura leva ao desenvolvimento de abordagens mais maduras de engenharia de software”.

Qualidade é um termo amplo, mesmo analisando-o apenas sob o ponto de vista da produção de software. Diferentes estudiosos no assunto definiram qualidade de diferentes formas [11]:

- W. Edwards Deming define qualidade em termos de precisão do esforço. Neste sentido, a qualidade é maior quando planos e especificações são feitos de acordo com a capacidade da organização;
- Para Philip Crosby qualidade está relacionada com a conformidade às especificações. Um produto de qualidade é obtido a partir de especificações bem escritas que descrevam as necessidades do cliente;
- Joseph M. Juran define qualidade em termos de adequação ao uso. Um produto deve atender às necessidades dos usuários, mesmo que não atenda às suas especificações;
- Tom Peters define qualidade como atingir às expectativas dos clientes. A qualidade do produto é medida a partir da conformidade das especificações às reais necessidades dos clientes.

Apesar de ser difícil de definir, os benefícios da qualidade de software são amplamente discutidos na comunidade, dentre eles, é importante ressaltar: redução do número de defeitos, confiabilidade, diminuição de re-trabalho, diminuição do ciclo de vida de produção do software, redução de custos, maior valor agregado, satisfação do cliente, maior motivação da equipe de produção.

No intuito de aumentar, medir e garantir a qualidade do software produzido, surgiram alguns modelos de qualidade de software. Dentre os modelos de qualidade

propostos, os mais comentados atualmente são as normas da série ISO 9000 [5], a futura norma ISO/IEC 15504 (SPICE – *Software Process Improvement and Capability dEtermination*) [45], *Capability Maturity Model for Software* (CMM-SW) [30] e, recentemente, o *Capability Maturity Model Integrated* (CMM-I) [47].

Metodologias e processos descrevem "como deve ser realizado" os requisitos dos modelos de qualidade. Um processo de desenvolvimento de software possui os seguintes papéis [12]:

- Guiar as atividades da equipe;
- Especificar quais e quando artefatos devem ser desenvolvidos;
- Dirigir as atividades individuais dos desenvolvedores e da equipe como um todo;
- Oferecer critérios para monitorar e medir as atividades e produtos do projeto.

Dentre as metodologias de desenvolvimento de software mais conhecidas atualmente está o RUP – Rational Unified Process [35], desenvolvido pela Rational Software [59].

1.1.1 Metodologias Ágeis

Recentemente a comunidade de software vem se deparando com um grupo de novas metodologias de desenvolvimento de software, classificadas como "Metodologias Ágeis". Em fevereiro de 2001 um grupo de dezessete pessoas se reuniu no estado de Utah, Estados Unidos, e assinou o "Manifesto para o Desenvolvimento Ágil de Software". Deste grupo de pessoas faziam parte representantes de metodologias como Extreme Programming (XP) [25] e Crystal [48] e simpatizantes da necessidade de uma alternativa aos processos tradicionais de desenvolvimento de software dirigidos à documentação [19]. Este grupo se intitulou "Aliança Ágil de Desenvolvimento" [44].

Os envolvidos com este movimento, alegam que as regras e procedimentos criados em torno da Engenharia de Software tornaram a produção muito complexa e dispendiosa e que a comunidade perdeu o controle sobre os vários documentos e modelos tidos como necessários em qualquer desenvolvimento [53].

O "Manifesto para o Desenvolvimento Ágil de Software" definiu os seguintes valores para o desenvolvimento de software [19]:

1. **Indivíduos e iterações** *sobre processos e ferramentas*.
2. **Software funcionando** *sobre documentação compreensiva*.

3. **Colaboração do cliente** *sobre negociação de contrato*.

4. **Resposta à mudança** *sobre seguir um plano*.

Estes valores devem ser compreendidos da seguinte forma: a parte em negrito das sentenças refletem o coração das metodologias ágeis, o que é realmente valioso nestas metodologias. A parte em itálico da sentença não possui status de "não importante", mas de "menos importante" que a parte da esquerda, ou de menor prioridade [19].

O primeiro valor sugere que processos e ferramentas são importantes, mas que iterações e o talento das pessoas são muito mais. A Aliança Ágil alega que as metodologias, em geral, dão mais ênfase ao processo usado do que às pessoas que realizarão o trabalho. Da mesma forma, apesar de documentação apoiar o desenvolvimento de software, o foco do projeto deve estar no produto final: o software apto a ser executado, conforme sugere o segundo valor. O terceiro valor expressa que apesar de um contrato ser importante para estabelecer limites e responsabilidades de um projeto de software, uma equipe só pode entender e entregar o que o cliente deseja através de colaboração do cliente. Além disto, segundo o quarto valor, um plano é importante para um projeto, mas aceitar as mudanças ao que nele está contido de forma normal é ágil é ainda mais importante. [19].

Além dos valores, o "Manifesto para o Desenvolvimento Ágil de Software" também relaciona os princípios que a Aliança Ágil espera de um processo de desenvolvimento de software. Dentre eles, é importante ressaltar [28]:

- Satisfazer o cliente através de entrega contínua e rápida de software de valor;
- Mudanças em requisitos devem ser bem vindas, mesmo que em momentos tardios do desenvolvimento;
- Cliente e desenvolvedores devem trabalhar juntos, diariamente no projeto;
- Deve-se utilizar a comunicação face-a-face como o método mais eficiente e efetivo de fluxo de informações;
- Deve-se utilizar como medida principal de progresso do projeto o software funcionando;
- Deve-se assumir simplicidade para todos os aspectos do projeto.

Dentre as metodologias ágeis mais comentadas atualmente estão **[27]**: Extreme Programming (XP), Adaptive Software Development (ASD), Crystal, Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD) e Scrum, sendo XP a mais conhecida e utilizada.

1.1.2 O Problema

Os objetivos das metodologias ágeis são, indiscutivelmente, de interesse de todos os que trabalham com a produção de software: todos querem desenvolver de forma mais rápida, barata, com menos burocracia, exatamente o que o cliente deseja, com qualidade. No entanto, surgem algumas críticas relacionadas à garantia da qualidade do produto final.

Um dos valores da Aliança Ágil é "Indivíduos e iterações ao invés de processos e ferramentas" [19]. E se os indivíduos não forem bons tecnicamente? E se os indivíduos saírem do projeto durante o seu andamento? Ainda assim é possível produzir software de qualidade? Além disto, os relatos de sucesso publicados se devem ao emprego sistemático destas metodologias ou a esforços dos indivíduos envolvidos ou situação favorável de desenvolvimento?

Nawrocki et al, relatam **[21]** um experimento realizado em uma universidade em que uma equipe produziu software segundo a abordagem do CMM nível 2 e outra segundo XP. Como conclusão, relata sobre o uso de XP: "Provou ser não trivial para equipes formadas com pessoas novas, onde os membros são inexperientes e não se conhecem muito bem". Além disto, conclui que "A comparação entre os processos mostrou que a abordagem CMM é menos suscetível a falhas, pois coloca a responsabilidade sobre o processo da organização, enquanto o processo XP coloca a responsabilidade sobre o envolvimento e experiência dos membros da equipe".

Algumas questões emergem ao tentar relacionar modelos de qualidade de software e metodologias ágeis de desenvolvimento de software:

1. É possível conviver em uma mesma organização com os formalismos requeridos por estes modelos de qualidade e o não formalismo pregado pelas metodologias ágeis?
2. Os dois mundos, o mundo dos modelos de qualidade e o mundo das metodologias ágeis, podem coexistir em um mesmo ambiente?
3. É possível obter os benefícios propostos pelos modelos de qualidade e pelas metodologias ágeis?

4. Empresas que buscam ou que são certificadas e/ou avaliadas segundo modelos de qualidade existentes, como ISO e CMM, podem optar pelo uso de uma metodologia ágil de forma que não comprometa a avaliação e/ou certificação alcançada?

Um artigo publicado por Mark Paulk, um dos elaboradores do modelo CMM, inicia uma discussão visando responder algumas das perguntas citadas [31]. Este artigo relaciona a metodologia ágil mais difundida atualmente, XP e o modelo para melhoria de processos CMM, analisando como XP se comporta em relação ao modelo

Paulk relata que XP é um processo documentado e disciplinado e que atende a vários processos definidos pelo modelo CMM. Como conclusão, Paulk ressalta que:

- XP foca no trabalho técnico e o CMM nas questões de gerenciamento;
- XP não aborda questões relacionadas à institucionalização das práticas em uma organização, o que é considerado crucial para o CMM;
- XP aborda até um determinado limite questões de gerenciamento e acompanhamento de projetos.

Apesar de relacionar algumas diferenças entre XP e CMM, Paulk sugere que ambos podem conviver em um mesmo ambiente: "XP possui boas práticas de engenharia que podem funcionar bem com o CMM e outros métodos altamente estruturados. A chave é considerar, cuidadosamente, as práticas de XP e implementá-las no ambiente correto". Além disto: "As práticas de XP podem ser compatíveis com as práticas do CMM (metas ou áreas-chave de processo), mesmo que elas não contemplem completamente o modelo".

A análise de Paulk foi feita sobre as KPAs e metas do CMM, componentes de mais alto nível do modelo. Uma avaliação para determinar o nível de maturidade de uma organização é realizada tendo como base, componentes mais detalhados: as práticas e subpráticas prescritas no modelo. O Capítulo 3 detalhará um dos métodos para determinação de nível de maturidade.

Apesar da conclusão favorável de Paulk a respeito do uso da metodologia ágil XP, o mesmo nem sempre acontece com análises dos envolvidos com metodologias ágeis sobre modelos de qualidade, em especial, sobre o CMM.

Segundo Jim Higsmith, "os níveis do CMM não são baseados em resultados, mas na implementação de um conjunto de práticas. A premissa básica – bons processos levam a bons resultados – é questionável e leva a muita introspecção sobre processos e

pouco foco nos resultados" [19]. Além disto, "...minha reação é que muito das atividades do CMM simplesmente não adicionam valor ao cliente" [19].

Bob Charette, criador da metodologia ágil Lean Development (LD) e *chairman* do comitê do IEEE para o "Padrão IEEE para Processos de Ciclo de Vida de Software – Gerenciamento de Riscos" considera que "LD nunca se adequará ao CMM" e também que "algumas pessoas no SEI estão preocupadas que as abordagens ágeis tornem o CMM largamente irrelevante" [19].

Em [39], Ron Jeffries et al, considera o CMM da seguinte forma: "sim, o CMM pode ser, está sendo, e será mal utilizado".

1.2 Objetivos do Trabalho

Este trabalho tem como objetivo responder através de um estudo aprofundado e de resultados obtidos em um estudo de caso às quatro perguntas da Seção 1.1.2 em relação ao modelo de qualidade CMM nível 2, e a metodologia ágil XP. Desta forma:

- É possível conviver em uma mesma organização com os formalismos requeridos pelo CMM nível 2 e o não formalismo pregado por XP?
- Empresas que buscam ou que são consideradas CMM nível 2, podem optar pelo uso de XP de forma que não comprometa a avaliação e/ou certificação alcançada?

Primeiramente será realizado um diagnóstico de satisfação do CMM nível 2 por XP (referenciado de Diagnóstico XP-CMM2 neste trabalho). Como já citado anteriormente, a análise realizada por Paulk e publicada no artigo "Extreme Programming from a CMM Perspective" [31] foi feita sobre as metas das KPAs do modelo. O diagnóstico proposto irá realizar esta análise sobre as práticas e subpráticas do modelo, conforme o método de avaliação *Software Capability Evaluation* (SCE), utilizado para atribuir nível de maturidade às organizações, detalhado no Capítulo 3.

Este diagnóstico irá relacionar as práticas do CMM nível 2 atendidas e não atendidas por XP. A pesquisa considera que se todas as práticas do CMM nível 2 forem atendidas por XP, então os formalismos requeridos pelo CMM e não requeridos por XP são completamente compatíveis.

- Os dois mundos, o mundo do CMM nível 2 e o mundo XP, podem coexistir em um mesmo ambiente?

Da mesma forma, se todas as práticas do CMM nível 2 forem atendidas por XP, então XP e CMM nível 2 podem funcionar ao mesmo tempo em uma mesma organização.

Caso o diagnóstico indique que algumas práticas do CMM nível 2 não são atendidas por XP, o trabalho indicará as alterações necessárias a XP de forma que seu uso não comprometa o nível 2 de maturidade de uma organização. Estas indicações correspondem ao Guia XP-CMM2, um guia de uso de XP em uma organização nível 2.

- É possível obter os benefícios propostos pelo CMM nível 2 e por XP?

Para avaliar os benefícios da adoção das duas abordagens XP e CMM um estudo de caso será realizado. Este estudo de caso deverá implantar o Guia XP-CMM2.

A Figura 1-1 ilustra a metodologia deste trabalho. Inicialmente é elaborado um diagnóstico de conformidade de XP ao nível 2 do CMM. Este diagnóstico é utilizado como base para elaboração de um guia, que deverá propor soluções para as não conformidades identificadas. Este guia, quando aplicado a XP possibilita que uma organização nível 2 do CMM utilize XP e obtenha os benefícios de ambas as abordagens.

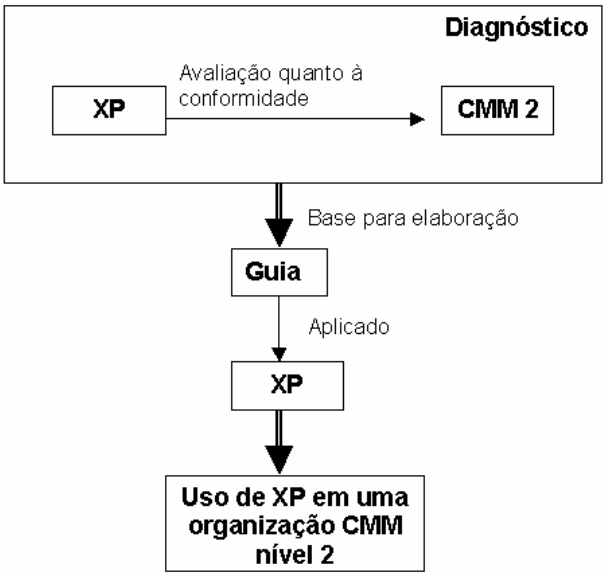

Figura 1-1 - Metodologia Aplicada para Desenvolvimento deste Trabalho

1.3 Estrutura da Dissertação

Este trabalho é organizado da seguinte forma:

- O Capítulo 2 descreve XP em detalhes;
- O Capítulo 3 detalha o modelo CMM: sua estrutura, requisitos, implantação e como as avaliações para aferir nível de maturidade às organizações são realizadas;
- O Capítulo 4 realiza um diagnóstico detalhado sobre a satisfação do CMM nível 2 pela metodologia XP;
- O Capítulo 5 propõe uma solução de uso de XP em uma organização CMM nível 2. Para cada item não atendido por XP no diagnóstico, é proposta uma alteração à metodologia;
- O Capítulo 6 apresenta um estudo de caso da solução proposta no Capítulo 5. Um projeto real utilizou a proposta e os resultados são apresentados;
- O Capítulo 7 apresenta a conclusão final deste trabalho.

2 Extreme Programming

Este capítulo aborda a metodologia ágil Extreme Programming. O capítulo está organizado como descrito abaixo:

- Seção 2.1 – Introdução: descreve o que é XP e restrições para usá-la;
- Seção 2.2 – A Estrutura de XP: descreve a estrutura de XP em termos de valores, princípios e práticas;
- Seção 2.3 – Um Projeto XP: descreve como funciona um projeto XP através da aplicação das práticas, valores e princípios;
- Seção 2.4 – Críticas a Extreme Programming: relata algumas críticas a XP;
- Seção 2.5 – Resumo: apresenta um breve resumo do Capítulo.

2.1 Introdução

Extreme Programming é considerada uma metodologia leve de desenvolvimento de software [25]. Beck e Fowler [24] classificam XP como um "sistema de práticas que a comunidade de desenvolvedores de software vem evoluindo para resolver os problemas de entregar software de qualidade rapidamente, e então alcançar as necessidades de negócio que sempre mudam" . XP surgiu a partir de idéias de Kent Beck e Ward Cunningham. A primeira vez que ela foi utilizada foi em um projeto piloto em março de 1996, do qual o próprio Beck fazia parte. Em 2000, Beck publicou o primeiro livro sobre XP, o "Exteme Programming Explained: Embrace Change" [25].

O "Extreme" do nome da metodologia se deve ao fato de ela empregar ao extremo boas práticas da engenharia de software [25]. Como exemplo, é importante ressaltar:

- Para implementar a boa prática da revisão de código, XP define que todo o código é desenvolvido por pares de programadores trabalhando juntos em uma mesma máquina;

- Para implementar a boa prática de testes, XP define que todos os testes são automatizados e executados várias vezes ao dia;
- Para implementar a boa prática de envolvimento do cliente, o cliente deverá estar no local de desenvolvimento, fazendo parte da equipe de um projeto XP.

XP não se aplica a todos os tipos de projetos, indica-se que seja utilizado em equipes de tamanho pequeno a médio, com no mínimo duas pessoas e no máximo dez [25]. No entanto, algumas pessoas defendem o uso de Extreme Programming em grandes projetos desde que eles sejam divididos em subprojetos independentes. Segundo Beck e Fowler, "XP endereça projetos longos quebrando-os em uma seqüência de mini projetos auto contidos, com duração de uma a três semanas" [24]. Outras restrições ao uso da metodologia são [25]:

- Cultura da documentação – ambientes em que a cultura, o gerente ou o cliente exigem especificações detalhadas e documentadas antes do início do desenvolvimento;
- Cultura de horas de trabalho para comprovar comprometimento – XP considera que duas semanas consecutivas de trabalho além das horas planejadas não deve ser visto como comprometimento, mas como problemas no projeto;
- Dificuldade para mudanças – se o software não pode ser mantido o mais simples possível e mudanças não podem ser realizadas freqüentemente (porque poderá impactar várias outras aplicações, por exemplo). Isto geraria a perda de flexibilidade, tão importante para um projeto XP;
- Ambiente onde é necessário muito tempo para obtenção de *feedback* – tecnologias que requerem bastante tempo para realizar a integração do software, por exemplo, irá ocasionar na demora para realização de testes e obtenção de *feedback* pela equipe;
- Ambiente em que não é possível realizar testes automáticos – a falta de testes sendo executados várias vezes ao dia dificulta a evolução do projeto de software e diminui a confiança da equipe no código produzido;
- Ambiente não propício a um projeto XP – pares de programadores devem trabalhar em conjunto em uma mesma máquina, a comunicação deve ser facilitada.

A metodologia Extreme Programming é de domínio público, o que facilita bastante a sua disseminação. Livros ([25], [39]), *web sites* ([53], [62]) e vários artigos ([4]) podem ser utilizados como fonte para entender e implantar XP.

2.2 A Estrutura de XP

Extreme Programming é definida através de valores, princípios e práticas [24], [25]. Os valores descrevem os objetivos de longo prazo de quem aplica XP e definem critérios para se obter sucesso. Os quatro valores de XP são [25]:

1. **Comunicação** – Parte do insucesso de alguns projetos de software é atribuído à falta de comunicação. Por isto, a metodologia emprega algumas práticas que forçam uma maior comunicação por parte da equipe, como programação em pares, por exemplo;
2. **Simplicidade** – Um dos lemas de XP é fazer "o mais simples possível que possa funcionar", ou "*the simplest thing that could possibly work*" como é conhecido entre os praticantes da metodologia. Para isto, deve-se desenvolver pensando apenas no presente. Um exemplo é que futuras funcionalidades a serem implementadas não devem ser utilizadas para determinar a arquitetura do software;
3. ***Feedback*** – Assim como o valor comunicação, XP estabelece várias práticas para que o *feedback* seja alto e ocorra o mais cedo possível. A equipe do projeto deve ter *feedback* a todo instante do cliente, do andamento do projeto, da qualidade do software. *Feedback* do código é obtido através de testes unitários e automáticos, que rodam todo o tempo. O *feedback* do cliente é estimulado através de *releases* curtos e entrega contínua de software de valor. O andamento do projeto é reportado a todos através de reuniões diárias e métricas simples, coletadas e reportadas freqüentemente;
4. **Coragem** – Coragem é uma virtude que os praticantes da metodologia devem ter. XP prega que pode acontecer – e a probabilidade não é baixa – momentos em que um programador deverá jogar fora todo o código trabalhado durante alguns dias porque o resultado final não foi o esperado. Outra situação naturalmente aceita é deparar-se com diferentes possibilidades de implementação e então implementar um pouco de cada para então escolher uma

dentre elas e passar a desenvolver a solução. Para estas situações é preciso coragem, segundo Kent Beck, em [25].

Para sustentar os valores e torná-los mais concretos, a metodologia define princípios que devem ser seguidos por todos os praticantes da metodologia. Eles devem guiar escolhas durante o andamento do projeto. Nestes casos, deve-se optar pela alternativa que mais fielmente atende aos princípios estabelecidos. Abaixo são listados os princípios básicos definidos por XP [25]:

- ***Feedback* rápido** – Além de se buscar *feedback* de tudo o que se produz, este *feedback* deve ser rápido;
- **Assumir simplicidade** – Os problemas devem ser tratados da forma mais simples possível. Não se deve buscar resolver um problema pensando no futuro ou na possibilidade de reuso. XP prega que o tempo ganho pensando desta forma, compensa o retrabalho que poderá surgir no futuro por não ter pensado em reuso inicialmente;
- **Mudança incremental** – Mudanças devem ser feitas pouco a pouco. Desta forma, mudanças no projeto do software ou no planejamento do projeto devem ser feitas de forma incremental e contínua;
- **Aceitar mudança** – Mudanças deverão ser sempre bem vindas, em qualquer momento do projeto;
- **Trabalho de qualidade** – Deve-se sempre buscar produzir um trabalho de qualidade, de outra forma a equipe perderá a motivação necessária ao projeto.

Outros princípios, considerados menos críticos, também são definidos por Extreme Programming [25]:

- **Ensinar aprendendo** – XP objetiva ensinar estratégias para que os praticantes da metodologia aprendam as próprias medidas do que projetar, de o quanto testar, de como fazer *refactoring*[1];
- **Investimento inicial pequeno** – Um projeto deve iniciar com poucos recursos e no decorrer, com o sucesso do projeto e *feedback* do cliente, ir aumentando estes recursos. É bastante citada durante a descrição da metodologia que um projeto

[1] *Refactoring* é a prática de alteração do código sem alterar sua funcionalidade, com o objetivo de melhorá-lo

pode ser cancelado ou ter o escopo diminuído, por exemplo. Nestes casos, é mais simples encerrar um projeto ou alterá-lo se a sua estrutura e investimentos ainda são pequenos;

- **Jogar para vencer** – A equipe deve ter o espírito de vencer, ou seja, de obter sucesso ao final do projeto;
- **Experimentos concretos** – Decisões abstratas devem ser testadas em forma de experimentos. Um exemplo de como isto deve acontecer em um projeto é após uma etapa de projeto de software, buscar implementar experimentalmente o modelo definido;
- **Comunicação aberta e honesta** – Problemas como atrasos, má qualidade do código produzido ou insegurança dos indivíduos, por exemplo, devem ser tratados abertamente pelo grupo do projeto;
- **Trabalhar a favor do instinto das pessoas e não contra** – Trabalhar com o instinto das pessoas, visto que estes são baseados em seus interesses de curto prazo, como vontade de vencer, aprender e interagir com outras pessoas;
- **Aceitar responsabilidades** – Responsabilidade deve ser aceita e não imposta. Assim, as pessoas realizam as atividades que mais gostam, as que acreditam poder realizar, tornando-se mais comprometidas com o projeto;
- **Adaptação ao local** – XP deve ser adaptada ao ambiente de trabalho em que será empregada, de acordo com a cultura e condições existentes;
- **Viajar leve** – O desenvolvimento deve gerar poucos e valiosos artefatos. A equipe estará a todo o tempo se adaptando às mudanças necessárias, como mudanças no escopo, nos requisitos, no projeto do software, a uma nova tecnologia e tantas outras. Por isto, deve estar "leve" de artefatos, documentações e especificações. De outra forma, seria requerido um esforço muito grande para manter as atualizações necessárias. O que há de valioso para o cliente e, conseqüentemente, para o projeto é o código e os testes. Outros documentos devem ser agregados se realmente forem importante ao projeto;
- **Medidas honestas** – As métricas a serem utilizadas devem ser o mais honestas possível e refletir as necessidades do projeto.

Extreme Programming define quatro atividades básicas de desenvolvimento de software, depois define as práticas para estruturar estas atividades de forma que as mesmas sigam os princípios, que sustentam os valores da metodologia. As atividades básicas de XP são: codificar, testar, escutar (o cliente e a equipe) e projetar. Abaixo são listadas as práticas definidas pela metodologia.

- **O jogo do planejamento** – Deve ser elaborado de forma fácil, rápida e simples o plano para o próximo *release*, que consiste basicamente em determinar o escopo baseado nas prioridades do negócio, nas possibilidades e estimativas técnicas;
- ***Releases* pequenos** – Os *releases* devem ocorrer em um espaço pequeno de tempo e devem conter sempre software de valor;
- **Metáforas** – Guiam o desenvolvimento do projeto através de uma estória que explique como o sistema funciona. Segundo Jeffries metáfora é "uma integridade conceitual baseada em uma simples metáfora que todos entendam, que explique a essência de como o programa funciona" [39];
- **Projeto de software simples** – O sistema deve ser projetado da forma mais simples que possa solucionar o problema;
- **Teste** – Há dois estágios de testes definidos: testes unitários e testes de aceitação. Ambos devem ser automatizados. Os testes unitários são feitos pelo programador durante o desenvolvimento. Os testes de aceitação são especificados pelo cliente e dita o que deve funcionar para que o software seja aceito;
- ***Refactoring*** – Reestruturação de parte do código, sem alterar o seu comportamento, visando simplificá-lo, torná-lo mais fácil de entender, remover duplicações;
- **Programação em pares** – Todo o código é produzido em pares de programadores trabalhando em uma mesma máquina;
- **Propriedade coletiva** – Todos os integrantes da equipe são donos e responsáveis pelo código produzido. Desta forma, todos podem alterar qualquer parte do código, a qualquer momento;

- **Integração contínua** – O código é integrado, o *build* do software é gerado e os testes são executados várias vezes ao dia, sempre que uma tarefa é completada;
- **40 horas semanais** – Os membros da equipe não devem trabalhar mais do que 40 horas semanais;
- **Cliente no local** – O cliente é integrante da equipe e deverá estar presente no local do desenvolvimento junto com todos os outros integrantes. Desta forma, ele estará sempre disponível para esclarecer dúvidas, escrever e priorizar estórias, especificar testes;
- **Padrão de codificação** – O código é utilizado como fonte de informação para comunicação e discussão entre os membros da equipe. Assim, ele deve estar bem escrito e estruturado, o que requer a conformidade a um padrão de codificação estabelecido para todo o projeto.

2.3 Um Projeto XP

Esta seção irá abordar como um projeto XP funciona: como e de que forma as práticas e valores definidos pela metodologia são empregados no dia-a-dia de um projeto.

Inicialmente, é definido o escopo do projeto. Depois disto há atividades para planejar o próximo *release* e a próxima iteração. A equipe parte, então, para a produção do software estabelecido para a iteração corrente. Antes de detalhar o funcionamento destas atividades, são listados os papéis e responsabilidades definidos por XP para o projeto.

2.3.1 Papéis

XP relaciona papéis, com responsabilidades explícitas, a serem desempenhados em um projeto. Os mais importantes são [25]:

- **Programador** – Segundo Beck "é o coração de XP". Responsável por projetar o software, codificar, implementar testes, estimar as suas tarefas;
- **Cliente** – Representante do cliente, responsável por estabelecer prioridades e escopo, escrever estórias, escrever os testes funcionais. Deve estar disponível no local do desenvolvimento para esclarecimentos de dúvidas dos programadores;

- **Testador** – O testador é responsável por apoiar o cliente a escolher e escrever os testes funcionais, além de assegurar a execução e reportagem dos problemas identificados nos testes funcionais;
- **Acompanhador (*tracker*)** – Segundo Beck "é a consciência da equipe". Responsável por reportar as métricas do projeto promovendo visibilidade sobre a acuidade das estimativas e progresso do projeto;
- **Técnico (*coach*)** – Este papel assimila grande parte das responsabilidades de um gerente de projeto de software. É responsável por identificar problemas e resolvê-los para que a equipe possa trabalhar da melhor forma. Não requer conhecimento técnico profundo.

2.3.2 Planejamento

- **Definição de viabilidade e escopo**

O primeiro passo de um projeto XP é determinar se o mesmo é viável e, neste caso, fechar o seu escopo. Neste momento, o pessoal técnico ainda não foi alocado ao projeto, o trabalho é de responsabilidade do gerente do projeto (o *coach*). [24].

Um plano inicial, chamado *Big Plan* é elaborado para determinar a viabilidade do projeto. Para isto, o gerente do projeto, em conjunto com o cliente, deverá identificar macro funcionalidades do projeto e estimá-las em termos de poucos meses. Poucas horas ou um dia é necessário para elaborar este plano que irá guiar o planejamento detalhado do projeto [24].

- **Planejando o release**

XP é contra a definição inicial de um plano detalhado, para todo o projeto. Como entende que vários fatores mudam durante o andamento do projeto, então, deve-se planejar para um período curto de tempo, e este planejamento deve ser alterado sempre que se mostrar inadequado. Desta forma, o planejamento deve ser feito apenas para o próximo *release*. Planejamento para todo o projeto pode ser feito desde que não inclua muitos detalhes e que todos tenham em mente que poderá ser bastante alterado [25].

Durante o planejamento do próximo *release*, se aplica a prática **"O jogo do planejamento"**. O planejamento ocorre em reuniões onde os participantes são o cliente – para responder por questões do negócio – e os desenvolvedores – para responder pelas questões técnicas.

O cliente leva para a reunião os requisitos do sistema, que XP chama de estórias, escritos em cartões de papéis, pois permitem facilidade de manipulação e armazenamento. Estes cartões, chamados de *Story Cards* pela metodologia, descreverem os requisitos de forma sucinta em poucas sentenças.

Os desenvolvedores devem então estimar a complexidade de cada estória. Para isto, durante a reunião, as estórias são apresentadas pelos clientes; os desenvolvedores fazem perguntas para melhor entendê-las. Os desenvolvedores acordam uma complexidade de desenvolvimento para cada estória. Esta complexidade está relacionada ao esforço para implementar cada estória.

O projeto deverá definir em que unidade realizar a estimativa das estórias: pode ser através de pontos de complexidade ou através de *Ideal Weeks* (bastante utilizado pela comunidade XP) – quantidade de semanas para implementar uma estória, considerando uma semana ideal de trabalho [24], [39]. Para estas estimativas, alguns conselhos são dados [39]:

- **Estimar em grupos** – não é adequado que destas reuniões participem apenas um representante do desenvolvimento. As estimativas são mais próximas da realidade se feitas em grupo;
- ***Spike Solution*** – quando não se tem idéia da complexidade de uma estória, deve-se realizar alguns experimentos através de implementação para só então realizar a estimativa;
- **Estimar baseado em estórias já implementadas –** utilizar o conhecimento passado para realizar as estimativas;
- **Estimar inicialmente as estórias mais simples –** deve-se dar pontos às estórias mais simples e estimar as outras através de comparação (se uma estória é considerada duas vezes mais complexa que uma simples, então deve possuir o dobro de pontos da mais simples).

Depois de estimada a complexidade de cada estória, então o cliente define o que deverá ser implementado para o próximo *release*, ou seja, o escopo do próximo *release*. Este escopo é baseado na velocidade da equipe, ou seja, na quantidade de pontos que a equipe consegue implementar em um intervalo de tempo. Esta velocidade é determinada pela quantidade de pontos que a equipe conseguiu implementar no passado. Se não há dados para determinar esta velocidade, então um número inicial deve ser considerado.

Assim, se a equipe possui uma velocidade de x pontos por *release*, então o cliente selecionará estórias, de forma que a soma de pontos de complexidade delas seja menor ou igual a x.

O planejamento do *release* é estruturado para que cada grupo de interessados interfira apenas sobre o que possui conhecimento. O cliente interfere fortemente sobre os requisitos e suas importâncias através da escrita das estórias e definição do escopo. Os desenvolvedores interferem sobre as questões técnicas definindo a complexidade das estórias e quanto é possível produzir em um intervalo de tempo.

Utilizando a prática ***"Releases* pequenos"***,* XP sugere como prazo ideal para liberação de *releases*, dois meses [39]. Neste prazo, deve ser entregue um software de valor, de forma que os usuários possam utilizá-lo diariamente. Mesmo sistemas grandes e complexos, podem ter partes independentes, importantes e de valor a serem entregues em um curto intervalo de tempo. Esta prática traz uma série de benefícios ao projeto devido ao *feedback* do cliente. Em pouco tempo (alguns meses) o cliente já poderá utilizar o software o qual está investindo dinheiro e com isto poderá informar a equipe de desenvolvimento sobre os aspectos positivos e negativos do software. Esta prática mostra a importância do valor *Feedback* definido pela metodologia.

- **Planejando a iteração**

Para facilitar o gerenciamento, guiar mais facilmente o projeto e diminuir riscos, o período até a data do *release* é dividido em iterações de poucas semanas – no máximo quatro, idealmente duas. Todas as iterações produzem software de valor. O cliente novamente é responsável por definir as estórias de cada iteração, no entanto aspectos técnicos podem ser considerados para priorizar a implementação das estórias que mais afetam a arquitetura do software. A seleção das estórias para uma iteração possui a mesma lógica da seleção para o *release*: o cliente deverá selecionar estórias onde a soma dos pontos de complexidade é menor ou igual à velocidade da iteração.

Para planejar uma iteração, deve ser realizada uma reunião onde o cliente comparece com as estórias. O cliente novamente apresenta as estórias e para cada uma delas os desenvolvedores listam, em conjunto, as tarefas que devem ser realizadas para implementá-las. As tarefas são as atividades de desenvolvimento propriamente ditas e são definidas em detalhes suficientes para prover o conhecimento necessário a sua implementação.

Os desenvolvedores, então, escolhem as tarefas que querem realizar e as estimam em termos de dias ou *perfect days* – dias ideais de programação. Assim, Extreme Programming permite que o responsável por realizar uma tarefa seja responsável também por estimar o tempo necessário para cumprí-la. Isto gera um maior comprometimento por parte dos desenvolvedores no cumprimento dos prazos. Outro resultado importante desta prática é a motivação obtida pela equipe após os próprios desenvolvedores escolherem o que desejam implementar.

A forma como os *releases* e as iterações são planejados colocam em prática outro valor: a Comunicação. Os requisitos (as estórias) não são descritos em muitos detalhes e nem formal ou estruturadamente, são descritos pelo próprio cliente, em cartões de papel, em algumas sentenças (é possível fazer referência a uma especificação em uma estória, se necessário), o suficiente para dar uma idéia do que se deseja obter do sistema. No momento de implementar as estórias, os programadores estarão comunicando-se entre si e com o cliente, que deverá estar disponível a todo o momento no local do desenvolvimento, conforme a prática **"Cliente no local"**.

2.3.3 Desenvolvimento

Após o planejamento do *release* e da iteração, a equipe passa a implementar as estórias selecionadas para a iteração corrente. É neste momento que a maioria das práticas definidas por XP é utilizada.

Algumas vezes, quando as pessoas ainda estão muito confusas sobre o que implementar, aconselha-se uma sessão rápida de *design* [39]. Nesta sessão, alguns integrantes da equipe se reúnem para acordar sobre um modelo de projeto. Devem ser sessões curtas, de dez a trinta minutos. O uso de CRC Cards [8] ou diagramas UML [29] é sugerido como facilitador para a reunião. Se realizadas, estas sessões de *design* devem ser guiadas pela prática **"Projeto de software simples"**, ou seja, deve-se projetar visando sempre um projeto de software o mais simples que possa resolver o problema atual, sem métodos, funções ou estruturas que não serão utilizadas no momento. Esta é tida como uma das práticas mais difíceis de serem utilizadas pela equipe. Isto porque, os programadores, especialmente os mais bem preparados, tendem a escrever pensando em reuso, em código elegante e nas várias práticas disseminadas pela Engenharia de Software ao longo dos anos. A prática **"Metáforas"** auxilia no projeto de software simples.

Apesar de não focar em um projeto de software elaborado, Extreme Programming não aprecia código e soluções deselegantes. Para evitar isto, insiste na

prática **"*Refactoring*"**. Em vários momentos do desenvolvimento, a equipe deve analisar o código produzido visando melhorá-lo. Neste momento, a equipe altera a estrutura do código, sem alterar a sua funcionalidade para remover duplicidade, melhorar *performance*, torná-lo mais simples.

Esta forma de trabalho – pouca ênfase na fase de projetar o software e alterá-lo sempre que necessário, é baseada em algumas premissas da metodologia. Primeiramente, Extreme Programming parte do princípio que requisitos podem mudar a qualquer momento, seja por melhor entendimento do cliente em relação ao produto a ser construído, por mudanças inerentes ao negócio do cliente, pelo *feedback* dos usuários após uso inicial do sistema. Por isto, não se deve gastar muito tempo projetando um software que pode ser completamente modificado. Deve-se projetar para o que foi estabelecido, para o que o cliente acordou e espera receber em alguns meses.

Outra premissa utilizada para sustentar as práticas "Projeto de software simples" e "Refactoring" como práticas a serem seguidas durante todo o desenvolvimento do software é a de que não é tão caro realizar mudanças no código como a Engenharia de Software tradicional prega. Há muitos anos se tem como consenso na comunidade de software de que quanto mais se demora a alterar uma parte de código, mais cara é esta alteração. A Figura 2-1 ilustra o consenso da Engenharia de Software, adotada também por Summerville [14].

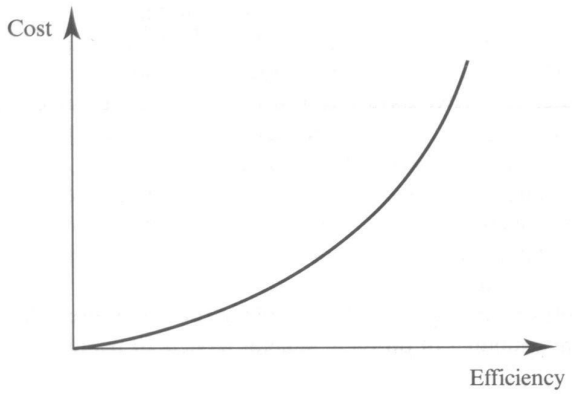

Figura 2-1 – Curva do Custo da Mudança

Pressman **[40]**, sugere valores para a realização da mudança de acordo com a fase de desenvolvimento do software, como mostra a Figura 2-2.

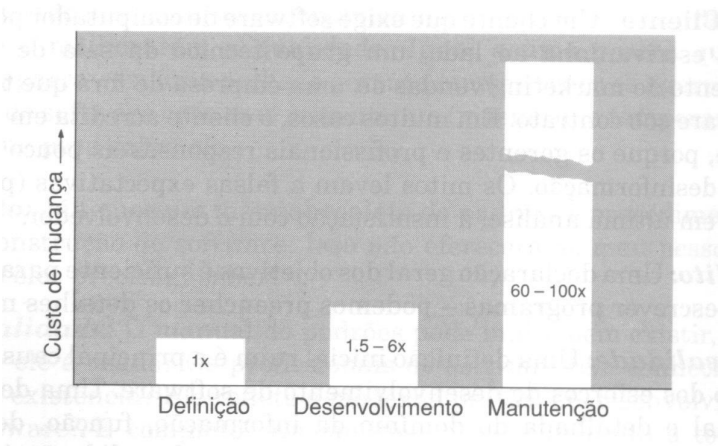

Figura 2-2 – Custo da Mudança no Tempo de Desenvolvimento

Para Extreme Programming [25], a mesma curva se comporta como sugerido pela Figura 2-3.

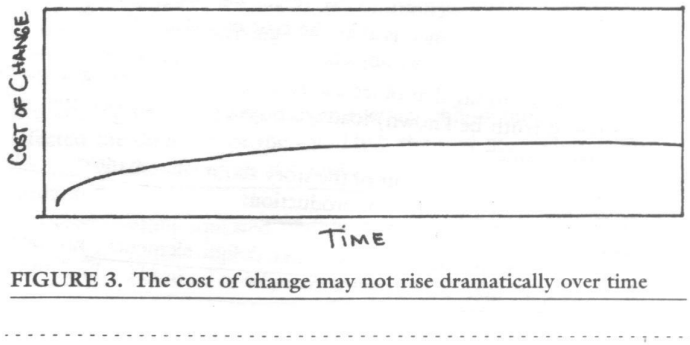

FIGURE 3. The cost of change may not rise dramatically over time

Figura 2-3 – Custo da Mudança Segundo XP

Os integrantes devem selecionar um par para trabalhar, pois todo o código produzido deve ser feito por duas pessoas trabalhando em conjunto em uma mesma máquina, desta forma se aplica a prática **"Programação em pares"** [25]. Os pares em XP não são fixos durante todo o projeto, eles devem mudar várias vezes, mesmo durante um dia de trabalho.

Extreme Programming define um pequeno processo para que o trabalho em pares seja efetivo. Nos pares, deverá sempre haver um *driver* – que está com o teclado – e um *partner* – que acompanha a produção do *driver*. O *partner* atua como inspetor analisando o que está sendo produzido em relação ao todo. O *driver* está mais focado com o que está programando no momento.

Algumas dicas podem ser levadas em consideração para se obter melhores resultados da programação em pares:

- O programador com menos certeza a respeito do que será produzido deverá ser o *driver*. Isto porque se o mesmo atuasse como *partner* a participação seria menor;
- Os pares devem trocar de papéis durante o trabalho.

Todo o trabalho de codificação dos pares deve ser feito utilizando um padrão de codificação estabelecido, conforme a prática **"Padrão de codificação"**.

A prática **"Teste"** define a realização dos testes unitários e de aceitação. Testes unitários são testes "caixa-branca", onde se busca testar o software linha a linha. Os testes unitários devem ser automáticos e desenvolvidos em conjunto com o software. XP fala de testar primeiro, ou seja, implementar os testes antes mesmo de implementar o software. A todo o momento, os testes devem estar funcionando com sucesso em sua totalidade. Esta forma de tratar os testes, motiva e gera mais confiança a respeito do código desenvolvido nos programadores.

Os testes de aceitação devem ser especificados pelo cliente, assim ele assume a responsabilidade de afirmar de que forma aceitará o produto que está sendo desenvolvido – aceitará se passar pelos testes especificados por ele. Assim como os testes unitários, os testes de aceitação devem ser automatizados e a equipe é responsável por implementá-los. Algumas vezes pode-se usar uma ou mais pessoas para apoiar o cliente na especificação dos testes e implementá-los. Estas pessoas assumem o papel de *Tester*.

Assim que um programador finaliza uma tarefa, deverá integrar o código produzido ao software já pronto. Desta forma, durante um dia, há vários *builds* do software sendo construído, conforme determina a prática **"Integração contínua"**. Sempre que um novo *build* é preparado, todos os testes devem ser executados. Se algum teste falhou, então o responsável pela integração (o programador que acabou a tarefa) é também o responsável por corrigir o problema, o qual pode estar localizado em um código produzido por ele ou por outro integrante da equipe. Ainda que o problema esteja em uma parte do software não produzida pelo responsável da integração, ele será o responsável por corrigi-lo, colocando em prática a **"Propriedade coletiva"**.

Para apoiar este processo de gerar vários *builds* diariamente, rodar os testes, controlar as alterações para a prática da "Propriedade coletiva", XP sugere fortemente o uso de ferramentas para tornar o uso destas práticas mais consistente e efetivo.

Ferramentas podem ser configuradas para gerar *builds* automáticos de tempos em tempos ou a cada novo código em um repositório, rodar testes e gerar relatórios a respeito dos testes.

As práticas utilizadas em conjunto direcionam para um desenvolvimento efetivo de software. As mudanças são mais baratas de implementar porque é mais fácil alterar código se este segue um padrão de codificação e os testes são automatizados e rodam várias vezes ao dia a cada novo código produzido, possível graças à integração contínua. Outro aspecto que deve ser salientado é que testes e integração contínua não dariam muito retorno efetivo se não houvesse a propriedade coletiva, pois se erros fossem sendo descobertos, mas só pudessem ser resolvidos quando o responsável estivesse disponível, possivelmente haveria um acúmulo de erros, o que poderia gerar, no mínimo, desorganização ao resto do desenvolvimento.

2.3.4 Acompanhamento

Para acompanhar o progresso da iteração e compará-lo ao planejamento, o Acompanhador (*tracker*) do projeto deve, algumas vezes por semana, perguntar a cada programador "Quantos dias ideais o programador trabalhou na tarefa que está realizando?" e "Quantos dias ideais o programador considera necessário para cumprir a tarefa?" [24]. Apesar de dias ideais não ter uma relação direta com os dias do calendário, o Acompanhador deverá considerar que algum problema está acontecendo se houver mais dias ideais a ser cumprido que os dias de calendário para completar a iteração ou se eles diferirem bastante. Se uma destas situações for detectada, então o Técnico (*coach*) deverá ajustar o problema dentro da equipe e, caso não seja possível, envolver o cliente para informá-lo.

Um outro apoio ao acompanhamento da equipe se dá através das chamadas *Stand-up Meetings*, ou seja, reunião em pé. Nesta reunião, que deve ocorrer diariamente, toda a equipe permanece em pé, de forma que contribua para que ela seja curta. Cada pessoa informa rapidamente o que fez no dia anterior e o que tem programado para realizar no dia de hoje. O objetivo desta reunião é comunicar os problemas e encaminhar as soluções. [24].

XP sugere o uso de gráficos visíveis para apoiar o acompanhamento do progresso do projeto, das estimativas e de problemas identificados. Os gráficos selecionados devem estar à vista de todos, preferencialmente colados nas paredes. Apesar de salientar que cada projeto deve medir o que poderá auxiliar a resolver problemas, alguns gráficos são sugeridos: Testes de aceitação definidos *versus*

executados, Densidade de *Bugs*, Progresso de estórias, Performance do sistema, entre outros [24].

2.3.5 Considerações Importantes

Extreme Programming relaciona alguns aspectos que, apesar de não fazer parte diretamente de um ciclo de vida de um projeto XP, são considerados críticos para seu sucesso. Um destes aspectos é o ambiente do projeto. Segundo Beck "Se você não tem um lugar para trabalho rezoável, seu projeto não terá sucesso" [25]. O ambiente para um projeto XP deve: ser amplo, com pequenos espaços privados e uma área de programação no centro. As mesas devem promover a programação em pares [25].

As comemorações planejadas por XP também contribuem para o sucesso do projeto. Beck cita em [25] "um projeto XP atrai brinquedos" e " Projetos XP sempre possuem comida em volta". Para Beck é dever do Técnico promover a aquisição de comidas e brinquedos. Em [39], Jeffries sugere comemorar a conclusão de tarefas e estórias com toda a equipe, pizza para comemorar o final de uma iteração e champagne para a entrega de um *release*, pois "o trabalho é divertido e gostoso quando há um sentimento de realização, de finalização. Estes momentos devem ser promovidos".

2.4 Críticas a Extreme Programming

2.4.1 Dificuldade em manutenção

A questão da manutenção de sistemas produzidos a partir de um projeto XP é bastante questionada quanto à sua eficácia devido à falta de documentação pregada pela metodologia.

Em [21] é relatada a dificuldade de prestar manutenção a um projeto XP. O código fonte e casos de testes não foram considerados suficientes para prestar manutenção. Além disto, percebeu-se que após três meses de trabalho vários detalhes, especialmente questões associadas aos requisitos do usuário, do projeto foram esquecidos pelos programadores.

2.4.2 Questões associadas à efetividade da programação em pares

Há várias questões associadas à efetividade da programação em pares:

- É possível realizar 100% da codificação de um projeto em pares, conforme sugerido por XP?

A programação em pares de 100% do código produzido é questionada por algumas equipes que adotaram a técnica. Em [36], Schuh afirma que a programação em pares tomou apenas 30% do tempo de desenvolvimento de um experimento realizado.

- Qual o custo associado à programação em pares?

Há alguns experimentos relacionados ao custo associado à programação em pares, em que diferentes conclusões são elaboradas.

Em [22] Nosek relata que a programação em pares consumiu 43% a mais do esforço total da programação individual. Nawrocki, em [21], afirma que nos experimentos realizados, a programação em pares exigiu um esforço 50% maior que a programação individual. Já Williams relata em [26] que os pares completaram o trabalho de 40% a 50% mais rápidos que se realizado individualmente.

Devido a falta de informações e embasamento científico e prático, muitas vezes, há resistência da gerência sênior das organizações em implantar programação em pares nos projetos [16].

- O benefício da programação em pares é o mesmo da inspeção?

Em [21] Nawrocki relata o uso de inspeções em um projeto XP. Esta prática foi utilizada pela dificuldade em produzir 100% do código através da programação em pares e também pelos ganhos de uma reunião de inspeção em que os erros encontrados são facilmente focados em relação a como corrigí-los, sem que culpas sejam atribuídas às pessoas.

- Como a equipe enfrenta programação em pares?

Há relatos de restrições quanto à programação em pares por parte dos programadores. Há pessoas que não desejam trabalhar em pares e há pares que não conseguem trabalhar em conjunto, seja por diferenças pessoais ou técnicas. Estes problemas foram enfrentados pela experiência realizada pela Poznam University of Technology **[57]**, em um projeto XP **[21]**.

2.4.3 Experiência da equipe

XP, assim como as metodologias consideradas ágeis, segue o princípio relatado pela Aliança Ágil de "Indivíduos e iterações ao invés de processos e ferramentas" [28]. Assim, grande parte das responsabilidades e ações é colocada para as pessoas dos projetos, ao invés de determinadas pelo processo.

Uma das questões que surgem em relação à XP é se o sucesso é completamente dependente da grande competência das pessoas que formam a equipe do projeto.

Em [21] é relatado o uso de XP em um projeto de desenvolvimento de software para um módulo acadêmico de dois anos da universidade Poznam University of Technology [57]. A equipe do projeto foi formada por estudantes do curso. O relato ressalta a dificuldade em implementar XP com equipes tecnicamente pouco experientes resultando em problemas relacionados à manutenção e comunicação da equipe.

2.4.4 Dificuldade associada ao cliente on site

Uma das práticas de XP é "Cliente no local", que coloca o cliente como integrante da equipe, presente no local do desenvolvimento durante todo o projeto [25]. O cliente no local é um dos pilares de sustentação das práticas de não detalhar requisitos (pois o cliente poderá tirar dúvidas a qualquer momento) e de não documentar decisões (pois o cliente estará tomando decisões durante o desenvolvimento do software).

A aplicação desta prática não depende apenas e diretamente da equipe do projeto, é preciso que o cliente disponibilize representantes dedicados a trabalhar no projeto, no local do desenvolvimento do mesmo. Além disto, algumas requisições a estes representantes são feitas, como: deve ser uma pessoa que realmente irá usar o sistema quando o mesmo entrar em produção, que possa estabelecer prioridades para o projeto e tomar decisões relacionadas ao escopo [25].

Nem sempre é possível atender a esta prática, por isto há diversos relatos de uso de XP onde o cliente não participa da equipe do projeto como sugerido [21]. O próprio Beck, relata em [25] um projeto em que ele participou onde, inicialmente, o cliente disponibilizado para o projeto não havia dedicação exclusiva. Somente após entrega com sucesso de uma parte do software, o cliente disponibilizou três representantes para atuar no projeto.

2.4.5 Dificuldade para estabelecer contrato de custo e tempo fixo e escopo variável

Segundo Beck [24], contratos que fixam as variáveis escopo, tempo e custo do projeto tendem a ter a qualidade comprometida. Uma abordagem melhor é estabelecer um contrato de custo e prazo fixos e escopo variável [24]. Um contrato como este firma que o projeto será desenvolvido com *x* recursos humanos, durante *y* meses a um preço *z*. Como exemplo, Beck cita em [25] "pelo valor de X, achamos que conseguimos produzir as seguintes estórias em Y meses". A equipe se compromete a trabalhar em

alta performance e o cliente tem a possibilidade de mudar a direção do projeto no início de cada iteração (através da adição ou remoção de estórias). Devido aos *releases* curtos, realizados em poucos meses, o cliente tem a possibilidade de cancelar o contrato se considerar a performance do projeto insuficiente ou se a projeto não fizer mais sentido para o negócio [25] .

Apesar das vantagens citadas por XP no estabelecimento de contratos deste tipo, nem sempre isto é possível de ser realizado. Mesmo sendo software uma entidade abstrata, o mercado, na maioria das vezes, define o preço para o produto que irá receber. Por isto, há uma expectativa natural de se tentar definir o escopo do projeto em conjunto com o valor a ser pago por ele.

Este tipo de contrato requer uma grande colaboração entre a equipe do projeto e o cliente. Se problemas ocorrerem e não puderem ser resolvidos dentro da equipe, deve-se recorrer ao cliente para que novo escopo seja definido. Exemplo disto se dá quando é identificado que não será possível implementar todas as estórias da iteração. Neste caso, o cliente é chamado para retirar alguma parte de uma estória planejada. O mesmo se dá se as estórias planejadas para o *release* não puderem ser cumpridas. A velocidade assumida pela equipe durante o planejamento do release também deve ser revista e repassada ao cliente, caso seja de fato menor.

2.5 Resumo

XP é baseada na aplicação de práticas, as quais promovem alguns valores considerados essenciais para um projeto de software. Os principais valores de XP são: comunicação, simplicidade, feedback e coragem. As doze práticas definidas por XP são: O jogo do planejamento, Releases pequenos, Metáforas, Projeto de software simples, Teste, Refactoring, Programação em pares, Propriedade coletiva, Integração Contínua, 40 horas semanais, Cliente no local, Padrão de codificação.

O planejamento de um projeto XP é simples, mas é de fundamental importância para o sucesso do projeto e se dá para os *releases* e iterações. O desenvolvimento de software é feito de forma bastante dinâmica favorecendo a comunicação entre a equipe e o *feedback* contínuo do funcionamento do sistema e da satisfação do cliente.

O acompanhamento em um projeto XP é realizado em curtos intervalos de tempos: métricas são constantemente colhidas e tornadas visíveis para todos os membros da equipe, reuniões curtas ocorrem diariamente, há um responsável para acompanhar o progresso das atividades de cada programador.

Apesar de vários relatos de sucesso no emprego de XP [4], [16], algumas críticas são relacionadas e ainda não completamente respondidas.

3 Capability Maturity Model for Software

Este capítulo descreve o *Capability Maturity Model* (CMM), utilizado como um dos fundamentos deste trabalho. O capítulo está organizado da seguinte forma:

- Seção 3.1 – Capability Maturity Model for Software: descreve como o CMM está estruturado e os aspectos mais relevantes do modelo;
- Seção 3.3 – Avaliação CMM: descreve como uma avaliação de maturidade com base no CMM é realizada;
- Seção 3.2 – CMMI: descreve em linhas gerais a evolução do modelo CMM, o CMMI;
- Seção 3.4 – Resumo: apresenta algumas considerações sobre este capítulo.

3.1 O Modelo Capability Maturity Model for Software

O *Capability Maturity Model for Software* (CMM) foi desenvolvido pelo Software Engineering Institute (SEI) [60], criado em 1984 pelo Departamento de Defesa dos Estados Unidos. O CMM é considerado um *framework* para melhoria dos processos de desenvolvimento de software das organizações [30]. O modelo é descrito de forma a relacionar atributos essenciais aos processos das organizações em determinados níveis de maturidade. Descreve "o que" é esperado, de forma genérica o suficiente para que as organizações definam "o como", por isto diferentes soluções podem ser adotadas para atender o modelo [30].

Atualmente O CMM se encontra na versão 1.1 e é possível encontrá-lo em livros ou até mesmo gratuitamente a partir da página do SEI na Internet [60].

3.1.1 A Estrutura do CMM

O CMM define um caminho a ser seguido para que as empresas saiam de um desenvolvimento de software *ad-hoc* e evoluam para um desenvolvimento maduro e efetivo. Este caminho engloba cinco níveis de maturidade. A Figura 3-1 ilustra a estrutura do CMM.

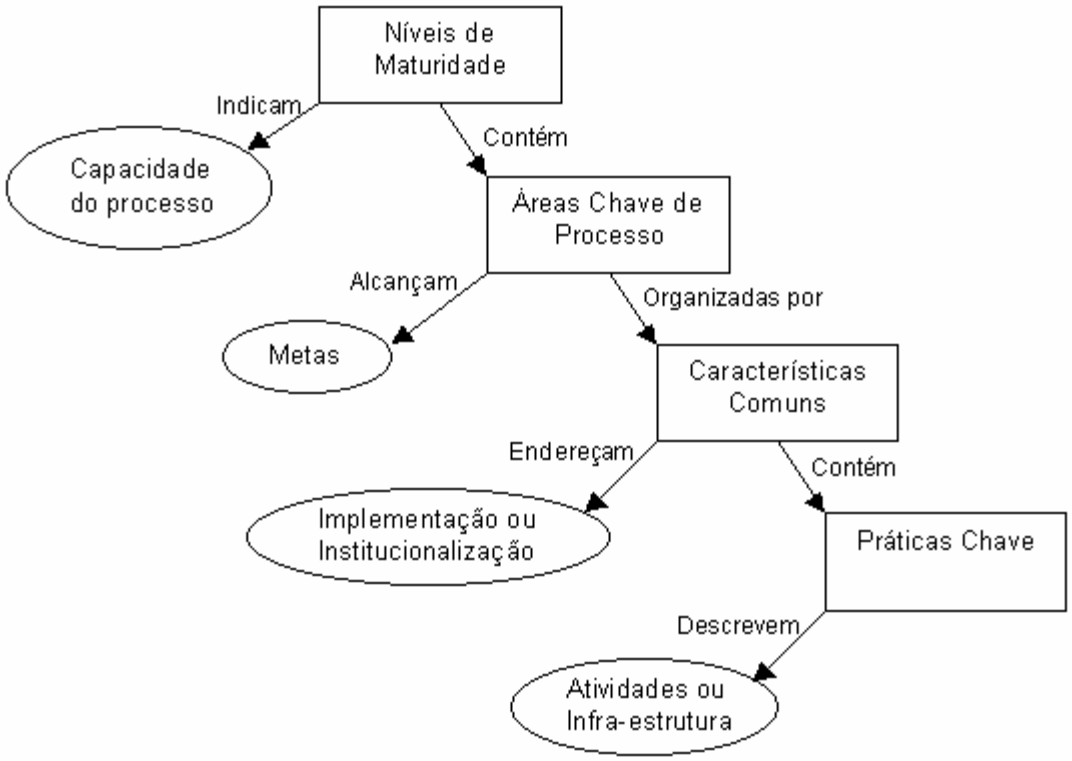

Figura 3-1 - Estrutura do CMM

Conforme a Figura 3-1, os níveis de maturidade indicam a capacidade do processo do software. Capacidade do processo é defina como sendo os resultados mais prováveis de serem obtidos pelos próximos projetos de uma organização [30]. Assim, quanto maior o nível de maturidade de uma organização, mais capaz ela é na produção de software.

Os níveis de maturidade, exceto o nível 1, possuem Áreas-chave de Processo ou *Key Process Áreas* (KPAs), que indicam onde uma organização deve se concentrar para

melhorar seu processo de desenvolvimento [30]. Para uma organização alcançar um nível de maturidade, ela deverá satisfazer todas as KPAs deste nível e dos níveis abaixo dele.

Para satisfazer uma KPA as metas definidas para ela devem ser alcançadas, o que implica que os processos definidos devem estar institucionalizados [30]. A definição destes processos pode variar entre empresas ou mesmo projetos, no entanto, as metas devem ser alcançadas.

Cada KPA só ocorre em um nível, ou seja, não há repetição de área-chave para níveis diferentes de CMM. A Tabela 3-1 lista as KPAs por nível de maturidade do CMM.

Nível de Maturidade	**KPAs**
1	Não há áreas-chave de processo definidas para o nível 1.
2	• Gerenciamento de requisitos • Planejamento de projeto de software • Acompanhamento de projeto de software • Gerenciamento de subcontrato de software • Garantia da qualidade de software • Gerenciamento de configuração de software
3	• Foco no processo da organização • Definição do processo da organização • Programa de treinamento • Gerenciamento de software integrado • Engenharia de produto de software • Coordenação de intergrupo • Revisões em pares
4	• Gerenciamento de processo quantitativo • Gerenciamento da qualidade de software
5	• Prevenção de defeito • Gerenciamento de troca de tecnologia • Gerenciamento de troca de processo

Tabela 3-1 - KPAs por Nível de Maturidade

O CMM não é um modelo exaustivo, ou seja, não trata de todos os processos relacionados ao desenvolvimento de software. Os processos tratados pelo modelo, são os que foram considerados determinantes para a capacidade ou maturidade do desenvolvimento de software **[30]**. Desta forma, outros fatores que agreguem valor e possam facilitar o desenvolvimento podem ser utilizados pelas organizações, mas não serão considerados para ditar a sua maturidade.

A Figura 3-1 mostra que cada KPA é organizada por características comuns que descrevem como institucionalizá-la ou implementá-la. As características são atributos que indicam quando a implementação ou institucionalização de uma KPA é efetiva, repetível e duradoura [30]. Estas características são:

- Compromisso para Realizar – descreve ações que a organização deve tomar para assegurar que o processo é estabelecido e irá durar. Tipicamente envolve definição de políticas, funções e responsabilidades para executá-las;
- Habilidade para Realizar – Pré-condições que devem existir para implementar o processo de forma efetiva. Tipicamente relaciona recursos, treinamentos e infra-estrutura;
- Atividades Realizadas – Atividades, papéis e procedimentos necessários para implementar a KPA. Envolve tipicamente o desenvolvimento de planos, procedimentos, realização de trabalhos, acompanhamento de atividades;
- Medição e Análise – Medidas para avaliar o status relacionado ao processo. São usadas para controlar e melhorar o processo. Tipicamente inclui exemplos de possíveis medidas que podem ser realizadas para a KPA.
- Verificar Implementação – Descreve passos para assegurar que as atividades da KPA estão sendo realizadas conforme definido. Incluem, em geral, revisões e auditoria de qualidade.

As características comuns agrupam práticas chave, que descrevem as atividades e infra-estrutura que mais contribuem para a efetiva implementação e institucionalização da KPA [30]. Estas práticas descrevem “o que pode ser feito” objetivando a conquista das

metas da KPA. As práticas são relacionadas às metas através de tabelas na descrição do modelo [30], assim, é descrito que práticas devem ser implementadas para conquistar cada meta de cada KPA.

É possível não implementar todas as práticas relacionadas pelo modelo ou implementar práticas alternativas. Nestes casos, experiência e julgamento devem ser aplicados para verificar se as práticas alternativas realmente contribuem para o alcance das metas da KPA [30]. As práticas chave possuem, geralmente, subpráticas mais detalhadas que podem ser utilizadas como guia na interpretação a implementação adequada.

Os componentes do modelo que pontuam a capacidade de uma organização em produzir software são os níveis de maturidade, as áreas-chave de processo e as metas [30]. Isto porque se diz que uma organização possui um nível de maturidade se ela satisfaz todas as áreas-chave de processo daquele nível e dos níveis abaixo. A satisfação de uma área-chave se dá se todas as suas metas são satisfeitas. Os outros componentes do CMM, como práticas chave e subpráticas, são componentes informativos e têm como objetivo guiar a implementação do modelo [30].

O CMM é descrito através de 500 páginas, definindo de forma hierárquica seus componentes como exibido na Figura 3-2:

5	Níveis de Maturidade
18	Áreas-chave de Processo
52	Metas
316	Práticas Chave
Vários	Subpráticas e exemplo

Figura 3-2 – O CMM em números

3.1.2 Os Níveis de Maturidade do CMM

A Figura 3-3 exibe os níveis de maturidade definidos pelo CMM.

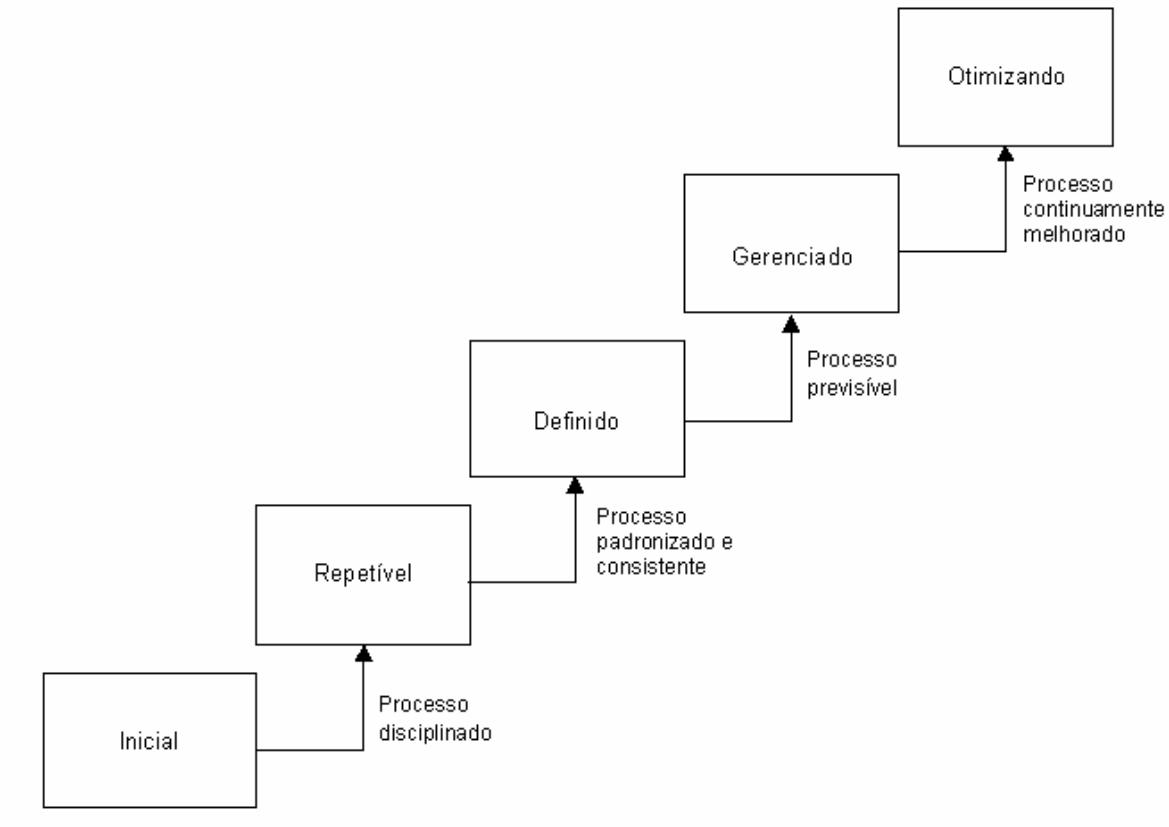

Figura 3-3 - Níveis de Maturidade do CMM

A Tabela 3-2 relaciona as principais características de cada nível de maturidade [30]:

Nível de Maturidade	Principais Características

1 (Inicial)	• Poucos (ou nenhum) processos definidos • Processos definidos não são seguidos em momentos de grande pressão • Geralmente projetos não cumprem prazo e orçamento • Equipes desestimuladas • Sucesso dos projetos fortemente dependente das pessoas • Comumente as pessoas trabalham mais do que a carga horária acordada
2 (Repetível)	• Há políticas para o desenvolvimento de software definidas e utilizadas por todos os projetos • Processo disciplinado: definido, documentado, praticado, treinado e medido • Processo pode variar entre projetos, mas obedecem as políticas da organização • Planejamento e acompanhamento de projetos se dá baseado em experiências passadas • Cronogramas, custos e riscos são planejados e acompanhados de forma efetiva • Os requisitos são gerenciados • Os produtos possuem controle de versões • Se há subcontrato associado ao projeto, este é gerenciado de forma a não comprometer o resultado final • Maior probabilidade de repetição de sucesso entre projetos sem dependência forte das pessoas

3 (Definido)	• Processo padrão para toda a organização é completamente definido, documentado • O processo padrão é utilizado por todos os projetos, no entanto adaptações são realizadas segundo critérios estabelecidos para satisfazer necessidades e restrições específicas • Um grupo de pessoas possui a responsabilidade de desempenhar as atividades relacionadas à criação, manutenção, melhoria e institucionalização deste processo padrão • A organização possui um programa de treinamento • Há um entendimento comum por toda a organização das atividades, papéis e responsabilidades definidas no processo padrão
4 (Gerenciado)	• A organização define objetivos quantitativos de qualidade para produto e processo • Projetos cumprem metas para atingir o padrão de qualidade definido pela organização • Há um banco de dados com informações extraídas de projetos que auxiliam na detecção e prevenção de problemas e no apoio a decisão. • Os projetos possuem resultados previsíveis • Devido às medidas realizadas nos projetos, é possível detectar e agir preventivamente para resolver problemas
5 (Otimizando)	• Foco na otimização de processos • Mudanças na tecnologia e processos são realizadas sem comprometer a qualidade do produto final • Problemas já resolvidos são disseminados para que não se repita

Tabela 3-2 - Principais características dos níveis de maturidade

O nível 2 de maturidade será detalhado no Capítulo 4, onde suas metas e práticas serão analisadas.

3.2 CMMI

Desde 1991, vários modelos CMM foram produzidos para áreas específicas como engenharia de sistemas, engenharia de software, aquisição de software, desenvolvimento integrado de produto e processo e outros. A melhoria dos processos das organizações foi afetada negativamente com a adoção de diferentes modelos, pois gera um aumento de complexidade e custos, devido a diferentes abordagens e arquiteturas destes modelos, necessidade de vários treinamentos, e diferentes avaliações [6].

Motivado por isto, o Departamento de Defesa dos Estados Unidos patrocinou o projeto de desenvolvimento do *Capability Maturity Model Integration* (CMMI) *Framework*, conjunto de modelos, métodos de avaliação e produtos de apoio, elaborado pelo SEI, representantes da indústria e do governo. O CMMI teve como objetivo combinar os modelos *Capability Maturity Model for Software* (CMM) v2.0 *draft* C, *Electronic Industries Alliance Ínterim Standard* (EIA/IS) 731 e *Integrated Product Development Capability Maturity Model* (IPD-CMM) v0.98 em um único *framework* a ser utilizado por toda a organização [6]. O CMMI foi lançado recentemente e deverá substituir o CMM dentro de alguns anos [60].

A arquitetura do CMMI é parecida com a do CMM, no entanto, existem pequenas mudanças. O CMMI possui duas representações: contínua e por estágios. A representação por estágios mede a melhoria de processos através de níveis de maturidade [60], assim como o CMM. Já a representação contínua mede a melhoria através de níveis de capacidade aplicadas a áreas de processo individuais [60].

3.3 Avaliação CMM

Para que uma organização seja avaliada a respeito da conformidade ao CMM, o SEI definiu um *framework* para documentar os requisitos necessários e características desejáveis a um método de avaliação baseado no CMM, chamado *CMM Appraisal Framework* (CAF) [34]. O CAF serve como base para comparação dos resultados de diferentes métodos de avaliação aumentando a consistência e confiabilidade entre eles [34]. Métodos de avaliação do modelo CMM são reconhecidos pelo SEI se forem compatíveis com o CAF, ou seja, implementarem todos os requisitos determinados por ele.

O SEI possui atualmente dois métodos de avaliação compatíveis com o CAF [34]:

1. *Software Capability Evaluation* (SCE): utilizado para selecionar fornecedores, monitorar contratos e para avaliar o processo interno da organização. Encontra-se atualmente na versão 3.0;
2. *CMM-Based Appraisal for Internal Process Improvement* (CBA IPI): o CBA IPI é mais específico em seus objetivos: é utilizado apenas para avaliação do processo interno da organização.

Os resultados de avaliações utilizando o CBA IPI e o SCE, conduzidas sobre as mesmas áreas de processo em uma organização, devem ser consistentes entre si [34]. Este trabalho irá apresentar o método de avaliação SCE, pois além de ser equivalente ao CBA IPI, é mais genérico. Alguns fundamentos do SCE serão utilizados no Capítulo 4 para realizar o Diagnóstico de satisfação de XP ao CMM nível 2.

3.3.1 Software Capability Evaluation (SCE)

Como objetivos, o método SCE deve ser confiável, repetível, treinável, consistente e alinhado ao CBA IPI [34]. O SCE foi elaborado para avaliar o processo da organização visando determinar a sua capacidade. O método SCE é composto por três fases: "planejar e preparar a avaliação", "conduzir a avaliação" e "reportar resultados da avaliação", conforme descrito nas próximas subseções.

3.3.1.1 Planejar e preparar a avaliação

Inicialmente, os requisitos da avaliação são identificados e a organização que contratou a avaliação define seus objetivos: selecionar fornecedor, avaliar subcontrato ou avaliar o processo interno. Além disto, outras decisões são tomadas, como:

- Escopo CMM da avaliação: parte do modelo que será usado como referência na avaliação: se a avaliação cobrirá apenas alguns processos do CMM (tipo de avaliação que não atribui nível de maturidade), ou todos os processos de um determinado nível de maturidade, selecionado para ser avaliado.
- Escopo organizacional: projetos da organização que serão avaliados.

A partir destas definições, um plano de avaliação é elaborado para guiar a sua execução. Este plano inclui: propósito, escopo, atividades, cronograma, membros da

equipe de avaliação, recursos, financiamento, logística (aspectos como viagens, hospedagens, local da avaliação) e riscos. O plano é então revisado pela alta gerência da organização.

Para conduzir uma avaliação SCE, uma equipe avaliadora deve ser selecionada, a qual é composta, em geral, por cinco ou seis avaliadores, sendo um deles, o líder [34]. Algumas restrições são impostas a esta equipe [34]:

- A equipe deve possuir somados, no mínimo, 25 anos de experiência técnica e 10 anos de experiência em gerenciamento;
- Pelo menos duas pessoas da equipe deve ter participado de avaliações SCE;
- Nenhum membro deve ter menos de 5 anos de experiência profissional;
- O líder deve ter participado em no mínimo duas avaliações SCE;
- Todos os avaliadores devem ter sido treinados no método SCE.

Para que uma organização receba um laudo do SEI [60] reconhecendo seu nível de maturidade, a avaliação deve ser conduzida por um avaliador líder credenciado pelo próprio SEI. Atualmente, no Brasil, duas pessoas possuem esta credencial e ambos são consultores da empresa ISD Brasil [55].

Alguns instrumentos podem ser utilizados para se obter informações prévias a respeito da organização. Um exemplo destes instrumentos é o questionário de maturidade [9], elaborado pelo SEI. Este questionário é enviado e a organização a ser avaliada responde-o e o envia para que a equipe avaliadora o utilize como fonte para detalhamento do planejamento da avaliação.

Uma outra atividade desta fase é a seleção e preparação das localidades, projetos e participantes [34]. É preciso selecionar as localidades (unidades) e projetos da organização, pois dependendo do tamanho da organização, não é possível avaliar todos os seus projetos e localidades. Esta seleção deve ser feita considerando a representatividade dentro da organização: se os projetos e/ou localidades selecionados representam bem, por exemplo, o negócio, o tamanho, o faturamento. As pessoas a participarem da avaliação também deverão ser selecionadas de acordo com o tamanho da organização, características das pessoas da organização e atividades desempenhadas [34].

A equipe de avaliação então conduz uma explanação aos participantes selecionados a respeito da avaliação, visando assegurar que eles saibam o que é o processo de avaliação e porque ele será realizado [34]. Além disto, se preparam para a coleta de dados realizando os seguintes passos: organizam as entrevistas a serem realizadas, relacionam os documentos a serem revisados, definem papéis e responsabilidades da equipe avaliadora (líder, monitor para cada KPA, bibliotecário, entre outros).

3.3.1.2 Conduzir a avaliação

A condução da avaliação é iniciada com uma apresentação da própria organização sobre: o que ela faz, sua estrutura organizacional, documentos da organização (políticas e procedimentos e outros) e como os grupos envolvidos com o desenvolvimento são gerenciados e integrados [34]. A equipe avaliadora deve escutar, questionar e tomar nota de aspectos importantes desta apresentação [34].

Três atividades são realizadas repetitivamente durante a condução da avaliação: revisão de documentos, condução de entrevistas e consolidação dos dados. Elas são realizadas em diferentes sessões, devidamente planejadas e comunicadas aos envolvidos:

- **Atividade revisão de documentos**

  Na revisão de documentos, a equipe avaliadora analisa três níveis de documentação: documentos no nível da organização, nível do projeto e nível de implementação em dois tipos de revisões: inicial e detalhada.

  Uma revisão inicial é realizada sobre os documentos do nível da organização (políticas, processos da organização) e do projeto (processos específicos dos projetos) visando identificar como eles definem e suportam o processo da organização e as KPAs que estão sendo investigadas **[34]**.

  Uma revisão detalhada é realizada sobre os documentos do nível do projeto e de implementação (registros gerados pelo projeto, como planos, atas de reuniões, estimativas) para validar as informações levantadas nas entrevistas e em outras revisões de documentos. O objetivo é encontrar evidências de como os processos são realmente implementados e se estão de acordo com o que foi dito em entrevistas [34].

- **Atividade condução de entrevistas**

A equipe de avaliação irá conduzir entrevistas para coletar dados a respeito do processo definido e seu uso pela organização [34]. Estas entrevistas podem ser individuais ou em grupo, conduzidas em diferentes sessões, onde: entrevistas com gerentes, em geral, são realizadas individualmente e com praticantes (pessoas que não atuam no nível gerencial) são realizadas em grupo, sem a presença de gerentes para evitar constrangimentos e cobranças não pertinentes à avaliação [34].

O método SCE considera dois tipos de entrevistas: exploratórias e de consolidação. A entrevista exploratória tem como objetivo prover conhecimento sobre como as KPAs estão sendo realmente implementadas pelos projetos. A entrevista de consolidação tem como principal objetivo confirmar ou esclarecer alguma informação ainda não obtida seguramente pela equipe de avaliação. Esta última pode ser conduzida por um subconjunto de avaliadores e utiliza perguntas mais específicas para esclarecer as questões pendentes.

Durante as entrevistas toda a equipe avaliadora deverá anotar todas as informações obtidas. Estas notas não devem conter julgamento do avaliador, mas informações obtidas nas entrevistas [34].

- **Atividade consolidação de dados**

Os dados coletados deverão ser consolidados para gerar observações formais da equipe de avaliação, que identificam pontos fortes, fracos e oportunidades de melhoria em relação ao CMM [34]. Estes dados são provenientes dos registros feitos pela equipe de avaliação a partir da análise de instrumentos, apresentações da organização, revisão de documentos e entrevistas [34].

As observações elaboradas são realizadas em relação às práticas chave definidas pelo modelo. Para transformar a análise de documento e registros de entrevistas em observações formais sobre a conformidade ao modelo, algumas vezes é preciso julgar com base nas informações coletadas. O método SCE define algumas regras para que este julgamento seja o mais objetivo possível e para que não dependa de opiniões individuais.

Primeiramente, todas as observações devem ser realizadas sobre informações corroboradas, segundo as seguintes regras [34]:

- Uma observação deve ser baseada em uma evidência objetiva na forma de documento, a não ser se define um ponto fraco, onde a falta de documentação é considerada evidência objetiva;
- Quando a evidência provém de entrevistas, apresentações e análise de documentos esta deve ser observada em duas sessões independentes;
- As observações formais devem ser geradas através de consenso da equipe avaliadora. Se a equipe não entrar em consenso sobre algum tópico, então deverá relacionar informações adicionais necessárias para resolver a questão e elaborar um plano para capturar estas informações adicionais.

O método não define regras definitivas a respeito da corroboração e sugere que quanto mais crítica for a observação, maior a necessidade de corroborar as informações com outras fontes. Segundo o método, se houver qualquer dúvida a respeito da observação, então uma nova coleta de dados deverá ser promovida [34]. Além disto, considera importâncias diferentes para diferentes tipos de coleta de dados. Como exemplo, sugere que se há uma observação que pode comprometer uma KPA, vinda de análise de instrumento, então a mesma deve ser melhor investigada através de outras fontes [34].

Para cada observação gerada, a equipe de avaliação deverá julgar se está clara; se é baseada em fatos ou fortes inferências; se foi suficientemente corroborada; se é categorizada contra o CMM (se está relacionada a alguma prática chave); se está classificada como sendo um ponto forte, fraco ou oportunidade de melhoria; se é relevante e significativa; se não é redundante nem contraditória em relação às outras observações; se as observações cobrem tudo o que está sendo pesquisado [34].

Depois que as atividades de revisão de documentos, condução de entrevistas e consolidação de dados foram realizadas repetidamente conforme planejado, duas outras atividades ainda são realizadas: Apresentar resultados iniciais e realizar julgamento de pontuação.

- **Atividade apresentar resultados iniciais**

O método SCE busca assegurar que injustiças não sejam feitas à organização que está passando pela avaliação. A atividade de apresentar resultados iniciais comprova esta afirmação. Nesta atividade, a equipe de avaliação apresenta os pontos fracos aos participantes da organização em diferentes sessões. Nesta apresentação, a organização tem a oportunidade de refutar o resultado, apresentando informações e evidências. A equipe de avaliação anota todas as novas informações obtidas para validar o que foi apresentado e/ou rever os pontos fracos [34].

- **Atividade realizar julgamento de pontuação**

Finalmente, a equipe de avaliação irá pontuar os componentes selecionados durante o planejamento da avaliação [34]. A pontuação é feita de baixo para cima, ou seja, os componentes de mais baixo nível na estrutura do CMM são pontuados primeiro e utilizados na pontuação dos componentes de mais nível [34]. Desta forma, para que uma organização seja classificada em um nível de maturidade, a equipe deverá pontuar inicialmente as metas, depois as KPAs e depois o nível de maturidade.

Os valores aplicados na pontuação aos componentes são [34]:

- Satisfeito: se o componente foi implementado e institucionalizado como requerido pelo modelo ou através de uma alternativa que atenda a intenção do componente;
- Não satisfeito: se há pontos fracos significativos associados ao componente comprometendo a sua implementação e/ou institucionalização;
- Não aplicável: se o componente não se aplica à realizada da organização. Um exemplo de quando um componente é pontuado como não aplicável é a KPA Gerência de Subcontrato do nível 2, quando a organização não subcontrata;

– Não pontuado: um componente não é pontuado se a avaliação não cobriu os requisitos para pontuação ou se não está no escopo da avaliação.

Para pontuar as metas a equipe de avaliação irá julgar se a meta foi implementada e institucionalizada pela organização [34]. Para verificar isto, as observações registradas e já validadas com a organização são utilizadas.

Se todas as práticas relacionadas a uma meta possuem apenas pontos fortes associados, a meta é alcançada. No entanto, é possível que uma meta seja atingida mesmo tendo pontos fracos associados, desde que eles não sejam críticos [34], apesar de o método não determinar objetivamente o que é considerado crítico, este julgamento fica a cargo da equipe avaliadora. É possível que metas sejam alcançadas através da implementação de práticas alternativas [34], ou seja, práticas implementadas pela organização, diferentes de práticas chave descritas no CMM. Também fica a critério da equipe avaliar se a prática alternativa atinge a intenção por trás da prática chave não implementada, não comprometendo neste caso, a meta relacionada. Assim, a pontuação das metas é uma das atividades mais subjetivas do método.

A pontuação das KPAs e do nível é mais objetiva e segue o critério já citado anteriormente: uma KPA é satisfeita se todas as suas metas são satisfeitas; um nível é alcançado se todas as KPAs aplicáveis do seu nível e de níveis inferiores são satisfeitas [34].

3.3.1.3 Reportar resultados da avaliação

A avaliação é finalizada após apresentação do resultado final e entrega de relatórios. Primeiramente a equipe apresenta para a organização os resultados obtidos: pontos fortes, fracos, oportunidades de melhoria, problemas encontrados (mesmo os não relacionados com o modelo de referência) [34]. O método encoraja a participação de todos os interessados da organização e não apenas dos que participaram diretamente [34].

Nesta apresentação a equipe de avaliação deverá estar preparada para responder a eventuais questionamentos, respeitando os princípios de não atribuição e confidencialidade [34] definidos pelo método, ou seja, os problemas encontrados não podem ser divulgados para a organização e não são atribuídos a pessoas e/ou projetos.

A equipe de avaliação ainda é responsável por elaborar e entregar relatórios formais do trabalho realizado. Eles incluem: pontos fracos, pontuação, riscos associados à valiação, recomendações ao uso dos resultados, informações a respeito da avaliação e outros [34]. Templates para os relatórios são disponibilizados no curso oficial do SCE do SEI [41].

3.4 Resumo

Este capítulo apresentou o CMM, sua estrututra, níveis de maturidade, áreas-chave de processo e características comuns. Além disto, abordou os principais aspectos de um dos métodos de avaliação do CMM elaborado pelo SEI, O SCE. Os fundamentos do SCE foram utilizados como base para elaboração do Diagnóstico XP-CMM2, descrito no Capítulo 4.

O CMM deverá ser substituído no médio prazo pelo CMMI. A representação por estágios do CMMI promove uma fácil transição do CMM para o CMMI.

4 Diagnóstico XP-CMM2

Este capítulo irá determinar a aderência de XP ao CMM nível 2. O objetivo é identificar as práticas definidas pelo modelo que são e as que não são satisfeitas por XP. Este capítulo está organizado da seguinte forma:

- Seção 4.1 – O Diagnóstico: descreve os objetivos, escopo, metodologia e nomenclatura utilizada neste diagnóstico;
- Seção 4.2 – Mapeamento de XP no CMM: mapeia termos utilizados em XP no CMM;
- Seção 4.3 – Diagnóstico da KPA Gerenciamento de Requisitos: realiza o diagnóstico para a KPA Gerenciamento de Requisitos;
- Seção 4.4 – Diagnóstico da KPA Planejamento de Projetos de Software: realiza o diagnóstico para a KPA Planejamento de Projetos;
- Seção 4.5 – Diagnóstico da KPA Acompanhamento de Projetos de Software: realiza o diagnóstico para a KPA Planejamento de Projetos;
- Seção 4.6 – Diagnóstico da KPA Garantia da Qualidade de Software: realiza o diagnóstico para a KPA Planejamento de Projetos;
- Seção 4.7 – Diagnóstico da KPA Gerência de Configuração de Software: realiza o diagnóstico para a KPA Planejamento de Projetos;
- Seção **Erro! Fonte de referência não encontrada.** – **Erro! Fonte de referência não encontrada.**: apresenta uma breve conclusão deste capítulo.

4.1 O Diagnóstico

Como já apresentado no Capítulo 3, os componentes normativos do CMM são os níveis, KPAs e metas. Estes componentes contribuem na definição da capacidade de uma organização. Práticas e sub-práticas são componentes informativos que auxiliam no entendimento do CMM. Cada prática está relacionada a uma ou mais metas, de forma que elas ajudam a alcançá-las. Apesar de sua implementação não ser considerada obrigatória para alcance de um nível, as práticas são, em geral, aplicáveis. Avaliações, como o SCE, fazem observações relacionadas às práticas e sub-práticas (o Capítulo 3 fornece maiores detalhes).

Práticas não implementadas, se aplicáveis, devem dispor de alternativas de implementação para satisfazer às metas relacionadas. Em avaliações, tipicamente de 3 a 5 práticas alternativas são utilizadas para atingir a meta [33], ou seja, considerando o maior número, apenas 1,59% das 316 práticas do CMM são consideradas alternativas; todas as outras são consideradas aplicáveis. Segundo Paulk, "Práticas-chave não são requeridas e deve existir implementações alternativas, mas isto não tira a responsabilidade de realizar julgamentos profissionais sobre cada prática e meta associada" [33].

Devido à importância das práticas para entendimento do CMM e alcance das metas, um diagnóstico de satisfação será realizado contra as práticas do nível 2, pois se as práticas forem satisfeitas, então as metas também serão.

4.1.1 Escopo do Diagnóstico

Este diagnóstico irá focar apenas no Nível 2 do CMM, que inclui as áreas-chave de processo: Gerenciamento de Requisitos, Planejamento de Projeto de Software, Acompanhamento de Projeto de Software, Gerenciamento de Subcontrato de Software, Garantia da Qualidade de Software e Gerenciamento da Configuração de Software.

Dentre estas KPAs, o objetivo da Gerenciamento de Subcontrato de Software é selecionar subcontratados qualificados e gerenciá-los de forma eficaz [30]. Esta KPA não será considerada neste trabalho, pois foge ao escopo de uma metodologia de desenvolvimento de software, que foca no ciclo de desenvolvimento do software.

O CMM é um modelo para construir capacidade organizacional na produção de software e foca em questões de gerenciamento envolvidas na implementação de

processos efetivos e eficientes e na melhoria de processos sistemática [31]. O CMM diz "o que", em termos gerais, deve ser feito. O "como fazer" é de responsabilidade da organização, que deve definir ou utilizar um processo para isto [31]. Além de abordar aspectos técnicos do desenvolvimento de software, o CMM aborda também aspectos organizacionais visando institucionalização [31], como já mencionado no Capítulo 3.

Metodologias de desenvolvimento de software poderão complementar o CMM para estabelecer o "como fazer" dos aspectos técnicos relacionados pelo modelo, que ditam "o que" deve fazer. Os aspectos relacionados à institucionalização ou à organização, muitas vezes fogem ao escopo destas.

Um exemplo de aspecto técnico definido pelo CMM é a prática **"O plano para o projeto de software é documentado"**, referente à área chave de processo Planejamento de Projeto de Software. Metodologias de desenvolvimento de software, em geral, definem como se dá o planejamento e a documentação do mesmo para projetos de software. Desta forma, o processo utilizado irá definir o "como fazer" o plano do projeto e documentá-lo, contemplando a prática citada.

Um exemplo de prática definida pelo CMM relacionada a aspectos de institucionalização é **"O projeto segue uma política organizacional escrita para planejar um projeto de software"**, também referente à área chave de processo Planejamento de Projeto de Software. Esta é uma atividade relacionada à organização, não possuindo relação direta com o desenvolvimento de um projeto de software. Em geral, atividades como esta não são mencionadas pelas metodologias de desenvolvimento de software.

No CMM, as características comuns "Compromisso para Realizar", "Habilidade para Realizar", "Medição e Análise" e "Verificar Implementação" agrupam práticas-chave relacionadas à institucionalização. As práticas de implementação, relacionadas à característica comum "Atividades Realizadas", descrevem o que deve ser feito para estabelecer a capacidade do processo de software adotado [30].

Desta forma, este diagnóstico irá considerar apenas as práticas da característica comum "Atividades Realizadas", analisando para cada uma, se a metodologia *Extreme Programming* a atende e, neste caso, como isto é feito. Caso não atenda, o risco associado ao problema detectado será identificado.

4.1.2 Metodologia para Realização do Diagnóstico

Para realizar o diagnóstico a seguinte abordagem foi utilizada: cada prática de implementação do CMM nível 2 foi analisada e uma pontuação lhe foi atribuída, a qual representa a satisfação da prática pela metodologia XP. Os valores desta pontuação foram baseados nos valores determinados pelo método de avaliação de capacidade SCE (o Capítulo 3 fornece maiores detalhes), no entanto, algumas alterações foram realizadas para atender ao objetivo deste diagnóstico. Estes valores são:

- Satisfeito: se a prática foi completamente satisfeita por XP conforme requerido pelo modelo ou através de uma alternativa que atenda a intenção da prática.
- Não satisfeito: se há pontos fracos significativos associados à prática comprometendo a sua implementação.
- Parcialmente satisfeito: se há pontos fracos não significativos associados à prática.
- Não aplicável: se o componente não se aplica ao escopo deste trabalho.

Determinar a satisfação de cada prática requer a realização de julgamento. Em avaliações, este julgamento é realizado por profissionais competentes, treinados e experientes que definem processos [33]. De 10 a 15% das práticas do CMM exigem interpretação, ou seja, a equipe de avaliação deve discutir a implementação da prática até que se chegue a um consenso [33]. Devido a este fator de interpretação, Paulk reconhece que "É verdade que um julgamento pode diferir – e algumas vezes isto acontece" [33].

Assim como avaliações oficiais, este diagnóstico é resultado de algumas interpretações e julgamentos, de forma que pode diferir de outros diagnósticos se forem realizados por outras pessoas. As premissas utilizadas para determinar a satisfação das práticas serão documentadas para justificar o resultado.

4.1.3 Nomenclatura Utilizada no Diagnóstico

Algumas siglas são comumente utilizadas para referenciar o CMM e também serão utilizadas neste trabalho. A Tabela 4-1 relaciona as siglas utilizadas para as áreas-chave de processo do CMM nível 2. A Tabela 4-2 lista as siglas utilizadas para referenciar as características comuns do modelo.

Sigla	Descrição
RM	Do inglês *Requirements Management*, referente à KPA Gerenciamento de Requisitos.
SCM	Do inglês *Software Configuration Management*, referente à KPA Gerenciamento de Configuração de Software.
SPP	Do inglês *Software Project Planning*, referente à KPA Planejamento de Projeto de Software.
SPTO	Do inglês *Software Project Tracking and Oversight*, referente à KPA Acompanhamento de Projeto de Software.
SQA	Do inglês *Software Quality Assurance*, referente à KPA Garantia da Qualidade de Software.
SSM	Do inglês *Software Subcontract Management*, referente à KPA Gerenciamento de Subcontrato de Software.

Tabela 4-1 - Siglas utilizadas para as KPAs do CMM nível 2

Sigla	Descrição
Ab	Do inglês *Ability*, referente à característica comum "Habilidade para Realizar".
Ac	Do inglês *Activity*, referente à característica comum "Atividades Realizadas".
Co	Do inglês *Commitment*, referente à característica comum "Compromisso para Realizar".
Me	Do inglês *Measurement*, referente à característica comum "Medição e Análise".
Ve	Do inglês *Verification*, referente à característica comum "Verificar Implementação".

Tabela 4-2 - Siglas utilizadas para as características comuns do CMM

Para referenciar uma prática específica do modelo, a nomenclatura "X.YZ" é utilizada, onde:

- X corresponde à sigla da KPA que possui a prática
- Y corresponde à sigla da característica comum que agrupa a prática
- Z corresponde ao número da prática

Como exemplo, "SQA.Ac1" está referenciando a prática "atividade 1" da KPA garantia da qualidade de software.

Algumas vezes é preciso citar sub-práticas. Nestes casos, o número da sub-prática é adicionado à referência após ".". Desta forma, "SQA.Ac1.1" está referenciando a sub-prática 1 da prática "atividade 1" da KPA garantia da qualidade de software.

4.2 Mapeamento de XP no CMM

Antes de iniciar o diagnóstico é importante definir como conceitos, termos e papéis citados pelo CMM serão interpretados em XP.

Apesar de ser um modelo para medir a capacidade de desenvolvimento de software das organizações, CMM faz referência a projetos de sistemas, os quais contemplam atividades relacionadas a software, hardware e outros componentes. Devido a isto, cita práticas para separar requisitos de software dos requisitos do sistema, cita o gerente do projeto e o gerente do projeto de software, por exemplo. XP é um processo para desenvolvimento de software e não aborda possíveis interseções com outros tipos de projetos (hardware, por exemplo) associados. Desta forma, o diagnóstico levará em consideração projetos apenas de software, assim o gerente do projeto do CMM será considerado como o próprio gerente de projeto de software.

4.2.1 Mapeamento dos Papéis

A Tabela 4-3 faz um mapeamento entre os papéis citados pelo CMM e os papéis definidos por XP, que serão utilizados por todo o diagnóstico.

Papéis no CMM	Papéis em XP
Gerência Sênior	XP não define papéis para a gerência sênior.
Grupo de engenharia de software	Equipe de XP.
Gerente de Projeto	Mesmo que o Gerente de Projeto de Software.
Gerente de Projeto de Software	Técnico *(Coach)*.
Grupo de Garantia da Qualidade	XP não define papéis para atuar como o grupo de garantia da qualidade.
Grupo de Gerência de Configuração	XP não define papéis para atuar como o grupo de gerência da configuração.

Tabela 4-3 - Mapeamento de papéis CMM e XP

4.2.2 Mapeamento de Termos

A Tabela 4-4 faz um mapeamento entre conceitos citados pelo CMM e conceitos definidos por XP, que serão utilizados por todo o diagnóstico.

Conceitos no CMM	Conceitos em XP
Requisitos alocados ao software	Estórias
Procedimento documentado	A descrição de XP, obtida em livros e artigos.
Plano de desenvolvimento	Planos de XP: *Big Plan*, Plano do *Release*, Plano da Iteração.

Tabela 4-4 - Mapeamento de conceitos CMM e XP

4.3 Diagnóstico da KPA Gerenciamento de Requisitos

O objetivo desta KPA é estabelecer um entendimento comum, um acordo, entre o cliente e o projeto de software a respeito dos requisitos deste cliente [30]. Este cliente pode ser externo ou interno à organização.

Um projeto que envolve desenvolvimento de software poderá muitas vezes conter requisitos de hardware, padrões e outros além dos requisitos de software propriamente ditos. A KPA RM utiliza o termo "requisitos alocados ao software" para se referir especificamente aos requisitos de software de um sistema.

Os requisitos alocados, devidamente acordados entre cliente e projeto, devem servir de base para a criação de planos, documentos técnicos e do andamento do projeto como um todo. Tudo isto deverá ser alterado para refletir mudanças que por ventura sejam realizadas aos requisitos.

4.3.1 Diagnóstico de Satisfação das Práticas-chave

Atividade 1	**O grupo de engenharia de software revisa os requisitos alocados antes de eles serem incorporados ao projeto de software.**

Diagnóstico. O planejamento inicial do projeto - o *Big Plan* – contém as funcionalidades de alto nível do negócio (os requisitos alocados a software) e é desenvolvido pelo cliente e por um integrante do projeto, que faz parte e é representativo do grupo de engenharia de software do projeto. Além disto, durante as reuniões de

planejamento do release e iteração toda a equipe do projeto revisa as estórias (requisitos) visando entendê-las e estimá-las. Desta forma, as estórias são sempre revisadas pela equipe técnica antes de serem incorporados, como requer a prática. Esta prática é considerada satisfeita.

Atividade 2	**O grupo de engenharia de software usa os requisitos alocados como base para planos de software, produtos de trabalho e atividades.**

Diagnóstico. As estórias são base para todas as atividades de XP: *Big Plan*, Plano de *Release*, Plano de Iteração, desenvolvimento, testes. No entanto, as estórias constituem basicamente os requisitos funcionais do software. Para XP o sistema é completamente especificado através de estórias e as define como sendo uma pequena descrição do comportamento do sistema do ponto de vista do usuário [39].

Um problema associado a esta prática é que para o CMM os requisitos alocados incluem critérios de aceitação de requisitos, requisitos técnicos de software (funcionais, de performance, de restrição de linguagem de programação, entre outros) e não técnicos, que afetam e determinam as atividades de software do projeto (produtos a serem entregues, marcos do projeto, datas de entregas, entre outros) **[30]**. XP não enfatiza e não descreve como nem onde documentar os requisitos não técnicos e técnicos não funcionais.

O risco associado a este problema é de os projetos não documentarem e acordarem requisitos críticos para o sucesso do projeto. Esta falta de documentação e acordo pode levar ao não atendimento das necessidades do cliente. Segundo Paulk, "Sempre documente compromissos e os requisitos para o trabalho a ser realizado – estes documentos são cruciais para esclarecimento e resolução de conflitos durante o progresso do projeto" [33].

Um outro problema associado a esta prática é que os requisitos não são gerenciados e controlados conforme sugere a sub-prática RM.Ac2.1. As estórias são documentadas em cartões de papéis e XP não faz menção de controlar a versão de cada estória ou mesmo

do conjunto de estórias que define o *release* e iteração correntes, o que pode levar a futuras inconsistências nestes mesmos planos, produtos de trabalho e atividades.

Esta prática é considerada parcialmente satisfeita, pois XP utiliza as estórias para guiar o andamento do projeto, mas não menciona a documentação e uso dos requisitos não técnicos e o controle das versões dos requisitos documentados.

Atividade 3	**Mudanças aos requisitos alocados são revisadas e incorporadas ao projeto de software.**

Diagnóstico. As mudanças aos requisitos de um projeto XP são identificadas (em geral pelo cliente), avaliadas nas reuniões de planejamento de *release* e/ou iteração, discutidas e entendidas pela equipe de desenvolvimento, documentadas em cartões através de uma nova estória e planejadas para fazerem parte de um *release* e iteração.

Apesar disto, alguns problemas foram identificados quanto à satisfação desta prática. Primeiramente, a sub-prática RM.Ac3.1 descreve que mudanças a compromissos feitos a indivíduos e grupos externos à organização sejam revisadas com a gerência sênior e que mudanças aos compromissos internos à organização sejam negociadas com grupos afetados. XP não menciona o envolvimento da alta gerência em seus projetos e da relação com grupos internos. Em [18], Rasmusson relata a dificuldade em cumprir os prazos estabelecidos em um projeto XP devido à falta de compromisso firmado com grupos internos da organização, como o grupo de administração de dados.

Além disto, da forma como é implementado, XP não possui um registro de histórico das mudanças, conforme requerido pela sub-prática RM.Ac3.2, que menciona a documentação das mudanças. As mudanças geram novas estórias que são incorporadas ao projeto. Não há visibilidade sobre: motivo, premissas utilizadas, responsável pelas mudanças. O risco de não haver registro sobre isto é o de avançar no projeto e não ter dados para responder questões como: "por que a estória X não foi implementada?", "por que o *Big Plan* inicial não foi cumprido?". Em [33], enquanto comenta sobre mudanças evolutivas aos requisitos em um desenvolvimento usando prototipagem, Paulk afirma que "isto é aceitável se o histórico for mantido e os requisitos consolidados".

Esta prática é considerada parcialmente satisfeita, pois, apesar de a equipe revisar as mudanças antes de incorporá-las, esta revisão não inclui a alta gerência, os grupos afetados e não são documentadas de forma a manter um histórico das mudanças.

4.3.2 Sumário do Diagnóstico de Satisfação das Práticas-chave

Prática	Satisfação	Resumo
Atividade 1	Satisfaz	– O grupo de engenharia de software revisa e documenta os requisitos de software em conjunto com o cliente durante o planejamento inicial do projeto (elaboração do *Big Plan*).
Atividade 2	Satisfaz Parcialmente	– XP utiliza as estórias para guiar o andamento do projeto, mas não menciona a documentação e uso dos requisitos não técnicos e o controle das versões dos requisitos documentados.
Atividade 3	Satisfaz Parcialmente	– Apesar de a equipe revisar as mudanças antes de incorporá-las, esta revisão não inclui a alta gerência, os grupos afetados e não são documentadas de forma a manter um histórico das mudanças.

Tabela 4-5 - Satisfação das Práticas de RM por XP

4.4 Diagnóstico da KPA Planejamento de Projetos de Software

O propósito do Planejamento do Projeto de Software é o de estabelecer planos factíveis para realizar a engenharia de software e gerenciar o projeto de software [30]. Este processo aborda aspectos como: estimativa, cronograma, riscos, recursos computacionais críticos, compromissos estabelecidos entre outros.

O planejamento do projeto inicia com um Plano de Trabalho, restrições e metas que definem o projeto de software [30]. O planejamento do projeto deverá abordar o que deve ser feito e de que forma [43]. Se bem documentado, permitirá a comparação com o que foi efetivamente realizado no projeto.

4.4.1 Diagnóstico de Satisfação das Práticas-chave

Atividade 1	**O grupo de engenharia de software participa da equipe de proposta do projeto.**

Diagnóstico. O CMM define através das sub-práticas desta prática chave que o grupo de engenharia de software deve estar envolvido na preparação da proposta e submissão, discussões de esclarecimentos e negociações de mudanças aos compromissos que afetam o projeto de software. Além disto, o grupo revisa os compromissos do projeto (exemplos de compromissos revisados: cronograma, orçamento, recursos, padrões e procedimentos de software). **[30]**.

Esta prática foi considerada satisfeita, pois o contrato (proposta) é elaborado a partir do *Big Plan*, desenvolvido pelo cliente e por um integrante do projeto, que faz parte e é representativo do grupo de engenharia de software do projeto.

Atividade 2	**O planejamento do projeto de software é iniciado nos estágios iniciais e em paralelo com o planejamento do projeto completo.**

Diagnóstico. O planejamento de um projeto XP é iniciado nos estágios iniciais do projeto. Quanto a ser iniciado em paralelo com o planejamento do projeto completo, foi considerado não aplicável. Com esta frase o CMM espera que o planejamento do projeto de software ocorra em paralelo com o planejamento do projeto como um todo, considerando um projeto que possui desenvolvimento de hardware e outros componentes. Como já mencionado, este diagnóstico considera apenas projeto de software.

Atividade 3	**O grupo de engenharia de software participa com outros grupos afetados do planejamento do projeto completo durante a vida do projeto.**

Diagnóstico. Esta atividade do CMM, bem como a Atividade 2, se referem a projetos de sistema (hardware, software e outros componentes) para que o grupo de engenharia de software participe e revise o Plano do Projeto como um todo.

Este item também é considerado não aplicável a este diagnóstico, pois vai além do escopo de projetos de software.

Atividade 4	**Compromissos do projeto de software feitos para indivíduos e grupos externos à organização são revisados com a gerência sênior de acordo com um procedimento documentado.**

Diagnóstico. Não satisfaz. XP não define o relacionamento entre a gerência sênior e o projeto.

Atividade 5	**Um ciclo de vida de software com estágios pré-definidos de tamanho gerenciado é identificado ou definido.**

Diagnóstico. Esta prática é considerada satisfeita, pois as iterações de um projeto XP são vistas como os estágios pré-definidos no projeto e possuem tamanho pequeno visando facilitar o gerenciamento. O propósito destas iterações é justamente promover ciclos menores de desenvolvimento que os ciclos de entrega (*release*) para realizar correções necessárias durante o andamento do projeto **[24]**.

Atividade 6	**O plano de desenvolvimento do projeto de software é desenvolvido de acordo com um procedimento documentado.**

Diagnóstico. Interpretando o plano de desenvolvimento de um projeto XP como os planos *Big Plan*, Plano de Release e Plano de Iteração, XP descreve como documentá-los. Apesar disto, o CMM ressalta aspectos importantes que tipicamente são descritos no procedimento para elaboração do plano de desenvolvimento e que XP não menciona, como:

- Planejamento a partir de padrões estabelecidos (SPP.Ac6.1) – XP não descreve como e nem onde documentar os padrões adotados. O planejamento sobre estes padrões não é descrito pela metodologia. O risco associado é o não seguimento de padrões estabelecidos para o projeto;

- Envolvimento dos grupos afetados (SPP.Ac6.3 e SPP.Ac6.2) – Como já mencionado, XP não cita como deve se dar o envolvimento de grupos externos à equipe do projeto, o que pode acarretar no não cumprimento de compromissos do projeto relacionados a prazo, custo e até mesmo funcionalidades;
- Revisão do plano por parte de grupos afetados e cliente (SPP.Ac6.4) – A revisão do plano por parte de grupos afetados objetiva obter acordo sobre os compromissos firmados. Em XP o acordo é estabelecido nas reuniões de planejamento de *release* e iteração. XP não descreve como obter acordo dos grupos afetados que não participam destas reuniões;
- Gerenciamento e controle dos planos (SPP.Ac6.5) – Objetiva saber a versão do plano de desenvolvimento e as mudanças incorporadas. XP não define como obter estas informações dos planos citados.

Esta prática é considera parcialmente satisfeita, pois a descrição de XP deve ser considerada um procedimento documentado para o planejamento do projeto, no entanto, ela não aborda aspectos importantes do planejamento, como: planejamento a partir de padrões estabelecidos, envolvimento dos grupos afetados, revisão do plano por parte de grupos afetados e cliente, gerenciamento e controle dos planos.

Atividade 7	**O plano para o projeto de software é documentado.**

Diagnóstico. O plano do projeto para o CMM é o mesmo que o plano de desenvolvimento citado nas práticas anteriores **[30]**. O CMM relaciona o que é coberto por este plano nas sub-práticas desta atividade. Exemplos do que deve estar contido no plano para o projeto são: propósito, escopo, metas e objetivos do projeto de software; ciclo de vida do software; procedimentos, métodos e padrões para desenvolvimento e/ou manutenção do software; produtos de trabalho a serem desenvolvidos; estimativas de tamanho dos produtos de trabalho e mudanças aos produtos de trabalho; estimativa de custo e esforço do projeto de software; estimativas de uso dos recursos críticos computacionais; cronograma do projeto de software, incluindo identificação de marcos e revisões; avaliações dos riscos de software do projeto; planos para ferramentas de suporte para a engenharia de software do projeto.

Os planos contemplados por XP (*Big Plan*, Plano do *Release* e Plano da Iteração) não abordam todos os aspectos esperados pelo CMM.

Para realizar o planejamento inicial, o que dará subsídios para determinar o escopo (alto nível), prazo e custo do projeto, informações detalhadas não são conhecidas. A Tabela 4-6 mostra como é organizado o *Big Plan* de um projeto XP **[24]**.

Estória	Estimativa
Nome da estória	Quantidade de meses para implementá-la

Tabela 4-6 - *Big Plan*

Beck sugere que a forma mais apropriada do Plano de *Release* é o conjunto de cartões de estórias que fazem parte dele **[24]**. Um típico possui as seguintes informações: data, tipo da atividade (nova, correção, melhoria ou teste funcional), número da estória, descrição, notas, estimativa técnica, prioridade, entre outras **[25]**. Esta forma de armazenar o Plano de *Release* não deixa claro onde foram documentadas a data do *release* e velocidade da equipe assumida, informações importantes para este plano.

Uma outra forma de registrar o Plano do *Release* é utilizar uma planilha eletrônica simples [24]. Como exemplo, Beck exibe em [24] duas tabelas como as Tabela 4-7 e Tabela 4-8.

Estória	Estimativa de Tempo (Semanas Ideais)	Iteração Atribuída	Release Atribuído
Nome da estória	Quantidade de semanas ideais	Número da iteração atribuída para a estória	Número do *release* atribuído para a estória

Tabela 4-7 - Planejamento de *Release*

Evento	Data
Release X completo	Data em que o *release* X deve ser completado.

Tabela 4-8 - Datas de *Releases*

O planejamento da iteração irá gerar cartões de tarefas com as seguintes informações: data, número da estória, descrição da tarefa, notas, estimativa da tarefa, engenheiro de software, entre outras [25]. O conjunto dos cartões de tarefas de uma iteração constitui o Plano da Iteração. Em [24], um exemplo de Plano de Iteração é exibido, o qual contém as seguintes informações: estória, tarefas da estória, estimativa e responsável por implementar cada estória.

Os três planos definidos por XP não contemplam todas as informações importantes e esperadas pelo CMM como propósito, escopo, metas e objetivos do projeto de software, procedimentos, métodos e padrões para desenvolvimento e/ou manutenção do software, entre outros. Por isto, esta prática é considerada parcialmente satisfeita.

Atividade 8	**Produtos de trabalho de software necessários para estabelecer e manter controle do projeto de software são identificados.**

Diagnóstico. Se o cliente requer a elaboração de algum produto de trabalho (como o manual do usuário, por exemplo), então uma estória para isto deverá ser escrita, associada a um *release*, detalhada para ser realizada em iterações. Desta forma, apesar de XP não relacionar atividades específicas para identificação de produtos de trabalho a serem construídos, isto é feito durante as reuniões de planejamento de release e/ou iterações. Portanto, esta prática é considerada satisfeita.

Atividade 9	**Estimativas para o tamanho dos produtos de trabalho de software (ou mudanças aos tamanhos dos produtos de trabalho de software) são derivadas de acordo com um procedimento documentado.**

Diagnóstico. O CMM exemplifica como tipo de estimativas de tamanho: pontos de função, quantidade de linhas de código, quantidade de requisitos e quantidade de páginas.

Em um projeto XP são estimadas estórias e tarefas. Beck afirma em [24], que a unidade para estimar estórias e tarefas é uma decisão do projeto e não importa qual seja,

o que importa é a equipe acordar uma unidade e todos utilizá-la. Apesar disto, ele sugere que a unidade mais simples para estimar e acompanhar é o "Tempo Ideal". "Tempo Ideal é o tempo sem interrupções onde você pode se concentrar no seu trabalho e você se sente completamente produtivo" [24]. Apesar de usar a palavra "tempo", esta é uma unidade de esforço. "A noção de tempo ideal tem pouco haver com o tempo" ou "nós usamos o termo tempo ideal, mas na realidade é esforço ideal", afirma Beck [24]. Quando um integrante da equipe afirma que a tarefa que ele desprenderá *x* dias ideais em uma tarefa, ele estará considerando a quantidade de dias de trabalho sem interrupções, nem mesmo para ajudar um outro integrante atuando como seu par de programação, o que acontece freqüentemente.

Este diagnóstico considera que XP não define como estimar tamanho para os principais produtos de trabalho e atividades, mas apenas define como estimar esforço para todas as atividades (estórias e tarefas). Desta forma, esta prática é considerada não satisfeita.

Em relação a possibilidade de se estimar tamanho em um projeto XP, Beck [24], considera afirma que é ineficaz o processo de estimar quantidade de linhas de código com o objetivo de se prever o esforço necessário.

Atividade 10	**Estimativas para esforço e custo do projeto de software são derivadas de acordo com procedimento documentado.**

Diagnóstico. Como já mencionado, XP determina como estimar esforço. Além disto, XP atende várias das sub-práticas relacionadas pelo CMM, como:

- "Dados de produtividade são usados para as estimativas quando disponíveis" – a velocidade de uma iteração definida por XP é a produtividade da equipe, que deve sempre ser atualizada [24];
- "Esforço, alocação de pessoal, e estimativas de custo são baseadas em experiências passadas" – XP sempre reforça a necessidade de usar o "*Yesterday Weather*", que significa olhar dados passados para realizar as estimativas [24];
- "Estimativas e premissas feitas para derivar as estimativas são documentadas, revisadas e acordadas" – Em XP, as estimativas e premissas são documentadas

nos cartões de papel e revisadas por todos da equipe, já que são realizadas em conjunto [24].

Quanto às estimativas de custo, XP considera que devem ser derivadas da quantidade de pessoas e dedicação de esforço ao projeto [24]. Apesar de, em geral, esforço necessário ser o principal custo de um projeto de software, outros custos em geral são envolvidos, como viagens, estrutura física e treinamentos. Desta forma, XP não define como estimar outros custos relacionados ao projeto, nem mesmo menciona que deve ser feito.

Esta prática é considerada parcialmente satisfeita, pois apesar de estimar esforço, não atende à sub-prática de relacioná-las às estimativas de esforço e custo. Além disto, não define de forma satisfatória como estimar custos.

Atividade 11	**Estimativas para os recursos computacionais críticos do projeto são derivadas de acordo com um procedimento documentado.**

Diagnóstico. CMM exemplifica os recursos computacionais críticos como recursos no ambiente de desenvolvimento, no ambiente de teste e integração, no ambiente de implantação ou em qualquer combinação destes **[30]**.

XP em nenhum momento aborda a questão de planejar e estimar os recursos computacionais críticos, portanto, esta prática é considerada não satisfeita.

Atividade 12	**O cronograma de software do projeto é derivado de acordo com um procedimento documentado.**

Diagnóstico. Para Sommerville, "realizar um cronograma envolve separar o trabalho total em atividades e julgar o tempo requerido para completá-las" **[14]**. Além disto, coordenar as atividades que ocorrem em paralelo evitando que o projeto seja atrasado porque uma atividade crítica não foi finalizada ou foi atrasada. Em geral, cronogramas são representados por gráficos que mostram dependências entre atividades, alocação de recursos, caminho crítico de atividades, como o gráfico PERT **[14]**.

XP não possui um cronograma convencional para o controle das atividades. Para cada *release* XP define datas de início e fim e estórias, para cada iteração define datas de início e fim e tarefas. As estórias não possuem responsáveis, as tarefas sim. Dentro dos *releases* e iterações não há datas intermediárias para encerrar estórias e tarefas respectivamente, isto deve ocorrer até o final do *release* e iteração. Não há controle das dependências entre as estórias e tarefas. Um conselho de Beck é "ignore dependência entre partes – planeje como se as partes do desenvolvimento pudessem ser trocadas entre si" [25]. Durante a reunião de planejamento do *release* estas partes são as estórias e na de planejamento da iteração, as tarefas [25].

Apesar de não ter o controle das dependências das tarefas e trabalho paralelo, o cronograma de XP é considerado por este diagnóstico satisfatório para a forma como os projetos são conduzidos. Segundo Sommerville, "não é útil subdividir atividades em unidades menores que uma semana ou duas para executar" [14]. Em XP as iterações são tipicamente de duas a três semanas de duração e tarefas possuem duração menor do que isto.

Apesar de XP não citar tudo o que, do ponto de vista do CMM, tipicamente um procedimento documentado para elaboração de cronograma deve conter, esta prática é considerada satisfeita, pois XP define de forma objetiva todos os aspectos importantes de como elaborar o cronograma em um projeto XP.

Atividade 13	**Os riscos de software associados com custo, recursos, cronograma e aspectos técnicos do projeto são identificados, avaliados e documentados.**

Diagnóstico. Esta prática é considerada não satisfeita, pois XP não aborda a questão de riscos do projeto de software.

Atividade 14	**Planos para as instalações e ferramentas de apoio para a engenharia de software do projeto são preparados.**

Diagnóstico. O CMM cita exemplos destas instalações e ferramentas de suporte: computadores de desenvolvimento de software, periféricos, computadores de teste de software e software de suporte **[30]**.

Os planos de XP não abordam estes aspectos. Em relação às instalações necessárias, XP aborda apenas o local de desenvolvimento e cita algumas características para que ele seja efetivo:

- O ambiente deve ser montado de forma que as pessoas possam trabalhar perto uma das outras para que vejam umas às outras e escutarem questões levantadas entre si;
- O ambiente deve prover algum lugar com a privacidade necessária para realização de telefonemas e reuniões.

Desta forma, esta prática é considerada não satisfeita, já que não há planos explícitos em XP para as instalações e ferramentas de apoio necessárias.

Atividade 15	**Dados do planejamento de software são registrados.**

Diagnóstico. Através das sub-práticas, o CMM aborda o que espera do registro do planejamento:

- SPP.Ac15.1: "Informação registrada inclui as estimativas e informações associadas necessárias para reconstruir as estimativas e avaliar a sua sensatez";
- SPP.Ac15.2: "Os dados de planejamento de software são gerenciados e controlados".

Como já mencionado, XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados ao planejamento. Por isto, esta prática é considerada parcialmente satisfeita. A falta de controle de versão nos planos de *release* e iteração pode levar à falta de histórico sobre as mudanças realizadas no planejamento, o que pode ser bastante prejudicial num projeto XP, onde mudanças podem ser realizadas a qualquer momento do projeto, de forma flexível. Manter o histórico destas mudanças contribui para o entendimento do progresso do projeto e renegociação de compromissos.

4.4.2 Sumário do Diagnóstico de Satisfação das Práticas-chave

Prática	Diagnóstico	Resumo
Atividade 1	Satisfaz	– O contrato é elaborado com a participação de um integrante do projeto, representante do grupo de engenharia de software.
Atividade 2	Não Aplicável	– XP foca em projetos de software. Esta prática é aplicável para projetos que envolvem outros componentes além de software.
Atividade 3	Não Aplicável	– XP foca em projetos de software. Esta prática é aplicável para projetos que envolvem outros componentes além de software.
Atividade 4	Não Satisfaz	– XP não define o relacionamento entre a gerência sênior e o projeto.
Atividade 5	Satisfaz	– As iterações de um projeto XP são vistas como estágios predefinidos de tamanho gerenciado.
Atividade 6	Satisfaz Parcialmente	– A descrição de XP deve ser considerada um procedimento documentado para o planejamento do projeto, no entanto, ela não aborda aspectos importantes do planejamento, como: planejamento a partir de padrões estabelecidos, envolvimento dos grupos afetados, revisão do plano por parte de grupos afetados e cliente, gerenciamento e controle dos planos.
Atividade 7	Satisfaz Parcialmente	– Os três planos definidos por XP não contemplam todas as informações importantes e esperadas pelo CMM como propósito, escopo, metas e objetivos do projeto de software, procedimentos, métodos e padrões para desenvolvimento e/ou manutenção do software, entre outros.

Atividade 8	Satisfaz	– Os produtos de trabalho são identificados durante o Planejamento do Release e/ou da Iteração.
Atividade 9	Não Satisfaz	– XP não contempla estimativa de tamanho dos produtos de trabalho.
Atividade 10	Satisfaz Parcialmente	– XP define como estimar esforço, mas não relaciona estas estimativas com as estimativas de tamanho. Além disto, não define de forma satisfatória como estimar custos.
Atividade 11	Não Satisfaz	– XP não aborda a questão de planejar e estimar os recursos computacionais críticos.
Atividade 12	Satisfaz	– XP define de forma objetiva todos os aspectos importantes de como elaborar o cronograma em um projeto.
Atividade 13	Não Satisfaz	– XP não aborda a questão de riscos do projeto de software.
Atividade 14	Não Satisfaz	– XP não contempla planos explícitos para as instalações e ferramentas de apoio do projeto de software.
Atividade 15	Satisfaz Parcialmente	– XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados ao planejamento.

Tabela 4-9 - Satisfação das Práticas de SPP por XP

4.5 Diagnóstico da KPA Acompanhamento de Projetos de Software

"O propósito do Acompanhamento de Projeto de Software é prover visibilidade adequada no progresso real de forma que o gerenciamento possa tomar ações efetivas

quando a performance do projeto de software desvia significativamente dos planos de software" [30]. O Plano do Projeto é utilizado como base para o acompanhamento do projeto. Os resultados obtidos são comparados com o planejado para o projeto.

4.5.1 Diagnóstico de Satisfação das Práticas-chave

Atividade 1	**Um plano de desenvolvimento de software documentado é usado para acompanhamento das atividades de software e comunicação do *status*.**

Diagnóstico. Esta prática é considerada satisfeita, pois os planos de *Release* e iteração guiam as atividades da equipe e estão sempre disponíveis para todos os envolvidos no projeto.

Atividade 2	**O plano de desenvolvimento de software do projeto é atualizado de acordo com um procedimento documentado.**

Diagnóstico. Num projeto XP, os planos mudam **[24]**:

- Se a velocidade da equipe assumida para a iteração e/ou *release* diferir do planejado. Neste caso, se a velocidade da equipe é menor, então a mesma deverá requisitar ao cliente que selecione estórias para serem retiradas da iteração e/ou *release*;
- Se a velocidade da equipe é maior, então a mesma deverá requisitar ao cliente que selecione estórias para serem inseridas na iteração e/ou *release*;
- A qualquer momento, se o cliente desejar inserir novas estórias ao *release* ou retirar outras.

"Planejamento de *release* acontece todo o tempo. Todas as vezes que o cliente muda de idéia sobre os requisitos ou de prioridades, o plano muda" [24]. Para isto, uma reunião de planejamento de *release* é realizada para que o cliente selecione uma estória de mesma ou mais baixa complexidade para ser retirada do *release*. Feito isto, o cliente irá priorizá-la para ser implementada em alguma iteração.

Em ambos os casos, os planos de *release* e iteração são alterados. "Os planos são apenas um atalho para a visão corrente das coisas que serão feitas. Ele será alterado todo

o tempo. Todos – desenvolvedores, clientes, e gerente – precisam aceitar mudanças constantes" [24]. Desta forma, esta prática é considerada satisfeita.

Atividade 3	**Compromissos e mudanças aos compromissos do projeto de software feitos a indivíduos e grupos externos à organização são revisados com a gerência sênior de acordo com um procedimento documentado.**

Diagnóstico. Não satisfaz. XP não define o relacionamento entre a gerência sênior e o projeto.

Atividade 4	**Mudanças aprovadas aos compromissos que afetam o projeto de software são comunicadas aos membros do grupo de engenharia de software e outros grupos relacionados a software.**

Diagnóstico. Num projeto XP, os compromissos que afetam o projeto de software são relacionados com datas de *release* e fim de iteração, velocidade da equipe assumida, estimativas, estórias do *release* e iteração, entre outros. Todos estes compromissos são acordados em reuniões onde participam clientes e equipe, portanto, esta prática é considerada satisfeita.

Atividade 5	**Os tamanhos dos produtos de trabalho de software (ou tamanhos das mudanças aos produtos de trabalho de software) são acompanhadas, e ações corretivas são tomadas quando necessário.**

Diagnóstico. Esta prática é considerada não satisfeita, pois XP não aborda estimativa de tamanho de produto de trabalho, tão pouco seu acompanhamento.

Atividade 6	**Esforço e custo de software do projeto são acompanhados, e ações corretivas são tomadas quando**

	necessário.

Diagnóstico. O acompanhamento do esforço é realizado pelo *tracker*, que deve questionar pessoalmente cada programador quanto já foi realizado de cada tarefa atribuída e quanto ainda resta **[24]**. A periodicidade sugerida para este acompanhamento é de duas vezes por semana **[24]**.

A partir deste acompanhamento, XP sugere ações a serem tomadas caso desvios sejam identificados [24]:

- Se identificado que algum programador não poderá cumprir uma tarefa a ele atribuída, então deve-se procurar outro que possua tempo disponível para implementá-la. Caso não encontre, então deve-se levar o problema para que o cliente selecione o que poderá sair da iteração;
- Se identificado que há tempo sobrando na iteração, então o cliente deverá ser informado para incluir uma outra estória.

Apesar de acompanhar esforço e promover a tomada de ação corretiva, XP não menciona o acompanhamento de custos. Assim, esta prática é considerada parcialmente satisfeita.

Atividade 7	**Recursos computacionais críticos do projeto são acompanhados e ações corretivas são tomadas quando necessário.**

Diagnóstico. Esta prática não é satisfeita, pois XP não aborda de forma explícita o planejamento e acompanhamento de recursos críticos computacionais.

Atividade 8	**O cronograma de software do projeto é acompanhado, e ações corretivas são tomadas quando necessário.**

Diagnóstico. Como já citado anteriormente, XP não possui um cronograma convencional para o controle das atividades: datas e atividades são definidas para cada *release* e iteração do projeto.

A realização das atividades é verificada pelo *tracker* durante e no início das iterações. Num projeto XP, uma estória é considerada finalizada quando "é demonstrada para o cliente e ele concorda que ela atende às expectativas" [24]. Para isto, os testes funcionais devem ser executados e os problemas encontrados serão julgados pelo cliente se são grandes o suficiente para impactar a aceitação (finalização) da estória [24].

O acompanhamento das datas das iterações é muito simples em um projeto XP: "a iteração é finalizada na data acordada no início. Uma iteração de duas semanas encerra-se em duas semanas, sem mais" [24]. Se uma estória planejada para uma iteração não foi finalizada a tempo, ela deverá ser replanejada para a próxima iteração, com o conhecimento do cliente, nas reuniões de início da iteração.

Esta prática é considerada então satisfeita, pois em um projeto XP o cronograma é acompanhado e ações corretivas são tomadas quando necessário.

Atividade 9	**Atividades técnicas de engenharia de software são acompanhadas, e ações corretivas são tomadas quando necessário.**

Diagnóstico. As atividades do grupo de engenharia são acompanhadas diariamente através das reuniões *Stand Up Meetings*, que tem como objetivo comunicar os problemas, o que cada um está e o que não está fazendo **[24]**. Os problemas identificados são, então, encaminhados para serem resolvidos. Portanto, esta prática é considerada satisfeita.

Atividade 10	**Os riscos de software associados a custo, recursos, cronograma e aspectos técnicos do projeto são acompanhados.**

Diagnóstico. Esta prática é considerada não satisfeita, pois XP não aborda a identificação e acompanhamento dos riscos do projeto.

Atividade 11	**Dados de medida e dados de replanejamento reais para o projeto de software são registrados.**

Diagnóstico. Assim como a atividade SPP.Ac15, esta atividade é considerada parcialmente satisfeita, pois XP não menciona o controle das versões dos produtos de trabalho associados ao planejamento, conforme sugerido pelas sub-práticas desta prática (consultar SPP.Ac15 para maiores detalhes).

Atividade 12	**O grupo de engenharia de software conduz revisões internas periódicas para acompanhar progresso técnico, planos, performance, e outros itens contra o plano de desenvolvimento de software.**

Diagnóstico. Esta prática é satisfeita através das reuniões de planejamento de iteração. Estas reuniões são periódicas e, para se planejar a próxima iteração, o progresso técnico, planos e performance são utilizados.

Atividade 13	**Revisões formais para endereçar realizações e resultados do projeto de software são conduzidas nos marcos selecionados do projeto de acordo com procedimento documentado.**

Diagnóstico. As reuniões de planejamento de *release* podem ser mapeadas para estas revisões formais. Estas reuniões satisfazem algumas das sub-práticas do CMM, como:

- SPTO.Ac13.1: "são planejadas para ocorrer em pontos significativos do cronograma do projeto de software, como o começo ou finalização de estágios selecionados";
- SPTO.Ac13.2: "São conduzidas com o cliente, usuário final, e grupos afetados da organização, conforme apropriado".

No entanto, esta prática não é considerada completamente satisfeita, pois XP não menciona a sub-prática citadas pelo CMM:

- SPTO.Ac13.6: "Relaciona os riscos do projeto de software".

4.5.2 Sumário do Diagnóstico de Satisfação das Práticas-chave

Prática	Diagnóstico	Resumo
Atividade 1	Satisfaz	– Os planos de Release e iteração guiam as atividades da equipe e estão sempre disponíveis para todos os envolvidos no projeto.
Atividade 2	Satisfaz	– Os planos de release e iteração são atualizados para refletir o andamento do projeto.
Atividade 3	Não Satisfaz	– XP não define o relacionamento entre a gerência sênior e o projeto.
Atividade 4	Satisfaz	– Todos os compromissos de um projeto XP são acordados em reuniões onde participam clientes e equipe.
Atividade 5	Não Satisfaz	– XP não aborda estimativa de tamanho de produto de trabalho, tão pouco seu acompanhamento.
Atividade 6	Satisfaz Parcialmente	– Apesar de acompanhar esforço e promover a tomada de ação corretiva, XP não menciona o acompanhamento de custos.
Atividade 7	Não Satisfaz	– XP não aborda de forma explícita o planejamento e acompanhamento de recursos críticos computacionais.
Atividade 8	Satisfaz	– Em um projeto XP o cronograma é acompanhado e ações corretivas são tomadas quando necessário.
Atividade 9	Satisfaz	– As atividades do grupo de engenharia são acompanhadas diariamente através das reuniões *Stand Up Meetings*.
Atividade 10	Não Satisfaz	– XP não aborda a identificação e acompanhamento dos riscos do projeto.

Atividade 11	Satisfaz Parcialmente	– XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados a replanejamento.
Atividade 12	Satisfaz	– As reuniões de planejamento de iteração ocorrem periodicamente, onde informações sobre progresso técnico, planos e performance são utilizados.
Atividade 13	Satisfaz Parcialmente	– As reuniões de planejamento de *release* ocorrem em pontos significativos do projeto, contam com a participação de cliente e equipe, relaciona planos, mas não aborda riscos do projeto.

Tabela 4-10 - Satisfação das Práticas de SPTO por XP

4.6 Diagnóstico da KPA Garantia da Qualidade de Software

O objetivo da garantia da qualidade é prover visibilidade sobre o processo utilizado e o projeto construído. Para isto, esta KPA menciona um grupo de garantia da qualidade, o qual deve possuir um canal direto de comunicação com a gerência sênior independente da equipe do projeto. O grupo de garantia da qualidade cumpre seu objetivo através de auditorias e revisões do processo e dos produtos. Os problemas identificados devem ser resolvidos pela equipe do projeto e documentados e acompanhados até fechamento pelo grupo de garantia da qualidade. Problemas não resolvidos pela equipe devem ser escalados para a alta gerência. [30].

Além das revisões e auditorias, o grupo de garantia da qualidade deve reportar suas atividades para a equipe, grupo de garantia da qualidade do cliente e alta gerência. As atividades de garantia da qualidade devem ser planejadas no início do projeto [30].

4.6.1 Diagnóstico de Satisfação das Práticas-chave

XP não define um grupo de garantia da qualidade, tão pouco atividades para revisão e/ou auditoria de processos.

Paulk vê a programação em pares como uma atividade de garantia da qualidade, pois ela contribui para a aderência a padrões adotados pelo projeto [31]. Apesar disto, Paulk também alega que esta atividade não provê visibilidade para a alta gerência e pode não ser eficaz para projetos grandes e para situações de grande pressão externa [31].

O diagnóstico considera que nenhuma das atividades descritas pelo CMM para esta KPA é atendida, pois não há um pré-requisito necessário para as atividades: um grupo de garantia da qualidade, independente do projeto. Por isto, o diagnóstico desta KPA não foi detalhado como nas KPAs RM, SPP e SPTO. As atividades desta KPA são [30]:

- SQA.Ac1: **Um plano de SQA é preparado para o projeto de software de acordo com um procedimento documentado;**
- SQA.Ac2: **As atividades do grupo de SQA são realizadas de acordo com o plano de SQA;**
- SQA.Ac3: **O grupo de SQA participa na preparação e revisão do plano de desenvolvimento de software, padrões e procedimentos;**
- SQA.Ac4: **O grupo de SQA revisa as atividades da engenharia de software para verificar conformidade;**
- SQA.Ac5: **O grupo de SQA audita os produtos de trabalho designados para verificar conformidade;**
- SQA.Ac6: **O grupo de SQA periodicamente reporta os resultados de suas atividades ao grupo de engenharia de software;**
- SQA.Ac7: **Desvios identificados nas atividades e produtos de trabalho de software são documentados e tratados de acordo com procedimento documentado;**
- SQA.Ac8: **O grupo de SQA conduz revisões periódicas das suas atividades e problemas encontrados com o pessoal de SQA do cliente, como apropriado.**

4.7 Diagnóstico da KPA Gerência de Configuração de Software

A gerência da configuração de software tem como objetivo estabelecer e manter a integridade dos produtos de software do projeto. Uma das práticas da característica comum "Habilidade para Realizar" desta KPA, descreve que "um grupo responsável por coordenar e implementar SCM para o projeto existe".

Dentre as principais atividades de gerência de configuração estão: planejar como a gerência de configuração será tratada no projeto, identificar a configuração de software, identificar itens de configuração, estabelecer baselines, controlar mudanças, registrar o status da configuração e realizar auditorias de configuração [3].

4.7.1 Diagnóstico de Satisfação das Práticas-chave

Atividade 1	**Um plano de SCM é preparado para cada projeto de software de acordo com um procedimento documentado.**

Diagnóstico. Esta prática é considerada não satisfeita, pois XP não menciona plano para o gerenciamento de configuração.

Atividade 2	**Um plano de SCM documentado e aprovado é utilizado como base para a realização das atividades de SCM.**

Diagnóstico. Esta prática é considerada não satisfeita, pois, como citado no diagnóstico da prática SCM.Ac1, XP não menciona plano para o gerenciamento de configuração.

Atividade 3	**Um sistema de biblioteca de gerenciamento de configuração é estabelecido como um repositório para as baselines de software.**

Diagnóstico. Não satisfeita. XP não menciona o estabelecimento de um sistema de gerenciamento da configuração. Sub-práticas e exemplos citados pelo CMM para este sistema inclui: possuir níveis de controle, prover armazenamento e recuperar itens de configuração, apoiar a criação correta de produtos a partir da baseline de software.

Atividade 4	**Os produtos de trabalho de software a serem colocados sob gerência de configuração são identificados.**

Diagnóstico. O principal produto de trabalho em um projeto XP é o código. Outros produtos de trabalho que precisem ser desenvolvidos no projeto são identificados nas reuniões de planejamento de release e/ou iterações. Esta prática é considerada parcialmente satisfeita, pois se o projeto precisar possuir outros itens de configuração além de código, o que é bastante comum, várias das sub-práticas não são atendidas, como:

- SCM.Ac4.1: "Os itens de configuração são selecionados com base em um critério documentado". XP não define critérios para selecionar itens de configuração;
- SCM.Ac4.4: "As baselines a que cada item/unidade de configuração pertence são especificadas". XP não determina *baselines* para itens de configuração além de código;
- SCM.Ac4.5: "O ponto no desenvolvimento em que cada item/unidade de configuração é colocado sob gerência de configuração é especificado". XP não menciona o momento para iniciar a gerência de configuração para itens que não sejam código.

Atividade 5	**Requisições de mudança e reportagens de problema para todos os itens de configuração são iniciados, registrados, revisados, aprovados e acompanhados de acordo com um procedimento documentado.**

Diagnóstico. Para o código, depois de seu *release*, problemas ou melhorias devem ser identificadas, documentadas e tratadas como uma estória **[24]**. Uma estória é registrada em cartões de papel, revisada e aprovada pela equipe, quando é também estimada e é acompanhada durante a iteração, conforme sugere a prática. Desta forma, esta prática é considerada satisfeita.

Atividade 6	**Mudanças às *baselines* são controladas de acordo com um procedimento documentado.**

Diagnóstico. Considerando o *build* integrado como a *baseline* de software, esta prática é considerada satisfeita. Em sub-práticas, o CMM relaciona atividades típicas realizadas para o controle das *baselines*:

- SCM.Ac6.1: "Revisões e/ou testes de regressão são realizados para assegurar que mudanças não causaram efeitos indesejados à *baseline*";
- SCM.Ac6.3: "Itens/unidades de configuração são colocadas e retiradas de maneira a manter a corretude e integridade da *baseline*".

A prática de XP integração contínua descreve que "o código é integrado e testado após poucas horas – um dia de desenvolvimento no máximo" **[25]**. Para mudar um código na *baseline*, um par de desenvolvedores deve integrar seu código com as mudanças efetuadas ao código da *baseline* e rodar os testes até que 100% deles passem **[25]**. XP também sugere que o projeto possua uma máquina dedicada à realizar esta integração **[25]**.

Atividade 7	**Produtos da biblioteca de *baseline* de software são criados e seus releases são controlados de acordo com procedimento documentado.**

Diagnóstico. XP determina que o produto (software) é criado a partir da *baseline* de software (*build* integrado, com 100% dos testes rodando), conforme o planejamento de *releases*. Esta prática é considerada satisfeita.

Atividade 8	**O status dos itens/unidades de configuração é registrado de acordo com um procedimento documentado.**

Diagnóstico. Não satisfaz: XP não menciona o registro do status dos itens de configuração. Segundo a sub-prática SCM.Ac8.1, este registro deve ser feito em detalhe suficiente para que o conteúdo e status de cada item/unidade de configuração sejam conhecidos e versões anteriores possam ser recuperadas **[30]**.

Atividade 9	**Relatórios padrões documentando as atividades e SCM e conteúdo das baselines são desenvolvidos e tornados disponíveis para os grupos e indivíduos afetados.**

Diagnóstico. Não satisfaz, pois XP não menciona o uso de relatórios de gerência de configuração e disponibilização aos grupos afetados.

Atividade 10	**Auditorias nas baselines de software são conduzidas de acordo com um procedimento documentado.**

Diagnóstico. XP não menciona auditorias nas *baselines* de software, portanto esta prática não é considerada satisfeita.

4.7.2 Sumário do Diagnóstico de Satisfação das Práticas-chave

Prática	Diagnóstico	Resumo
Atividade 1	Não Satisfaz	– XP não menciona plano para o gerenciamento de configuração.
Atividade 2	Não Satisfaz	– XP não menciona plano para o gerenciamento de configuração.
Atividade 3	Não Satisfaz	– XP não menciona o estabelecimento de um sistema de gerenciamento da configuração.
Atividade 4	Satisfaz Parcialmente	– Se o projeto possuir outros itens de configuração além de código, várias das sub-práticas não são atendidas.
Atividade 5	Satisfaz	– Problemas ou melhorias devem ser identificados, documentados e tratados como uma estória, que são registradas em cartões de papel, revisadas e aprovadas pela equipe e acompanhadas durante a iteração.
Atividade 6	Satisfaz	– Para mudar um código na *baseline*, um par de desenvolvedores deve integrar seu código com

		as mudanças efetuadas ao código da *baseline* e rodar os testes até que 100% deles passem.
Atividade 7	Satisfaz	– XP determina que o produto (software) é criado a partir da *baseline* de software (*build* integrado, com 100% dos testes rodando), conforme o planejamento de *releases*.
Atividade 8	Não Satisfaz	– XP não menciona o registro do *status* dos itens de configuração.
Atividade 9	Não Satisfaz	– XP não menciona o uso de relatórios de gerência de configuração e disponibilização aos grupos afetados.
Atividade 10	Não Satisfaz	– XP não menciona auditorias nas *baselines* de software.

Tabela 4-11 - Satisfação das Práticas de SCM por XP

4.8 Resumo

O objetivo deste diagnóstico foi identificar os problemas de não aderência de XP às práticas da característica comum "Atividades para Realizar" das KPAs do nível 2 do CMM. A KPA Gerenciamento do Subcontrato de Software não foi analisada, pois foge ao escopo da metodologia de desenvolvimento de software XP.

XP implementa grande parte das práticas das KPAs Gerenciamento de Requisitos, Planejamento de Projetos, Acompanhamento de Projetos e Gerenciamento de Configuração, no entanto foi identificado que algumas práticas não são implementadas e outras não são completamente implementadas.

5 Guia XP-CMM2

Este Capítulo irá propor uma solução, ou seja, uma forma de atender o CMM, para cada prática classificada como "não satisfeita" ou "parcialmente satisfeita" pelo Diagnóstico XP-CMM2. A organização do Capítulo é descrita abaixo:

- Seção 5.1 – A Abordagem Utilizada: descreve a abordagem utilizada para a seleção das soluções propostas;
- Seção 5.2 – Solução para a KPA Gerenciamento de Requisitos: descreve as soluções selecionadas para as práticas classificadas como "não satisfeita" ou "parcialmente satisfeita" pelo Diagnóstico XP-CMM2 para a KPA Gerenciamento de Requisitos;
- Seção 5.3 – Solução para a KPA Planejamento de Projetos de Software: descreve as soluções selecionadas para as práticas classificadas como "não satisfeita" ou "parcialmente satisfeita" pelo Diagnóstico XP-CMM2 para a KPA Planejamento de Projetos de Software;
- Seção 5.4 – Solução para a KPA Acompanhamento de Projetos de Software: descreve as soluções selecionadas para as práticas classificadas como "não satisfeita" ou "parcialmente satisfeita" pelo Diagnóstico XP-CMM2 para a KPA Acompanhamento de Projetos de Software;
- Seção 5.5 – Solução para a KPA Garantia da Qualidade de Software: descreve a solução selecionada para a KPA Garantia da Qualidade de Software;
- Seção 5.6 – Solução para a KPA Gerenciamento de Configuração de Software: descreve as soluções selecionadas para as práticas classificadas como "não satisfeita" ou "parcialmente satisfeita" pelo Diagnóstico XP-CMM2 para a KPA Gerenciamento de Configuração de Software;

- Seção 5.7 – Visão Geral da Solução: apresenta um breve resumo da solução por KPA;
- Seção 5.8 – Análise da Solução: analisa o impacto da solução proposta na metodologia XP.

5.1 A Abordagem Utilizada

Para cada problema identificado no Diagnóstico XP-CMM2, será proposta aqui uma solução. As soluções propostas utilizaram como base os seguintes fundamentos:

- Atender às práticas – Todas as práticas do nível 2 de maturidade do CMM deverão ser satisfeitas. Vale ressaltar mais uma vez que é possível alcançar um nível de maturidade sem implementar rigorosamente todas as práticas relacionadas (para mais detalhes, consultar o Capítulo 2). Apesar disto, o fato de atender a todas as práticas de um nível, traz mais segurança em atingir o nível desejado e diminui a quantidade de julgamento em uma avaliação oficial;
- Estar alinhada à cultura XP – As soluções deverão estar alinhadas à filosofia XP, de forma a não descaracterizá-la e a realizar o menor impacto possível na metodologia. Aspectos serão inseridos em XP para contemplar todas as práticas do nível 2 do CMM, no entanto, isto deverá ser feito de forma que a metodologia continue sendo XP. Os valores e princípios de XP serão levados como base para apontar uma solução para os problemas;
- Ser genérica – Este trabalho não tem como objetivo definir uma nova metodologia, mas propor uma Guia de Uso. Assim, para utilizar XP numa organização nível 2 do CMM, o interessado poderá utilizar o Guia XP-CMM2 para identificar o que é necessário complementar em XP. Este trabalho propõe soluções visando facilitar o uso de XP por organizações nível 2, mas estas soluções são genéricas para que sejam adotadas de acordo com a cultura e estilo das organizações. Templates, ferramentas e métodos poderão ser sugeridos no intuito de contribuir e facilitar a utilização do guia;

- Utilizar as boas práticas da engenharia de software amplamente difundidas [40], [14].

Este trabalho irá propor uma solução para cada problema, mas várias outras alternativas poderiam ser adotadas. Assim como o CMM e XP requerem bom senso ao serem aplicados, se uma organização desejar adotar a solução proposta por este trabalho, deverá avaliar a aderência à sua cultura e necessidades.

Além de apontar soluções para os problemas identificados no diagnóstico, este trabalho inclui aspectos visando enriquecer e facilitar o uso do Guia, como *templates* e ferramentas. Apesar disto, este trabalho não inclui os procedimentos documentados requeridos pelo CMM nível 2 para algumas práticas. O motivo disto é que um procedimento documentado deve ser elaborado de acordo com cultura e diretrizes das organizações. Estas deverão adaptar as diretrizes de XP e sugestões do Guia XP-CMM2 visando atender suas necessidades.

Algumas sugestões poderão impactar o Diagnóstico XP-CMM2 realizado. Assim, este trabalho poderá se tornar um pouco mais amplo que sugerir soluções apenas para os problemas detectados pelo diagnóstico.

5.2 Solução para a KPA Gerenciamento de Requisitos

Para cada problema identificado pelo Diagnóstico XP-CMM2 em relação à KPA Gerenciamento de Requisitos uma solução será proposta.

5.2.1 RM.Ac2

Prática	**Descrição**
Atividade 2	**O grupo de engenharia de software usa os requisitos alocados como base para planos de software, produtos de trabalho e atividades**
Diagnóstico	**Problema**
Satisfaz Parcialmente	XP utiliza as estórias para guiar o andamento do projeto, mas não menciona a documentação e uso dos requisitos não técnicos e o controle das versões dos requisitos documentados.

Solução. Para atender completamente esta prática, é necessário:

- Documentar todos os requisitos do projeto, incluindo requisitos não técnicos, técnicos funcionais e não funcionais e critérios de aceitação;
- Controlar a versão dos requisitos.

O Plano do Projeto deverá conter uma breve descrição do escopo do projeto, incluindo requisitos técnicos, não técnicos e critérios de aceitação. Este Plano deverá ser criado especialmente para atender a prática SPP.AC7. Este trabalho sugere um template para Plano do Projeto (**Erro! Fonte de referência não encontrada.**).

Os requisitos serão detalhados em estórias, conforme sugere XP, em um documento chamado Documento de Requisitos. Este Documento de Requisitos conterá as estórias agrupadas por *release* (o Plano de *Release* de XP). Para os *releases* ainda não planejados, o detalhamento das estórias deverá estar alinhado ao detalhamento do *Big Plan* sugerido por XP, ou seja, os requisitos estarão documentados num nível macro, sem muitos detalhes. As mudanças aos requisitos serão gerenciadas neste nível. Segundo Beck et al, a forma mais simples de registrar um plano de *release* em um computador é em uma planilha simples [24]. Este trabalho sugere um Template do Documento de Requisitos (Apêndice C).

O Plano do Projeto deverá apontar para o Documento de Requisitos. Os motivos para separar os documentos Plano do Projeto e Documento de Requisitos são:

- O Plano do Projeto possui informações gerais pertinentes a todo o projeto, enquanto o Documento de Requisitos foca nos requisitos do produto;
- O Documento de Requisitos será utilizado primordialmente para o planejamento e acompanhamento dos *releases* e iterações, enquanto o Plano do Projeto como documento base para acompanhamento do projeto como um todo;
- O Plano do Projeto e o Documento de Requisitos poderão possuir distintos controle das mudanças;
- O Documento de Requisitos poderá ser extenso, de acordo com o produto a ser desenvolvido. Um documento a parte torna o acesso à descrição do produto mais usual pela equipe.

O CMM sugere que os requisitos sejam gerenciados e controlados, ou seja, que a versão dos requisitos em um dado momento é conhecida (controle de versão), e que mudanças são incorporadas de uma maneira controlada (controle de mudanças). Isto é mais simples do que colocar sob gerência de configuração, pois a gerência de configuração envolve, entre outras coisas, entrada em *baselines* e controle formal das mudanças.

Para controlar a versão dos requisitos, este trabalho sugere diferentes abordagens para os documentos citados:

- Plano do Projeto: fazer parte de um repositório de uma ferramenta de controle de versão adotada pelo projeto. As alterações realizadas ao Plano devem ser descritas na ferramenta;
- Documento de Requisitos: estar sob gerência de configuração.

O maior rigor sobre o Documento de Requisitos se deve ao fato de que a grande maioria das mudanças a este documento impactam diretamente o trabalho de toda a equipe. Desta forma, estas mudanças devem ser realizadas de forma controlada e serem comunicadas.

5.2.2 RM.Ac3

Prática	**Descrição**
Atividade 3	**Mudanças aos requisitos alocados são revisadas e incorporadas ao projeto de software**
Diagnóstico	**Problema**
Satisfaz Parcialmente	Apesar de a equipe revisar as mudanças antes de incorporá-las, esta revisão não inclui a alta gerência, os grupos afetados e não são documentadas de forma a manter um histórico das mudanças.

Solução. O Plano do Projeto e o Documento de Requisitos deverão ser revisados/aprovados pela alta gerência da organização (gerência sênior), representantes dos grupos afetados, se houverem, e representantes do cliente.

Como já mencionado, mudanças ao Plano do Projeto deverão ser realizadas pelo Gerente do Projeto e documentadas na ferramenta de controle de versão que o plano estará submetido. Mudanças críticas, deverão ser submetidas à nova revisão/aprovação da

gerência sênior da organização, representantes dos grupos afetados, se houverem, e representantes do cliente. O Plano do Projeto deverá conter critérios para determinar o que é considerada uma mudança crítica. Exemplos de critérios são: mudanças que impactem o Documento de Requisitos, variação de custo e cronograma em mais ou menos um percentual definido.

Como o Documento de Requisitos estará sob gerência de configuração, as mudanças devem seguir o rigor do processo de gerência de configuração estabelecido. Assim, solicitações formais de mudança devem ser registradas, revisadas, aprovadas, e acompanhadas.

5.2.3 Sumário do Guia para Implementação da KPA

A Tabela 5-1 relaciona resumidamente os problemas identificados pelo Diagnóstico XP-CMM2 e respectiva solução sugerida pelo Guia XP-CMM2 para a KPA Gerenciamento de Requisitos.

Prática	Problema	Solução
Atividade 2	– XP utiliza as estórias para guiar o andamento do projeto, mas não menciona a documentação e uso dos requisitos não técnicos e o controle das versões dos requisitos documentados.	– Requisitos técnicos e não técnicos deverão ser documentados no Plano do Projeto e o detalhamento funcional no Documento de Requisitos. – O Documento de Requisitos deverá estar sob gerência de configuração. – O Plano do Projeto deverá estar num repositório em uma ferramenta de controle de versão. Mudanças devem ser documentadas nesta ferramenta.
Atividade 3	– Mudanças aos requisitos são revisadas com a equipe, no entanto, mudanças aos compromissos estabelecidos com	– O Plano do Projeto e o Documento de Requisitos deverão ser revisados/aprovados pelos responsáveis.

	grupos externos à organização não são levadas à gerência sênior.	– Mudanças aos requisitos documentados no Plano do Projeto deverão gerar atualizações no Plano, realizadas pelo gerente do projeto. Quando críticas, estas mudanças levarão o Plano à aprovação pela gerência sênior. – Mudanças aos requisitos documentados no Documento de Requisitos deverão seguir o sistema de gerência de configuração instalado.

Tabela 5-1 - Sumário do Guia para Implementação da KPA RM

5.3 Solução para a KPA Planejamento de Projetos de Software

Para cada problema identificado pelo Diagnóstico XP-CMM2 em relação à KPA Planejamento de Projeto de Software uma solução será proposta.

5.3.1 SPP.Ac4

Prática	Descrição
Atividade 4	**Compromissos do projeto de software feitos para indivíduos e grupos externos à organização são revisados com a gerência sênior de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não define o relacionamento entre a gerência sênior e o projeto.

Solução. Os compromissos externos à organização e ao projeto (compromissos com outros grupos da organização) deverão estar documentados no Plano do Projeto, o qual deverá ser revisado e aprovado pela gerência sênior, como já citado.

5.3.2 SPP.Ac6

Prática	Descrição

Atividade 6	**O plano de desenvolvimento do projeto de software é desenvolvido de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Satisfaz Parcialmente	A descrição de XP deve ser considerada um procedimento documentado para o planejamento do projeto, no entanto, ela não aborda aspectos importantes do planejamento, como: planejamento a partir de padrões estabelecidos, envolvimento dos grupos afetados, revisão do plano por parte de grupos afetados e cliente, gerenciamento e controle dos planos.

Solução. De acordo com a solução já citada, o Plano do Projeto deverá descrever compromissos com grupos afetados externos ao projeto, deverá ser aprovado pela gerência sênior e representantes destes grupos e deverá fazer parte de um repositório de controle de versão, onde será gerenciado e controlado. Além disto, o Plano do Projeto deverá descrever os padrões adotados para o projeto.

5.3.3 SPP.Ac7

Prática	**Descrição**
Atividade 7	**O plano para o projeto de software é documentado**
Diagnóstico	**Problema**
Satisfaz Parcialmente	Os três planos definidos por XP não contemplam todas as informações importantes e esperadas pelo CMM como propósito, escopo, metas e objetivos do projeto de software, procedimentos, métodos e padrões para desenvolvimento e/ou manutenção do software, entre outros.

Solução. Este trabalho sugere que os documentos Plano do Projeto e Documento de Requisitos substituam os planos *Big Plan* e Plano de *Release*.

O Plano do Projeto deverá conter informações sobre todo o projeto, conforme sugerido pelo CMM. Exemplos destas informações são: propósito; meta e objetivos do projeto; ciclo de vida utilizado (o ciclo definido por XP), procedimentos, padrões e métodos a serem adotados para o projeto; estimativas de custo; riscos; planejamento das instalações; equipe; recursos críticos computacionais; compromissos com grupos externos; requisitos não técnicos e critérios de aceitação de requisitos; referência para o Documento de Requisitos.

O Documento de Requisitos deverá conter o Plano de *Releases*. Inicialmente conterá as informações do *Big Plan*: breve descrição das estórias e estimativas preliminares. À medida que o projeto progride, conterá o detalhamento do próximo *release*: estórias mais detalhadas e estimativas mais confiáveis.

5.3.4 SPP.Ac9

Prática	Descrição
Atividade 9	**Estimativas para o tamanho dos produtos de trabalho de software (ou mudanças aos tamanhos dos produtos de trabalho de software) são derivados de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não contempla estimativa de tamanho dos produtos de trabalho.

Solução. XP não realiza estimativas de tamanho, mas de esforço para as tarefas e estórias, no entanto, a forma como XP é estruturada facilita a eficácia das estimativas e seu aprimoramento. Como tarefas são unidades de estórias que devem caber em uma iteração, elas têm uma duração, no pior caso, de no máximo a duração da iteração. Como as iterações devem durar de duas a três semanas, esta é a duração máxima de uma tarefa. Se a estimativa não couber neste prazo, a estória deve ser quebrada em outras estórias. Desta forma, projetos XP sempre estimam unidades pequenas de atividades. Paulk afirma em [33] que "no nível de iterações de duas semanas, estas estimativas são tipicamente exatas".

Além disto, o responsável por realizar a tarefa é responsável por estimá-la. O que, provavelmente, contribui para o acerto das próximas estimativas. Um outro fator de XP que contribui bastante na melhoria das estimativas é que os projetos são divididos em unidades muito pequenas de trabalho: as iterações. A cada iteração completada, dados históricos e experiência são acumulados para apoiar as estimativas futuras.

Paulk, em [33], afirma que "para pequenos projetos, esforço e cronograma podem ser estimados diretamente do *work breakdown structure*, e as conseqüências de o projeto falhar serão relativamente menores". Esta afirmação ocorre num projeto XP, onde esforço é estimado diretamente a partir da lista de tarefas identificadas.

Por tudo isto, apesar de XP não atender a prática SPP.Ac9, este trabalho não irá sugerir nenhuma alteração quanto ao problema da não realização de estimativa de

tamanho. Em uma avaliação oficial, esta prática deveria ser considerada como Prática Alternativa.

5.3.5 SPP.Ac10

Prática	Descrição
Atividade 10	**Estimativas para esforço e custo do projeto de software são derivadas de acordo com procedimento documentado**
Diagnóstico	**Problema**
Satisfaz Parcialmente	XP define como estimar esforço, mas não relaciona estas estimativas com as estimativas de tamanho. Além disto, não define de forma satisfatória como estimar custos.

Solução. Um dos problemas identificados nesta prática foi o de XP não associar as estimativas de esforço às estimativas de tamanho. Este trabalho sugere adotar o procedimento de XP na íntegra, mesmo que não estimando tamanho como requerido pela prática SPP.Ac9. Desta forma, este trabalho também não irá propor uma solução para o problema da falta de relação entre as estimativas de tamanho e esforço.

Para a estimativa de custo, este trabalho sugere que uma planilha simples seja utilizada para estimar os custos relacionados a categorias como: treinamentos, pessoal, instalações, viagens, consultoria.

5.3.6 SPP.Ac11

Prática	Descrição
Atividade 11	**Estimativas para os recursos computacionais críticos do projeto são derivadas de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não aborda a questão de planejar e estimar os recursos computacionais críticos.

Solução. O Plano do Projeto deverá possuir uma seção para identificação e estimativa dos recursos críticos computacionais. Como a estimativa de recurso crítico computacional é bastante dependente do tipo de recurso crítico e este do projeto, este trabalho também sugere que o próprio Plano do Projeto descreva ou faça referência ao procedimento utilizado para estimar o recurso crítico.

5.3.7 SPP.Ac13

Prática	Descrição
Atividade 13	**Os riscos de software associados com custo, recursos, cronograma e aspectos técnicos do projeto são identificados, avaliados e documentados**
Diagnóstico	**Problema**
Não Satisfaz	XP não aborda a questão de riscos do projeto de software.

Solução. Riscos do projeto deverão ser identificados, priorizados e planos de contingência deverão ser elaborados para cada risco identificado. Este trabalho sugere que tudo isto seja feito no próprio Plano do Projeto para que os grupos afetados sejam informados a respeito.

5.3.8 SPP.Ac14

Prática	Descrição
Atividade 14	**Planos para as instalações e ferramentas de apoio para a engenharia de software do projeto são preparados**
Diagnóstico	**Problema**
Não Satisfaz	XP não contempla planos explícitos para as instalações e ferramentas de apoio do projeto de software.

Solução. O Plano do Projeto deverá conter seções para planejamento das instalações.

5.3.9 SPP.Ac15

Prática	Descrição
Atividade 15	**Dados do planejamento de software são registrados**
Diagnóstico	**Problema**
Satisfaz Parcialmente	XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados ao planejamento.

Solução. Como já citado, o Plano do Projeto deverá estar num repositório de uma ferramenta de controle de versão e mudanças a ele deverão ser registradas. As estimativas das estórias estarão registradas no Documento de Requisitos, que estará sob gerência de configuração.

5.3.10 Sumário do Guia para Implementação da KPA

A Tabela 5-2 relaciona resumidamente os problemas identificados pelo Diagnóstico XP-CMM2 e respectiva solução sugerida pelo Guia XP-CMM2 para a KPA Planejamento de Projeto de Software.

Prática	Problema	Solução
Atividade 4	– XP não define o relacionamento entre a gerência sênior e o projeto.	– Como já citado, os compromissos externos deverão estar documentados no Plano do Projeto, o qual deverá ser revisado e aprovado pela gerência sênior.
Atividade 6	– XP não aborda aspectos importantes do planejamento, como: planejamento a partir de padrões estabelecidos, envolvimento dos grupos afetados, revisão do plano por parte de grupos afetados e cliente, gerenciamento e controle dos planos.	– Como já citado, o Plano do Projeto deverá descrever padrões adotados, compromissos com grupos externos afetados, deverá ser aprovado pela gerência sênior e representantes destes grupos e deverá fazer parte de um repositório de controle de versão.
Atividade 7	– Os três planos definidos por XP não contemplam todas as informações importantes e esperadas pelo CMM.	– O Plano do Projeto e Documento de Requisitos substituirão os planos Big Plan e Plano de Release. – O Plano do Projeto deverá conter informações sobre todo o projeto, conforme sugerido pelo CMM. – O Documento de Requisitos deverá conter o Plano de Releases.
Atividade 9	– XP não contempla estimativa de tamanho dos produtos de trabalho.	– Este trabalho não irá sugerir nenhuma alteração quanto ao problema da não realização de

		estimativa de tamanho. Em uma avaliação oficial, esta prática deveria ser considerada como Prática Alternativa.
Atividade 10	– XP não relaciona as estimativas de esforço e custo com as estimativas de tamanho.	– Como este trabalho não irá sugerir realização de estimativa de tamanho, a estimativa de esforço não poderá ser relacionada à de tamanho. Assim, este trabalho não propõe solução para este problema. – Os custos deverão ser estimados em categorias e registrados em planilhas.
Atividade 11	– XP não aborda a questão de planejar e estimar os recursos computacionais críticos.	– O Plano do Projeto deverá possuir uma seção para identificação, procedimento de estimativa e estimativa realizada para os recursos críticos computacionais.
Atividade 13	– XP não aborda a questão de riscos do projeto de software.	– O Plano do Projeto deverá relacionar os riscos do projeto, priorizados e respectivos planos de contingência.
Atividade 14	– XP não contempla planos explícitos para as instalações e ferramentas de apoio do projeto de software.	– O Plano do Projeto deverá conter seções para planejamento das instalações.
Atividade 15	– XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não	– Como já citado, o Plano do Projeto deverá estar num repositório de uma ferramenta de controle de versão e mudanças a

	menciona o controle das versões dos produtos de trabalho associados ao planejamento.	ele deverão ser registradas. As estimativas das estórias estarão registradas no Documento de Requisitos, que estará sob gerência de configuração.

Tabela 5-2 - Sumário do Guia para Implementação da KPA SPP

5.4 Solução para a KPA Acompanhamento de Projetos de Software

Para cada problema identificado pelo Diagnóstico XP-CMM2 em relação à KPA Acompanhamento de Projeto de Software uma solução será proposta.

5.4.1 SPTO.Ac3

Prática	Descrição
Atividade 3	**Compromissos e mudanças aos compromissos do projeto de software feitos a indivíduos e grupos externos à organização são revisados com a gerência sênior de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não define o relacionamento entre a gerência sênior e o projeto.

Solução. Como já citado, os compromissos externos deverão estar documentados no Plano do Projeto, que ao sofrer alterações críticas, passará a ser novamente revisado/aprovado pela gerência sênior da organização. Novamente, é importante que o Plano defina critérios para definir o que é mudança crítica.

Além disto, este trabalho sugere que o Plano do Projeto e Documento de Requisitos sejam atualizados e novamente revisados/aprovados durante a reunião e planejamento do *release*, pois o planejamento do próximo *release* irá ocasionar mudanças no Documento de Requisitos (detalhamento das estórias e estimativas do próximo *release*) e poderá gerar mudanças no Plano do Projeto, como a identificação de novos riscos, novos compromissos externos, mudanças em cronograma (data do próximo *release*) e outros.

5.4.2 SPTO.Ac5

Prática	Descrição
Atividade 5	**Os tamanhos dos produtos de trabalho de software (ou tamanhos das mudanças aos produtos de trabalho de software) são acompanhadas, e ações corretivas são tomadas quando necessário**
Diagnóstico	**Problema**
Não Satisfaz	XP não aborda estimativa de tamanho de produto de trabalho, tão pouco seu acompanhamento.

Solução. Como este trabalho sugere não estimar tamanho (prática SPP.Ac9), esta prática torna-se não aplicável, já que não é possível acompanhar uma estimativa que não é realizada.

5.4.3 SPTO.Ac6

Prática	Descrição
Atividade 6	**Esforço e custo de software do projeto são acompanhados, e ações corretivas são tomadas quando necessário.**
Diagnóstico	**Problema**
Satisfaz Parcialmente	Apesar de acompanhar esforço e promover a tomada de ação corretiva, XP não menciona o acompanhamento de custos.

Solução. Apesar de o CMM não requerer acompanhamento periódico, este trabalho sugere que os custos sejam acompanhados periodicamente, ao final de cada iteração, por categoria (pessoal, viagens, treinamentos e outros) e que este acompanhamento seja registrado.

5.4.4 SPTO.Ac7

Prática	Descrição
Atividade 7	**Recursos computacionais críticos do projeto são acompanhados e ações corretivas são tomadas quando necessário**
Diagnóstico	**Problema**
Não Satisfaz	XP não aborda de forma explícita o planejamento e acompanhamento de recursos críticos computacionais.

Solução. O gerente do projeto deverá acompanhar os recursos críticos computacionais estimados e atualizar a seção no Documento de Requisitos, pois este documento contém as estimativas do projeto e respectivo acompanhamento. Como as iterações são curtas, este trabalho sugere que esta atividade seja feita ao final de cada iteração, no entanto, de acordo com a criticidade do recurso crítico, este intervalo pode ser menor. Vale ressaltar, que não é necessário que o acompanhamento seja periódico, apesar de que fazê-lo periodicamente contribui na institucionalização.

5.4.5 SPTO.Ac10

Prática	**Descrição**
Atividade 10	**Os riscos de software associados a custo, recursos, cronograma e aspectos técnicos do projeto são acompanhados**
Diagnóstico	**Problema**
Não Satisfaz	XP não aborda a identificação e acompanhamento dos riscos do projeto.

Solução. O gerente do projeto deverá acompanhar os riscos e atualizar a seção de riscos do Plano do Projeto. Este trabalho sugere que esta atividade seja feita na reunião de planejamento da próxima iteração e *release* e sempre que um novo risco surgir e uma situação que modifique o status dos riscos identificados ocorra.

O fato de os riscos serem tratados na reunião de planejamento da iteração contribui para que toda a equipe se envolva no gerenciamento de riscos: identificando novos riscos, formas de contingenciá-los e informando o status.

5.4.6 SPTO.Ac11

Prática	**Descrição**
Atividade 11	**Dados de medida e dados de replanejamento reais para o projeto de software são registrados**
Diagnóstico	**Problema**
Satisfaz Parcialmente	XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados a replanejamento.

Solução. Este problema possui a mesma solução empregada para SPP.Ac15.

5.4.7 SPTO.Ac13

Prática	Descrição
Atividade 13	**Revisões formais para endereçar realizações e resultados do projeto de software são conduzidas nos marcos selecionados do projeto de acordo com procedimento documentado**
Diagnóstico	**Problema**
Satisfaz Parcialmente	As reuniões de planejamento de *release* ocorrem em pontos significativos do projeto, contam com a participação de cliente e equipe, relaciona planos, mas não aborda riscos do projeto.

Solução. Como o Plano do Projeto será revisado/aprovado durante a reunião de planejamento do *release* e ele contempla os riscos do projeto, então os riscos serão revisados nas revisões formais, no caso, as reuniões de planejamento do próximo *release*.

5.4.8 Sumário do Guia para Implementação da KPA

A Tabela 5-3 relaciona resumidamente os problemas identificados pelo Diagnóstico XP-CMM2 e respectiva solução sugerida pelo Guia XP-CMM2 para a KPA Acompanhamento de Projeto de Software.

Prática	Problema	Solução
Atividade 3	– XP não define o relacionamento entre a gerência sênior e o projeto.	– Como já citado, os compromissos externos deverão estar documentados no Plano do Projeto, que ao sofrer alterações críticas, passará a ser novamente revisado/aprovado pela gerência sênior da organização. – O Plano do Projeto e Documento de Requisitos devem ser atualizados e novamente revisados/aprovados durante a reunião e planejamento do release.
Atividade 5	– XP não aborda estimativa de tamanho de produto de trabalho,	– Como este trabalho sugere não estimar tamanho (prática

	tão pouco seu acompanhamento.	SPP.Ac9), esta prática torna-se não aplicável.
Atividade 6	– Apesar de acompanhar esforço e promover a tomada de ação corretiva, XP não menciona o acompanhamento de custos.	– Este trabalho sugere que os custos sejam acompanhados periodicamente, ao final de cada iteração, por categoria e que este acompanhamento seja registrado.
Atividade 7	– XP não aborda estimativa de tamanho de produto de trabalho, tão pouco seu acompanhamento.	– O gerente do projeto deverá acompanhar os recursos críticos computacionais estimados e atualizar a seção no Plano do Projeto.
Atividade 10	– XP não aborda a identificação e acompanhamento dos riscos do projeto.	– O gerente do projeto deverá acompanhar os riscos e atualizar a seção de riscos do Plano do Projeto.
Atividade 11	– XP armazena os dados de estimativas e premissas associados nos próprios cartões de estórias e tarefas, no entanto, não menciona o controle das versões dos produtos de trabalho associados a replanejamento.	– Como já citado, o Plano do Projeto deverá estar num repositório de uma ferramenta de controle de versão e mudanças a ele deverão ser registradas. As estimativas das estórias estarão registradas no Documento de Requisitos, que estará sob gerência de configuração.
Atividade 13	– As reuniões de planejamento de *release* ocorrem em pontos significativos do projeto, contam com a participação de cliente e equipe, relaciona planos, mas não aborda riscos do projeto.	– Como o Plano do Projeto será revisado/aprovado durante a reunião de planejamento do release e ele contempla os riscos do projeto, então os riscos serão revisados nas revisões formais.

Tabela 5-3 - Sumário do Guia para Implementação da KPA SPTO

5.5 Solução para a KPA Garantia da Qualidade de Software

Segundo Paulk, "um grupo independente de garantia da qualidade é inpensável na cultura de XP" [33]. Reifer relata o caso de um projeto XP numa organização nível 3 do CMM [10]. Nesta organização, alguns problemas emergiram entre a equipe do projeto e o grupo de garantia da qualidade, que não trabalhavam de forma cooperativa. Como conclusão, Reifer afirma que "ninguém parecia aplicar XP dentro de um contexto SW-CMM racionalmente e de forma proveitosa..." e "eles poderiam ter modificado o processo de garantia da qualidade para que tivesse um papel mais útil no projeto, especialmente em assegurar aderência às práticas de testes de XP".

Como não há intercessão entre XP e a KPA Garantia da Qualidade de Software, este trabalho não irá sugerir implementações para as práticas desta KPA. As organizações devem implementar o sistema de garantia de qualidade de acordo com suas necessidades e cultura. Apesar disto, algumas considerações serão feitas para que o sistema de garantia da qualidade definido pelas organizações não impactem negativamente um projeto XP.

Para atender as práticas da KPA, é preciso montar um sistema de garantia da qualidade aderente ao CMM, com plano de garantia da qualidade, revisões e/ou auditorias de processo e produto. Os processos de planejamento, acompanhamento, controle dos requisitos e gerência de configuração devem ser verificados e os problemas levantados, corrigidos. O grupo de garantia da qualidade também deve atuar sobre os padrões adotados pelo projeto, verificando o processo de testes e padrões adotados pelo projeto.

A atenção que se deve ter é que um projeto XP é essencialmente colaborativo e assim deve agir o grupo de garantia da qualidade. O grupo deve estar atento ao que é realmente necessário seguir num projeto XP para que a equipe do projeto não dedique esforço a corrigir problemas que não agreguem valor ao projeto. Uma sugestão para reforçar esta atitude é o grupo de garantia da qualidade levantar o risco associado a cada problema identificado nas auditorias e/ou revisões.

Paulk sugere em [33] que, para pequenos projetos e organizações, a função de SQA pode estar inserida no processo através de *checklists* e ferramentas, por exemplo.

Mesmo utilizando esta abordagem, Paulk ressalta que é importante definir um mecanismo de escalação para resolver conflitos a respeito dos problemas encontrados [33]. Num projeto XP, é interessante explorar esta sugestão, para que o grupo de SQA trabalhe de forma mais dedicada ao que realmente é importante do ponto de vista de processo e padrões e agregue mais valor ao projeto.

5.6 Solução para a KPA Gerenciamento de Configuração de Software

Como já citado, num projeto XP, problemas ou melhorias ao software (item de configuração de um projeto XP) são identificadas, documentadas e tratadas como uma estória, registrada em cartões de papel, revisadas e aprovadas pela equipe [25]. O diagnóstico considerou que esta prática de XP atende a prática SCM.Ac5 do CMM ("**Requisição de mudança e reportagem de problema para todos os itens de configuração são iniciados, registrados, revisados, aprovados e acompanhados de acordo com um procedimento documentado.**"). Apesar disto, esta abordagem pode acarretar em problemas devido à falta de registro do histórico das mudanças ocorridas, pois elas são tratadas da mesma forma que novas estórias.

Mudanças podem ocorrer, a todo o momento, por requisição do cliente (ele muda a prioridade das estórias de um *release*, por exemplo) ou por necessidade da equipe (quando ela não consegue, por exemplo, implementar todas as estórias de uma iteração). Não há como saber em um *release* do projeto o porquê da não implementação de estórias acordadas inicialmente: se a responsabilidade foi da indefinição do cliente ou de más estimativas realizadas pela equipe do projeto.

Como já citado no Capítulo 2, uma das críticas à XP é que muitas vezes não é possível contar com representantes do cliente na equipe. Outras, quando é possível, este representante não tem autoridade para determinar o escopo do sistema, selecionando o que será e o que não será desenvolvido. Mesmo quando há este cliente com autoridade, é possível que o resultado final de um *release* não atenda às expectativas do cliente porque estórias não foram implementadas e o *Big Plan* (a expectativa) inicial não foi cumprido. Neste caso, a falta de registro das mudanças: o porque foram realizadas, quem as

solicitou e quem as aprovou, por exemplo, poderá comprometer a aceitação do produto final e a satisfação do cliente.

Para diminuir estes riscos, este trabalho sugere o uso de uma ferramenta de controle de mudanças. Como exemplo, a ferramenta *Bugzilla* [46] é um ferramenta *freeware* utilizada para reportagem de *bugs* que pode também ser usada para registro de mudanças. O objetivo é que qualquer pessoa da equipe ou externa (inclusive cliente) solicite mudanças aos itens de configuração através da ferramenta de controle de mudanças. Estas mudanças, se aprovadas pelo SCBB, viram estórias e entram no fluxo de um projeto XP. Este SCCB, para ser o mais flexível possível, pode ser composto apenas pelo gerente do projeto (desde que ele seja da área de software, para responder a questões técnicas), pois é ele quem responde pelo sucesso do projeto. Requisições de mudanças não urgentes podem ser avaliadas em conjunto com toda a equipe nas reuniões de início da iteração.

É importante ressaltar que solicitações de mudança só devem ser abertas para itens de configuração nas *baselines*, ou seja, no que é considerado estável. Desta forma, é de extrema importância definir quando as *baselines* serão estabelecidas.

5.6.1 SCM.Ac1

Prática	**Descrição**
Atividade 1	**Um plano de SCM é preparado para cada projeto de software de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona plano para o gerenciamento de configuração.

Solução. Um plano de gerência de configuração deverá ser elaborado contendo: itens de configuração, critério para selecioná-los, baselines, conteúdo das baselines, momento de entrada dos itens de configuração nas baselines, comitê que aprova as mudanças e estabelecimento das baselines, planejamento de releases, planejamento de auditorias, padrões de gerência de configuração a serem seguidos, relatórios a serem gerados. Quanto menor a equipe do projeto, mais simples poderá ser este plano. Este trabalho sugere um Template para o Plano de Gerência de Configuração (Apêndice D).

É muito importante que este plano contemple uma gerência de configuração simples para não quebrar a dinâmica de um projeto XP. Para Beck, um dos benefícios da prática propriedade coletiva é que se alguém precisa de uma mudança, ela mesma pode fazer, não precisa requisitar e esperar que o responsável pelo produto realize a alteração necessária [25], pois, segundo ele, sem a propriedade coletiva, "o código permanece estável, mas não evolui rápido como deveria" [25]. Por este motivo, o controle das mudanças aos itens de configuração deve ser planejado de forma a não comprometer sua evolução.

5.6.2 SCM.Ac2

Prática	**Descrição**
Atividade 2	**Um plano de SCM documentado e aprovado é utilizado como base para a realização das atividades de SCM**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona plano para o gerenciamento de configuração.

Solução. O plano de gerência de configuração deverá ser aprovado pelo gerente do projeto. Assim como o plano do projeto, em mudanças críticas, este plano deverá ser reaprovado. O plano deverá descrever as mudanças que requerem reaprovação.

5.6.3 SCM.Ac3

Prática	**Descrição**
Atividade 3	**Um sistema de biblioteca de gerenciamento de configuração é estabelecido como um repositório para as baselines de software**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona o estabelecimento de um sistema de gerenciamento da configuração.

Solução. Esta prática é facilmente atendida quando adotada uma ferramenta de controle de versão. Ao selecionar uma ferramenta, deve-se verificar se ela atende às subpráticas desta prátiva. Caso não atenda, deve-se verificar a possibilidade de adotar uma outra ferramenta ou elaborar *scripts* para complementá-la.

Uma sugestão é o CVS [49], que é uma ferramenta *freeware*, largamente utilizada mundo afora.

5.6.4 SCM.Ac4

Prática	Descrição
Atividade 4	**Os produtos de trabalho de software a serem colocados sob gerência de configuração são identificados**
Diagnóstico	**Problema**
Satisfaz Parcialmente	Se o projeto possuir outros itens de configuração além de código, várias das subpráticas não são atendidas.

Solução. O plano de gerência de configuração deverá listar os itens de configuração do projeto. De acordo com a proposta deste trabalho minimamente o código e o Documento de Requisitos deverão ser itens de configuração.

Critérios sugeridos para identificar os itens de configuração do projeto são [3]:

- Produtos entregues ao cliente ou a outra organização;
- Produtos considerados críticos para o projeto;
- Artefatos utilizados por mais de um grupo;
- Artefatos produzidos por terceiros, que serão incorporados ao projeto;
- Artefatos que, ao serem alterados, podem causar mudanças em outros artefatos associados.

5.6.5 SCM.Ac8

Prática	Descrição
Atividade 8	**O status dos itens/unidades de configuração é registrado de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona o registro do *status* dos itens de configuração.

Solução. As mudanças aos itens de configuração deverão ser registradas na ferramenta de controle de versão adotada sempre que o item for submetido ao sistema de biblioteca de gerência de configuração. Os projetos poderão estabelecer padrões para estes registros no Plano de Gerência de Configuração.

5.6.6 SCM.Ac9

Prática	Descrição
Atividade 9	**Relatórios padrões documentando as atividades e SCM e conteúdo das baselines são desenvolvidos e tornados disponíveis para os grupos e indivíduos afetados**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona o uso de relatórios de gerência de configuração e disponibilização aos grupos afetados.

Solução. Com a utilização de ferramentas, facilmente são obtidos relatórios de gerência de configuração. As seguintes informações, sugeridas pelo CMM, podem ser coletadas a partir da implementação do Guia XP-CMM2:

- Solicitações de mudança e respectivos *status* – obtidos a partir de consultas à ferramenta de controle de mudança adotada;
- Mudanças as baselines – obtidas a partir de consultas à ferramenta de controle de mudança adotada;
- Histórico de alteração dos itens de configuração – obtido a partir de consultas à ferramenta de controle de versão adotada;
- *Status* da baseline – o conteúdo das *baselines* pode ser obtido a partir de consultas à ferramenta de controle de versão adotada;
- Relatórios de auditoria nas *baselines* – a serem gerados pelo gerente de configuração do projeto (ver Seção 5.6.7).

Mesmo ferramentas simples de controle de versão e mudanças oferecem consultas como as citadas.

5.6.7 SCM.Ac10

Prática	Descrição
Atividade 10	**Auditorias nas baselines de software são conduzidas de acordo com um procedimento documentado**
Diagnóstico	**Problema**
Não Satisfaz	XP não menciona auditorias nas *baselines* de software.

Solução. Segundo o CMM, estas auditorias tipicamente checam: a integridade das *baselines*, estrutura e instalações do sistema de biblioteca de gerência de configuração, conformidade aos padrões e outros. Estas auditorias podem ser automatizadas facilmente, de acordo com as ferramentas e padrões de gerência de configuração adotados. Este trabalho sugere a criação de *scripts* para realização das auditorias, com geração automática de relatórios descrevendo os problemas. Estes relatórios deverão ser enviados para toda a equipe. O gerente de configuração, gerente do projeto ou grupo de garantia da qualidade poderá ser responsável por acompanhar a resolução dos problemas.

5.6.8 Sumário do Guia para Implementação da KPA

A Tabela 5-4 relaciona resumidamente os problemas identificados pelo Diagnóstico XP-CMM2 e respectiva solução sugerida pelo Guia XP-CMM2 para a KPA Gerenciamento de Configuração de Software.

Além destes itens, este trabalho sugere a adoção de uma ferramenta de controle de mudança, registro e avaliação formal das solicitações de mudanças às *baselines* do projeto.

Prática	Problema	Solução
Atividade 1	– XP não menciona plano para o gerenciamento de configuração.	– Um plano de gerência de configuração deverá ser elaborado.
Atividade 2	– XP não menciona plano para o gerenciamento de configuração.	– O plano de gerência de configuração deverá ser aprovado pelo gerente do projeto.
Atividade 3	– XP não menciona o estabelecimento de um sistema de gerenciamento da configuração.	– Adotar ferramenta de controle de versão.
Atividade 4	– Se o projeto possuir outros itens de configuração além de código, várias das subpráticas não são atendidas.	– O plano de gerência de configuração deverá listar os itens de configuração do projeto e o critério utilizado para selecioná-los.

Atividade 8	– XP não menciona o registro do *status* dos itens de configuração.	– As mudanças aos itens de configuração deverão ser registradas na ferramenta de controle de versão adotada.
Atividade 9	– XP não menciona o uso de relatórios de gerência de configuração e disponibilização aos grupos afetados.	– Relatórios serão obtidos a partir das ferramentas adotadas para gerência de configuração.
Atividade 10	– XP não menciona auditorias nas *baselines* de software.	– Auditorias nas *baselines* deverão ser realizadas preferencialmente através de *scripts* e geração automática de relatório.

Tabela 5-4 - Sumário do Guia para Implementação da KPA SCM

5.7 Visão Geral da Solução

Esta seção descreve resumidamente o tratamento a cada KPA contemplando todas as sugestões propostas pelo Guia XP-CMM2.

5.7.1 Planejamento de Projeto de Software

Um projeto XP deverá elaborar dois documentos em seu início: o Plano do Projeto e o Documento de Requisitos, que deverão ser refinados no decorrer do projeto.

O Plano do Projeto deverá contemplar: propósito; meta e objetivos do projeto; ciclo de vida utilizado (o ciclo definido por XP), procedimentos, padrões e métodos a serem adotados para o projeto; estimativas de custo; riscos; planejamento das instalações; equipe; recursos críticos computacionais e procedimento para estimá-los; compromissos com grupos externos à organização e ao projeto; requisitos técnicos e não técnicos e critérios de aceitação de requisitos; referência para o Documento de Requisitos.

O Plano do Projeto deverá ser elaborado pelo Gerente do Projeto e submetido à revisão e aprovação do gerente sênior da organização, representantes de grupos externos ao projeto afetados pelo projeto e representantes do cliente. A revisão/aprovação é um mecanismo para promover o acordo entre as partes envolvidas, os compromissos firmados.

O Documento de Requisitos contemplará, inicialmente, os requisitos não funcionais (estórias) e o conteúdo do *Big Plan*: breve descrição das estórias e estimativas preliminares. Após a reunião de planejamento de cada *release*, este documento será refinado com as informações da reunião.

O planejamento do *release* continuará a ser conduzido como descrito por XP, iniciando com a reunião de planejamento de *release*, onde se seleciona as estórias do próximo *release* e as estima. As estórias poderão ser documentadas em cartões de papel como sugere XP, no entanto, ao final da reunião, o Documento de Requisitos deverá ser atualizado com o resultado do planejamento: as estórias do *release* e suas estimativas. Os cartões de estórias deverão ser descartados para não gerar inconsistências com o Documento de Requisitos.

O plano do *release* não mais é o conjunto de cartões, mas o Documento de Requisitos. O Documento de Requisitos também deverá ser submetido à revisão e aprovação do gerente sênior da organização, representantes de grupos externos ao projeto afetados pelo projeto e representantes do cliente, junto com o Plano do Projeto. Esta proposta segue a linha XP de não criar um planejamento detalhado para todo o projeto em seu início.

A reunião de planejamento da iteração não necessitou de nenhum ajuste. Os requisitos identificados/documentados nesta reunião continuam sendo tratados exatamente como descrito por XP, pois são bastante detalhados e focados na engenharia do produto. As mudanças às tarefas e estórias que não afetarem o Plano de *Release* (o Documento de Requisitos) seguem o fluxo definido por XP. Apesar de que, este trabalho sugere que as tarefas sejam registradas em meio digital para facilitar registro de dados históricos para outros projetos, acompanhamento mais fácil das estimativas das tarefas, segurança da informação através de *backups*. Este registro pode ser tão simples como o registro do plano de *release* em planilhas.

5.7.2 Gerenciamento da Configuração de Software

O projeto deverá planejar como se dará a gerência de configuração no projeto, identificado itens de configuração, *baselines*, SCCBs, entre outros em um Plano de Gerência de Configuração.

Ferramentas de controle de versão e de mudanças deverão ser adotadas pelo projeto para facilitar a gerência de configuração. Informações sobre as mudanças e versões dos itens de configuração deverão ser registradas nas ferramentas adotadas.

Auditorias deverão ser realizadas periodicamente, preferencialmente através de *scripts* automatizados.

O Documento de Requisitos deverá ser um item de configuração do projeto. O Plano do Projeto não será um item de configuração do projeto, mas deverá estar na ferramenta de controle de versão adotada para o projeto e todas as alterações realizadas sobre eles deverão ser registradas na própria ferramenta.

5.7.3 Garantia da Qualidade de Software

Um sistema de garantia da qualidade deverá ser definido e implantado num projeto XP.

5.7.4 Acompanhamento de Projeto de Software

O Plano do Projeto e o Documento de Requisitos deverão ser revisados ao final de cada *release* e início de planejamento do próximo *release* pela gerência sênior, representantes do cliente e grupos afetados. Além disto, o Plano deverá conter critérios para determinar quando mudanças realizadas antes do fim de um *release* precisam ser revisadas.

Recursos críticos computacionais e riscos deverão ser acompanhados pelo gerente do projeto e este acompanhamento registrado no Documento de Requisitos e Plano do Projeto respectivamente. Os riscos deverão ser tratados com toda a equipe na reunião de planejamento da iteração e de *release*. Sugere-se que o acompanhamento dos recursos críticos computacionais seja feito ao final de cada iteração. Custos deverão ser acompanhados e este acompanhamento registrado.

5.7.5 Gerenciamento de Requisitos

Para realizar mudanças como:

- Incluir/remover uma estória no projeto ou em um *release*;
- alterar a data de *release*;
- incluir/remover requisitos não funcionais.

Será preciso abrir uma solicitação de mudança na ferramenta de controle de mudança. Esta solicitação deverá ser aprovada pelo SCCB do projeto, documentado no Plano de Gerência de Configuração. Esta avaliação do SCCB deverá ser realizada, preferencialmente (caso a solicitação de mudança não seja urgente) na reunião de planejamento da iteração. Caso a mudança seja aprovada, o Documento de Requisitos deverá ser alterado e a mudança gerará uma estória como outra qualquer.

O controle das mudanças não tem como objetivo impedir que as mudanças ocorram, mas que elas ocorram de forma controlada, que sejam analisadas e registradas.

5.7.6 Ferramentas e *Templates*

O Guia XP-CMM2 sugere ferramentas e *templates* para apoiar a sua implantação, conforme descrito nas tabelas Tabela 5-5 e Tabela 5-6.

Ferramenta	Descrição	Exemplo
Controle de versão	Ferramenta para apoiar a gerência de configuração no controle das versões.	Ferramenta *freeware* CVS [49]
Controle de mudança	Ferramenta para apoiar a gerência de configuração no controle das mudanças.	Ferramenta *freeware* Bugzilla [46]

Tabela 5-5 - Ferramentas Sugeridas pelo Guia XP-CMM2

Template	Descrição	Localização
Plano do Projeto	Plano do projeto, descreve o planejamento macro do projeto.	**Erro! Fonte de referência não encontrada.**
Documento de Requisitos	Descreve o planejamento de releases e registra seu acompanhamento, bem como acompanhamento de recursos críticos computacionais.	Apêndice C
Plano de Gerência de Configuração	Descreve o planejamento da configuração do projeto.	Apêndice D

Tabela 5-6 - Templates Sugeridos pelo Guia XP-CMM2

5.8 Análise da Solução

Esta seção descreverá uma análise a respeito do impacto do Guia XP-CMM2 a XP, considerando práticas, príncipios e valores adotados pela metodologia.

5.8.1 Impacto do Guia XP-CMM2 às Práticas de XP

A solução proposta acarreta em:

- Mais atividades para o gerente do projeto, como: planejar e acompanhar recursos críticos computacionais, ferramentas e instalações, riscos e custos, elaborar um plano de projeto;
- Mais formalismos no gerenciamento: documentos formais e não apenas cartões de papéis que podem ser perdidos e/ou rasgados. Revisão e aprovação de alguns documentos por parte dos interessados/afetados com o projeto;
- Maior controle: planejamento da configuração do software, controle e manutenção do histórico das mudanças, garantia da qualidade.

A única prática de XP afetada pela solução é o Jogo do Planejamento, que define o planejamento do projeto. As outras práticas não foram afetadas e podem ser implementadas conforme definido por XP. São elas: *releases* pequenos, Metáforas, Projeto de software simples, Teste, *Refactoring*, Programação em pares, Propriedade coletiva, Integração contínua, 40 horas semanais, Cliente no local e Padrão de codificação.

Em relação à prática Jogo do Planejamento a sua essência não é alterada: o planejamento continua a ser orientado a pequenas entregas, detalhando-se o que será a próxima entrega e não realizando um planejamento detalhado no início do projeto. A esta prática foram adicionados documentos formais (Plano do Projeto e Documento de Requisitos) e formalismo de aprovação. Apesar de parecer que esta sugestão vai de encontro à filosofia de um projeto XP (que defende a informalidade e as mudanças sempre bem vindas, a qualquer momento), ela deverá ser aplicada apenas ao gerenciamento de mais alto nível do projeto, de forma que a dinâmica de desenvolvimento do projeto XP não seja alterada.

5.8.2 Impacto do Guia XP-CMM2 aos Valores e Princípios de XP

O valor **Comunicação** é melhorado com a revisão/aprovação de documentos formais, pois as altas gerências da organização e do cliente possuem informações consolidadas a respeito do projeto. Em um projeto XP tradicional a comunicação é forte dentro da equipe, mas não para os grupos afetados externos ao projeto.

Os outros valores não são influenciados pelo Guia XP-CMM2. São eles [25]:

- **Simplicidade** – fazer o mais simples que possa funcionar;
- ***Feedback* –** resposta sobre o produto, satisfação, qualidade e outros fatores do projeto devem ser providos a todo o momento, rapidamente;
- **Coragem** – virtude para realizar *refactorings*, implementar a solução mais simples e outras ações necessárias em um projeto XP.

Dentre os princípios relacionados por XP, o mais afetado é o "Aceitar Mudanças". Num projeto XP, mudanças são bem vindas a qualquer momento do projeto [25]. O Guia XP-CMM2 propõe o controle formal das mudanças através de solicitações formais registradas e avaliadas por um SCCB. O formalismo não deve ser visto como resistência à mudança, mas como controle. A qualquer momento, mudanças devem ser solicitadas, como sugere XP. No entanto, estas mudanças devem ter seu impacto avaliado e a integridade entre os produtos produzidos deve ser mantida. Isto é o que sugere o Guia XP-CMM2.

5.8.3 Aderência ao Nível 2 do CMM

Duas práticas foram consideradas não satisfeitas pelo Diagnóstico XP-CMM2, mas não tiveram soluções propostas pelo Guia XP-CMM2. Ambas as práticas estão relacionadas à estimativa de tamanho requerida pelo CMM, mas não sugerida pelo Guia XP-CMM2.

- SPP.Ac9: "Estimativas para o tamanho dos produtos de trabalho de software (ou mudanças aos tamanhos dos produtos de trabalho de software) são derivados de acordo com um procedimento documentado". Esta prática foi considerada não satisfeita, pois XP não contempla estimativa de tamanho dos produtos de trabalho e o Guia XP-CMM2 não sugere fazê-la;

- SPP.Ac10: “Estimativas para esforço e custo do projeto de software são derivadas de acordo com procedimento documentado”. Esta prática foi considerada não satisfeita, pois XP define como estimar esforço, mas não relaciona estas estimativas com as estimativas de tamanho. Como o Guia XP-CMM2 não sugere estimar tamanho, este problema ficou sem solução.

As práticas de SPP não atendidas estão relacionadas à seguinte meta 1: “Estimativas de software são documentadas para uso no planejamento e acompanhamento do projeto de software”. Conforme sugerido pelo Guia XP-CMM2, as estimativas são documentadas no Documento de Requisitos para uso no planejamento dos *releases* e iterações. Assim, esta meta é considerada atingida por este trabalho, mesmo sem ter as práticas SPP.Ac9 e SPP.Ac10 completamente satisfeitas.

6 Aplicação do Guia XP-CMM2

O Guia XP-CMM2 foi aplicado a um projeto real de desenvolvimento e manutenção de software, com o objetivo de avaliar o impacto da solução proposta. Esta aplicação se deu na forma de um experimento e de um estudo de caso. Experimento porque no decorrer da aplicação do Guia a um projeto, resultados foram colhidos e melhorias foram sendo adicionadas ao Guia, de forma que no final da aplicação, o Guia pôde ser considerado mais maduro. Além disto, a aplicação a um projeto deve ser vista como um estudo de caso, pois resultados e análises foram feitas sobre a solução proposta.

Este capítulo irá descrever a aplicação do Guia XP-CMM2 a um projeto real, ressaltando:

- Seção 6.1 – Objetivos da Aplicação do Guia XP-CMM2: relaciona os objetivos de aplicar o Guia XP-CMM2 a um projeto real;
- Seção 6.2 – O Ambiente: descreve onde o estudo foi realizado;
- Seção 6.3 – Seleção de um Projeto para Estudo de Caso: descreve como um projeto real foi selecionado, relacionando os aspectos envolvidos para esta seleção;
- Seção 6.4 – O Projeto Selecionado: descreve as características do projeto selecionado;
- Seção 6.5 – A Metodologia Utilizada: descreve a abordagem utilizada para realização deste estudo;
- Seção 6.6 – Cronograma: descreve o cronograma real do estudo realizado;
- Seção 6.7 – O Experimento: relata como se deu o estudo: dificuldades enfrentadas, aspectos do Guia XP-CMM2 implementados, aspectos não implementados.

Além disto, o Guia XP-CMM2 foi aplicado a uma unidade de negócios, ou seja, uma empresa incubada, visando concretização de um possível investimento. Esta aplicação também é descrita de forma resumida na Seção 6.8.

Finalmente, a Seção **Erro! Fonte de referência não encontrada.** apresenta algumas considerações sobre o estudo realizado.

6.1 Objetivos da Aplicação do Guia XP-CMM2

A aplicação do Guia XP-CMM2 a um projeto real, tem como principais objetivos:

- Verificar se o Guia é utilizável no mundo real;
- Verificar se os benefícios declarados do uso de XP podem ser alcançados com as modificações sugeridas pelo Guia;
- Verificar se é possível conviver em um mesmo ambiente com XP e com o nível 2 do CMM;
- Benefícios de aplicar a abordagem do nível 2 do CMM em um projeto XP;
- Verificar se um projeto XP perde a agilidade esperada se inserir práticas do CMM nível 2;
- Averiguar a dificuldade em assimilar práticas do nível 2 do CMM em um projeto com cultura XP.

6.2 O Ambiente

A aplicação do Guia XP-CMM2 foi realizada no C.E.S.A.R – Centro de Estudos e Sistemas Avançados do Recife. O C.E.S.A.R é uma organização sem fins lucrativos, associada à Universidade Federal de Pernambuco, que tem como missão a transferência auto sustentada de tecnologia entre a universidade e a sociedade. Para cumprir sua missão o C.E.S.A.R desenvolve projetos de software e cria empresas.

O C.E.S.A.R possui um processo padrão para desenvolvimento e manutenção de software, chamado ProSCes, que foi modificado para estar aderente ao nível 2 do CMM. Em Junho de 2003 o C.E.S.A.R foi reconhecido como nível 2 do CMM para uma área de projetos que desenvolve software para o mercado de telefonia celular.

6.3 Seleção de um Projeto para Estudo de Caso

Para iniciar a aplicação, um projeto deveria ser escolhido. Havia duas possibilidades para seleção do projeto:

- Abordagem 1: selecionar um projeto que seguisse o ProSCes, estando mais próximo do nível 2 do CMM, já que o ProSCes, conforme citado anteriormente, foi modificado para estar aderente ao nível 2;
- Abordagem 2: selecionar um projeto que utilizasse XP.

Não havia a possibilidade de selecionar um projeto do escopo da avaliação CMM, pois o risco de comprometer a avaliação seria alto para a organização, já que o Guia não havia sido utilizado por nenhum projeto e não tinha sido trabalhado pela consultoria especializada, como todos os outros projetos do escopo.

Era sabido que para qualquer escolha haveria uma forte resistência. Projetos acostumados com a cultura do nível 2 do CMM dificilmente gostariam de abrir mão dos seus processos para aderir a qualquer configuração de XP, conforme constatado por uma sondagem informal para a seleção de projetos para este estudo de casos. Por outro lado, os projetos que seguiam XP não aceitavam bem aspectos do CMM, com o receio de "burocratizar" o projeto.

Além da resistência por cultura adquirida de projeto – cultura CMM ou XP – notou-se uma aceitação diferente de XP de acordo com funções exercidas. Os gerentes de projeto muitas vezes encaram XP com receio pelo fato de a metodologia não abordar aspectos clássicos da gerência de projetos, como: riscos, cronograma convencional e planos. Provavelmente esta resistência é ainda maior no C.E.S.A.R, onde parte dos gerente são certificados PMP **[58]**. Arquitetos de software, por sua vez, valorizam a elaboração de uma arquitetura estável para o sistema, o que vai de encontro à prática "o mais simples possível" de XP. Desenvolvedores, em geral, aceitam bem XP devido à valorização de suas atividades.

O uso da abordagem 1 acarretaria, principalmente, em uma análise de impacto das metodologias ágeis, pois, neste caso, projetos já seriam acostumados com a cultura do nível 2 (apesar de nem todas as práticas do nível 2 serem implementadas em todos os projeto do C.E.S.A.R) e aspectos das metodologias ágeis seriam inseridos no decorrer dos

projetos. A introdução do Guia traria práticas de XP. A abordagem 2 favoreceria a análise de impacto do uso do CMM.

A abordagem selecionada foi a segunda: a de implantar o Guia num ambiente mais próximo de XP. Com isto, seria mais fácil averiguar se o CMM traria diminuição da agilidade no projeto. Além disto, esta abordagem utiliza a lógica empregada pelo Diagnóstico XP-CMM2, que partiu de XP para verificar aderência ao nível 2 do CMM.

6.4 O Projeto Selecionado

O projeto selecionado para aplicação do Guia XP-CMM2 foi firmado sobre um contrato de 36 meses, iniciado em junho de 2001, e envolve atualmente 19 pessoas. Ele engloba desenvolvimento de novas funcionalidades e manutenção, utilizando a linguagem de programação Java. O cliente é um órgão público do governo federal.

O projeto enfrenta algumas dificuldades devido a características próprias. Primeiramente, o projeto enfrenta a falta de um gestor do projeto no lado cliente, o que acarreta no não estabelecimento de um ponto focal, com responsabilidade e autoridade para tomar decisões em nome do cliente. O problema gerado por este fato é que as decisões não podem ser tomadas na velocidade necessária, mudanças são difíceis de serem negociadas e compromissos difíceis de serem estabelecidos. Esta situação foi agravada em 2003 devido à troca de presidentes, o que provocou mudanças significativas nas camadas gerenciais do cliente, que interagiam com o projeto.

Além disto, os usuários do sistema são representantes de diferentes áreas da empresa contratante, que possuem interesses divergentes. Estes usuários, responsáveis por definir requisitos e prioridades, e validar os produtos gerados pelo projeto, entram em conflitos que muitas vezes, comprometem o progresso do projeto.

Outro fator que torna o gerenciamento do projeto não trivial se deve ao fato de o projeto ser de desenvolvimento e manutenção ao mesmo tempo. Por um lado, os usuários cobram que todos os defeitos sejam corrigidos com a máxima urgência, por outro, o Patrocinador do projeto cobra que as novas funcionalidades sejam entregues nas datas acordadas.

Além disto, a equipe do projeto trabalha de forma distribuída: parte alocada no cliente, em Brasília e parte no C.E.S.A.R, em Recife. As equipes são comumente chamadas de equipe Brasília e equipe Recife.

A equipe Brasília tem a responsabilidade de elicitar e documentar os requisitos detalhados do cliente, os quais devem estar coerentes com os requisitos contratados, cronograma, metas e objetivos do projeto. Além disto, a equipe Brasília é responsável por registrar *bugs* e realizar testes de aceitação nos produtos que são implantados.

A equipe Recife trabalha sobre os requisitos repassados pela equipe Brasília. XP foi empregado inicialmente na equipe Recife, que tratava a equipe Brasília como o cliente.

A decisão de utilizar XP no projeto se deu objetivando aumentar a satisfação do cliente, a produtividade, comprometimento e motivação da equipe, aumentar a eficácia das estimativas, além de diminuir os grandes riscos do projeto e facilitar o planejamento. Os aspectos de XP que mais contribuíram para apoiar a decisão de utilizá-lo visando o alcance destes objetivos foram os *releases* curtos de software de valor, as iterações curtas de tamanho fixo, as reuniões diárias e as estimativas realizadas pelos próprios desenvolvedores.

6.5 A Metodologia Utilizada

A metodologia empregada para a realização dos resultados foi o de executar o papel de garantia da qualidade (SQA) no projeto, conforme definido pela KPA *Software Quality Assurance* do CMM.

Segundo o CMM, o objetivo da KPA SQA é "prover o gerenciamento com visibilidade apropriada no processo que está sendo usado pelo projeto de software e o produto que está sendo construído"[58]. O grupo de SQA cumpre este objetivo através da realização de revisões e auditorias.

Uma característica importante do SQA é que ele deve ser independente do projeto, como sugere o CMM na prática SQA.Co1.2 "O grupo de SQA tem um canal de reportagem para a gerência sênior que é independente de: gerente do projeto, grupo de engenharia de software do projeto e outros grupos relacionados a software" [58].

No C.E.S.A.R, o grupo de garantia da qualidade responde à gerência de qualidade, que responde à gerência de tecnologia. A gerência de tecnologia possui o mesmo nível da gerência de consultoria, que gerencia os projetos. Desta forma, o grupo é completamente independente do gerente de projeto e de outros grupos relacionados a software. A Figura 6-1 exibe o organograma do C.E.S.A.R no contexto de garantia da qualidade e projetos.

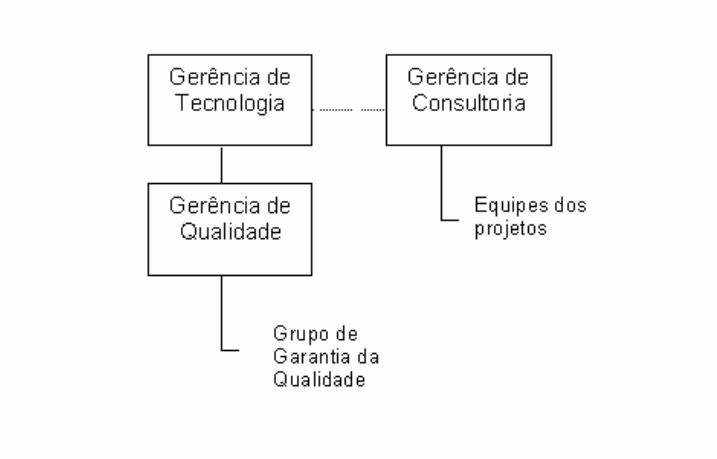

Figura 6-1 - Organograma no contexto do grupo de SQA

A independência de SQA garante que auditorias e revisões poderão ser realizadas e os desvios encontrados cobrados de acordo com as prioridades da organização quanto à qualidade de software. Assim, um desvio encontrado não poderá ser preterido pelo gerente do projeto devido a problemas de estouro de prazos, por exemplo.

Dificuldades não foram encontradas para desempenhar o papel do grupo de qualidade em um projeto visando à aplicação do Guia XP-CMM2. A autora faz parte do grupo de garantia da qualidade do C.E.S.A.R, sendo líder do grupo e, como tal, tendo as seguintes atribuições: definição e melhoria do processo de garantia da qualidade, alocação do grupo nos projetos, acompanhamento periódico do grupo.

O processo seguido para execução da garantia da qualidade no projeto foi o Procedimento de Garantia da Qualidade do ProSCes – o processo padrão do C.E.S.A.R. Para melhor entender o trabalho conduzido e o contexto do projeto é importante descrever o ProSCes e o Procedimento de Garantia da Qualidade do ProSCes, descritos nas próximas seções.

6.5.1 O ProSCes

O ProSCes foi elaborado em 2000, com base no RUP e, com isto, possui suas principais características: modelo de ciclo de vida iterativo e incremental, centrado em casos de uso, uso de UML como linguagem de modelagem, orientação a objetos, arquitetura baseada em componentes, gerenciamento de requisitos, entre outros [35].

O ciclo de vida do RUP é composto de quatro fases realizadas em zero ou mais iterações, sendo cada iteração realizada por zero ou mais disciplinas [35]. A Figura 6-2 exibe o ciclo de vida do RUP.

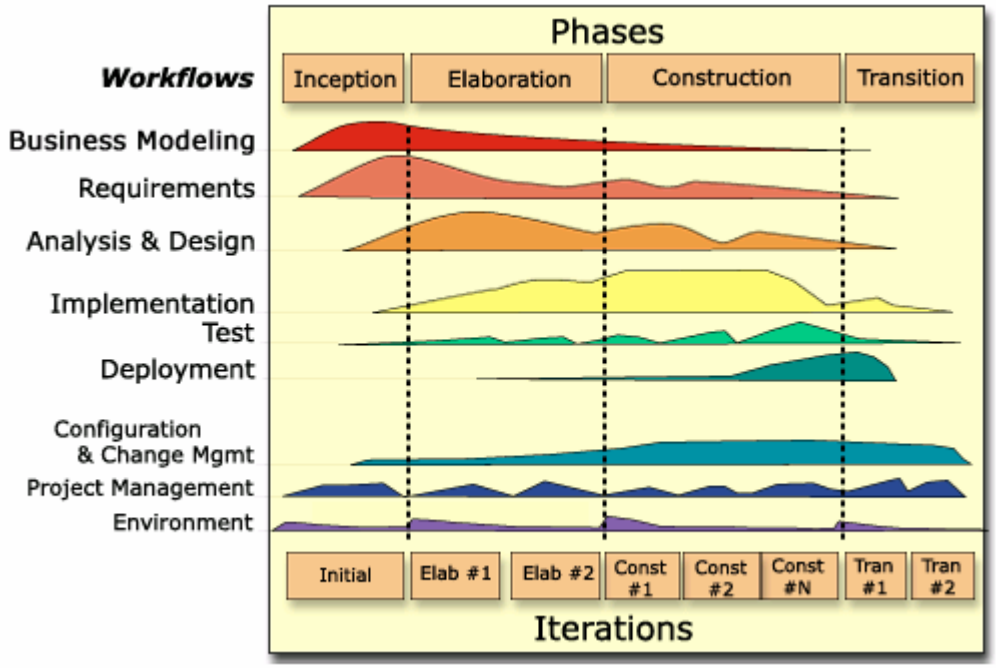

Figura 6-2 – Ciclo de vida do RUP

Para cada disciplina do RUP, o ProSCes definiu procedimentos, guias, ferramentas e *templates* para apoiar a equipe de desenvolvimento. Esta definição foi elaborada a partir de customizações sobre os artefatos e procedimentos do RUP. Além das disciplinas do RUP, o ProSCes adicionou duas disciplinas ao ciclo de vida:

- Fornecimento: determina processos para venda de serviço e acompanhamento de contrato;
- Garantia da qualidade de software: determina processos para realização das atividades de garantia da qualidade nos projetos.

O ProSCes define o "ProSCes Core" que é o processo mínimo necessário, a ser seguido por todos os projetos. Este "ProSCes Core" está completamente aderente ao nível 2. Assim, todos os artefatos obrigatórios do ProSCes, que implementam alguma prática do nível 2 fazem parte do "ProSCes Core". Por outro lado, a grande maioria dos artefatos que não implementam o nível 2, como por exemplo, modelo de classes ou projeto de testes, não fazem parte do "ProSCes Core".

Ainda assim, um projeto poderá não utilizar o "ProSCes Core" caso deva utilizar o processo do cliente por questões contratuais. Nestes casos, uma análise deve ser feita pelo gerente do projeto em conjunto com o grupo de garantia da qualidade, no início do projeto, para identificar se o processo do cliente está alinhado ao nível 2 do CMM e à política de desenvolvimento do C.E.S.A.R.

O "ProSCes Core" é genérico e ainda passível de adaptação, desde que aprovada pelo grupo de garantia da qualidade no projeto. Assim, um *template* definido pelo "ProSCes Core" poderá sofrer ajustes para atender às necessidades do projeto, desde que atenda ao nível 2 do CMM, à política de desenvolvimento do C.E.S.A.R e às restrições do cliente.

6.5.2 O Procedimento de Garantia da Qualidade do ProSCes

Conforme política de desenvolvimento publicada, o Procedimento de Garantia da Qualidade deveria ser seguido pelo grupo de garantia da qualidade, em todos os projetos do C.E.S.A.R. Este procedimento foi utilizado como base para a aplicação do Guia XP-CMM2. As atividades do procedimento de garantia da qualidade do ProSCes são descritas resumidamente abaixo.

- **Planejar garantia da qualidade do projeto**

O planejamento de garantia da qualidade deve ser realizado em conjunto com o gerente do projeto, no início do projeto. Esta atividade gera um Plano de Garantia da Qualidade que, no ProSCes, é inserido dentro do Plano do Projeto. Informações contidas neste plano são: processos e padrões adotados para o projeto, adequações (*tailors*) aplicados ao processo, planejamento das auditorias, métricas selecionadas, planejamento de revisões.

- **Realizar reunião de início do projeto**

No início do projeto, o grupo de garantia da qualidade é responsável por orientar toda a equipe sobre papel, responsabilidades, autoridade e valor do grupo. Isto é feito através de uma reunião chamada *Kickoff*. O ProSCes dispõe de uma apresentação padrão contendo as informações necessárias, dentre elas, é importante ressaltar: a independência do grupo de SQA, objetivos de auditoria, papel da equipe em relação aos desvios encontrados.

- **Realizar auditoria de processo**

As auditorias de processo devem ser realizadas periodicamente, conforme planejado na seção relacionada à garantia da qualidade do Plano do Projeto. As auditorias de processo têm como objetivo:

- Verificar conformidade e efetividade do processo de desenvolvimento utilizado pela equipe aos padrões e procedimentos estabelecidos;
- Reportar problemas e oportunidades de melhoria do processo utilizado pela equipe para os grupos interessados.

O ProSCes dispõe de um *checklist* padrão para a realização das auditorias de processo, contemplando o que típica e minimamente deve ser checado em auditorias. Este *checklist* deve ser acrescido de padrões e procedimentos específicos do projeto a cada auditoria.

As auditorias de processo são realizadas através de entrevistas e análise de documentos. As entrevistas podem ser realizadas em grupos ou individualmente, desde que gerentes e subordinados não sejam entrevistados ao mesmo tempo, para garantir conforto e confidencialidade aos entrevistados. O grupo de garantia da qualidade deve ter total liberdade para analisar os arquivos do projeto.

Ao final da auditoria, um relatório consolidando as informações obtidas deve ser produzido pelo grupo de garantia da qualidade e enviado à alta gerência e equipe do projeto.

Para cada desvio encontrado na auditoria, ações para correção deverão ser identificadas e prazos e responsáveis por executá-las, devem ser estabelecidos pelo grupo de garantia da qualidade em conjunto com o gerente do projeto.

- **Realização de auditorias de produto**

O objetivo desta auditoria é verificar a conformidade dos produtos (artefatos) gerados pelo projeto aos padrões estabelecidos. Para isto, o grupo de garantia da qualidade deverá selecionar os produtos a serem auditados. Auditorias de produto devem ser realizadas de forma amostral, segundo um critério documentado no plano de garantia da qualidade. As auditorias de produto ocorrem durante as auditorias de processo e à medida que os artefatos vão sendo construídos.

Em geral, os problemas encontrados nas auditorias de produto são de menor criticidade que os problemas encontrados nas auditorias de processo. Assim, não há um relatório de consolidação dos resultados. Os problemas identificados, assim como na auditoria de processo, deverão gerar ações para correção com prazos e responsáveis por executá-las, acordados em conjunto com o gerente do projeto. Os problemas devem ser comunicados à equipe através dos relatórios periódicos emitidos pelo grupo de garantia da qualidade.

- **Acompanhamento dos desvios encontrados**

O grupo de garantia da qualidade deve garantir que os desvios encontrados nas auditorias serão corrigidos. Isto é feito através da verificação, junto ao responsável, da realização das ações planejadas para correções dos desvios nas datas acordadas.

Se a equipe não realizar as ações acordadas, o grupo de garantia da qualidade poderá escalar o problema para a alta gerência de qualidade, que deverá intervir junto à alta gerência dos projetos. Problemas escalados deverão ser acompanhados pelo grupo de garantia da qualidade até resolução.

- **Apoiar a equipe no uso do processo**

O grupo de garantia da qualidade deve apoiar a equipe no uso do processo através de esclarecimentos de dúvidas, identificação de necessidades de treinamentos, consultoria de processo visando garantir que a equipe utilize o processo adotado de forma apropriada.

- **Reportagem dos resultados**

O grupo de garantia da qualidade deve reportar à equipe do projeto, ao grupo de garantia da qualidade do cliente e à gerência sênior do C.E.S.A.R o *status* das atividades de garantia de qualidade de software no projeto. Para isto deve elaborar um relatório mensal, seguindo um *template* disponível no ProSCes e o revisa com os interessados em reuniões periódicas.

- **Receber auditoria de garantia de qualidade**

O grupo de garantia da qualidade deve disponibilizar materiais e informações necessárias para realização efetiva de auditorias externas ao grupo. O objetivo destas auditorias é garantir que as atividades de garantia da qualidade nos projetos do C.E.S.A.R estão sendo realizadas conforme o procedimento do ProSCes.

Assim como nas auditorias de processo e produto, os desvios identificados serão reportados e prazo e responsáveis pela solução do problema identificados em conjunto com a gerência de qualidade, que é responsável por acompanhá-los.

- **Realizar fechamento de garantia da qualidade**

Ao final do projeto, o grupo de garantia da qualidade é responsável por consolidar as informações relacionadas à garantia da qualidade do projeto e comunicá-las à equipe e ao grupo de garantia da qualidade do cliente.

6.5.3 Resultados Esperados

Através da execução do papel de garantia da qualidade, a análise da aplicação do Guia XP-CMM2 foi realizada de forma sistemática e independente. O trabalho de garantia da qualidade – em especial as auditorias – trouxe dois tipos de resultado, como previsto:

- Resultado concreto: baseado em informações concretas a respeito da adequação à metodologia proposta para o projeto (XP aplicado ao Guia). Estes resultados são capturados, por exemplo, quando um documento não é produzido ou não é atualizado ou quando atividades do processo não são cumpridas;
- Resultado abstrato: sentimentos da equipe quanto ao processo e produto criado. Este tipo de resultado é capturado tipicamente em entrevistas de auditoria de processo. Nestas entrevistas, as pessoas costumam emitir gestos, opiniões e extravasar sentimentos, motivados justamente pela independência de garantia da qualidade em relação ao gerente do entrevistado.

Um resultado abstrato deve ser utilizado para investigação e identificação de resultados concretos. Como exemplo, se um entrevistado relatar que o gerente do projeto não acompanha o cronograma, o grupo de garantia da qualidade deve buscar evidências deste não acompanhamento – como um cronograma desatualizado.

No entanto, algumas vezes, um resultado abstrato pode não levar à identificação de um resultado concreto. Um exemplo é quando uma pessoa passa impressões como de o processo ser burocrático ou testes não serem efetivos. Mesmo nestes casos, este tipo de resultado é importante para análise do uso do processo.

Estes dois tipos de resultados serão relatados neste trabalho sempre que contribuir para o alcance dos objetivos da aplicação do Guia.

6.6 Cronograma

Um engenheiro de qualidade, representante do grupo de garantia da qualidade, foi alocado ao projeto em novembro de 2002. Durante o mês de novembro foi envolvido nas atividades de planejamento da garantia da qualidade. Em dezembro de 2002 as atividades passaram da fase de planejamento para execução, quando as auditorias passaram a ser realizadas e a resolução dos desvios identificados acompanhados.

Como o projeto já estava em andamento quando o engenheiro de qualidade foi alocado ao projeto, o planejamento da garantia da qualidade não foi iniciado no início do projeto, conforme Procedimento de Garantia da Qualidade do ProSCes. As principais dificuldades enfrentadas pelo grupo de garantia da qualidade a partir da alocação ao projeto foram:

- No início do projeto não houve planejamento para implementação de garantia da qualidade e tão pouco aderência aos processos do nível 2 do CMM. Em junho de 2001, quando o projeto iniciou, o ProSCes já existia, mas não possuía ainda todos os elementos e práticas do nível 2;
- O projeto já estava em execução há um ano e cinco meses. Já possuía uma cultura assimilada e equipe organizada. Dificuldades foram encontradas para definir um novo processo e cobrar desvios de processos identificados em auditorias, após todo este tempo sem atuação de garantia da qualidade.

As auditorias de processo foram planejadas para serem realizadas bimestralmente. Cada auditoria foi focada em uma disciplina do desenvolvimento de projetos, sendo tipicamente em disciplinas do nível 2 do CMM: gerência de requisitos, planejamento e acompanhamento de projetos e gerência de configuração. Não foram realizadas auditorias de SSM, pois o projeto não subcontratava. A cada auditoria, os produtos associados

também eram auditados. A Tabela 6-1 relaciona as auditorias realizadas e o foco de cada uma.

Mês	Foco
Dezembro/2002	Gestão de projetos, KPAs SPP, SPTO e RM
Fevereiro/2003	Gerência de Configuração, KPA SCM
Maio/2003	Gestão de projetos, KPAs SPP, SPTO e RM

Tabela 6-1 – Auditorias realizadas

A auditoria de Abril foi postergada para Maio devido a problemas internos do projeto, ocorridos após troca de gerentes.

As auditorias tomavam como base:

- ProSCes Core;
- Nível 2 do CMM;
- Diagnóstico XP-CMM2.

O Diagnóstico XP-CMM2 foi utilizado para apoiar na interpretação da satisfação das práticas do projeto em relação ao "ProSCes Core". Um exemplo desta interpretação, é que o "ProSCes Core" define que um cronograma deve ser elaborado e acompanhado periodicamente. O diagnóstico aponta se há cronograma e se este é acompanhado em XP. De acordo com a interpretação do diagnóstico as evidências de institucionalização seriam procuradas na equipe.

Para cada desvio encontrado, uma recomendação, um responsável e data para implementá-la foram definidos. As recomendações seriam baseadas no Guia XP-CMM2. Desta forma, o Guia seria introduzido aos poucos no projeto, conforme capacidade de assimilação da equipe e disponibilidade de treinamento. Os templates sugeridos pelo Guia XP-CMM2 não foram adotados pelo projeto, já que ele deveria seguir os templates do C.E.S.A.R.

Em julho de 2003 o estudo foi considerado encerrado visando consolidação dos dados obtidos. No entanto, a autora continua atuando como garantia da qualidade no projeto.

6.7 O Experimento

Esta seção descreverá como foi realizado o estudo de caso e os resultados obtidos. Para isto, abordará inicialmente como a equipe utilizava XP inicialmente, sem a interferência do Guia XP-CMM2.

Depois apresentará a fase de planejamento da garantia da qualidade no projeto e então a fase de execução da garantia da qualidade, descrevendo os desvios encontrados nas auditorias e as ações tomadas para correção.

Finalmente, é realizada uma breve análise de cada KPA em relação ao que foi implementado do Guia XP-CMM2.

6.7.1 Uso de XP no Projeto

Esta seção descreverá como XP era usado no projeto antes de qualquer interferência do Guia XP-CMM2. Nem todas as práticas de XP foram implementadas à risca. Além disto, algumas customizações, como a adoção de ferramentas, foram realizadas.

6.7.1.1 Práticas Implementadas

Dentre as práticas de XP adotadas pelo projeto, estão:

- Jogo do planejamento – O planejamento detalhado do projeto segue esta prática de XP: há um planejamento macro dos *releases*, os quais são detalhados em termos de estórias antes de iniciar. Cada *release* é composto de iterações de duas semanas, as quais também são detalhadas apenas antes de iniciar, em termos de tarefas para realizar as estórias. Estórias e tarefas são selecionadas para os *releases* e iterações de acordo com a prioridade do cliente e o esforço necessário para realizá-las é estimado pela equipe de desenvolvimento e registradas na ferramenta XPlanner **[51]**. A Figura 6-3 exibe o planejamento de uma das iterações do projeto no XPlanner;


XPlanner

Top | Project

Iteration: Iteration 11 (2003-05-27 to 2003-06-09)

Nesta Iteracao:

- Desenvolvedores : **Carol**, **Jeca**, **Katharine**, **Lane**, **Malu**, **Marcos**, **Nelson**, **Nelson**, **Renne**, **Tarcisio**
- Gerencia de Configuracao: **Jeca**
- Equipe SWAT oficial: **Tarcisio**
- Tracker : **Fabio**, **Mardonio**

Hours: Estimated 359.0, Actual 350.6

User Story	!	Progress	Est.
02.0 CR1339: Alterar consultas por demanda para o uso da reuniao	4		8.0
06.0 CR1257: Solicitar Reclasificacao de Bolsa	4		21.0
14.0 CR1128:CR1129:CR1130:CR1131:Data impl, inicio vigencia e termino vigencia	4		16.0
15.0 CR782: Remanejar Solicitacao	4		20.0
FORM.KYLIX:CR1343: Imprimir Proposta	4		58.0
FORM.KYLIX:CR1345: Compatibilizacao de CRs	4		5.0
FORM.KYLIX:CR1346: Grafico de Atividades	4		2.0
FORM.KYLIX:CR1347: Guiche Client (Modulo de Comunicacao)	4		13.0
SWAT	4		216.0

Figura 6-3 - O jogo do planejamento no XPlanner

- *Releases* pequenos – A entrega do software foi planejada para ocorrer incrementalmente, sendo cada pedaço do sistema entregue em um prazo médio de um mês e meio;
- *Refactoring* – Mesmo tendo uma arquitetura estabilizada para todo o projeto, ao receberem tarefas de desenvolvimento, os desenvolvedores analisam se é possível melhorar a forma como o software foi implementado. Recentemente uma grande reestruturação na arquitetura do software foi realizada para aumentar a performance do sistema;
- Padrão de codificação – O projeto segue o padrão de codificação definido pelo C.E.S.A.R para desenvolvimento em Java;
- Propriedade coletiva – Todos da equipe podem alterar qualquer arquivo que compõe o software;
- 40 horas por semana – O C.E.S.A.R possui um banco de horas para que as horas trabalhadas a mais possam ser compensadas com descanso, conforme acordado com o gerente do projeto e política da organização. Mesmo com o banco de horas, a equipe trabalha usualmente 40 horas por semana;
- Programação em pares – Esta prática não é adotada como sugere XP: toda a equipe trabalhar todo o tempo em pares. No projeto, a programação em pares é

adotada para novatos e pessoas pouco experientes na tecnologia adotada, os quais devem trabalhar com um membro mais experiente da equipe até que estejam aptos a realizar suas atividades plenamente. Atualmente, a possibilidade de estender esta prática para outras situações está sendo avaliada;

- Integração contínua – Diariamente o *build* do software é gerado automaticamente pela ferramenta *Ant* [50], adotada pelo projeto. Quando acionada, a ferramenta gera o *buid* do sistema, um arquivo informando os arquivos alterados e as requisições de mudança (CRs) fechadas na ferramenta de controle de mudança do projeto (o Bugzilla [46]).

6.7.1.2 Práticas Não Implementadas

As práticas de XP não adotadas são:

- Testes automáticos – Apesar de acreditar nos benefícios dos testes automáticos, esta prática ainda não foi adotada no projeto. Esta decisão se deu pelo fato de a adoção de XP ter ocorrido quando grande parte do código já estava implementada. Para obter todos os benefícios dos testes automáticos, todo o código deveria possuir testes, inclusive este código anteriormente produzido. Implementar todos estes testes traria um *overhead* muito grande para a equipe;
- Cliente no local – O cliente não se localiza no mesmo estado em que o projeto foi desenvolvido. Para representar o cliente dentro do projeto e para que o trabalho de entendimento dos requisitos e validação dos artefatos gerados fosse efetivo, parte da equipe passou a trabalhar no local do cliente. Esta equipe é responsável por especificar os requisitos, validá-los e conduzir testes de aceitação;
- Metáforas – O projeto utiliza o paradigma de programação orientado a objetos, o que torna o projeto de software perto da realidade do cliente. Devido a isto, o uso de metáforas não foi adotado no projeto;
- Projeto simples – Pelo fato de a equipe e o software a ser construído serem relativamente grandes, optou-se, no início do projeto, por definir uma arquitetura estável, que facilitasse a inserção de novas funcionalidades, manutenção, componentização e independência entre as regras de negócio e dados.

6.7.2 O Planejamento de Garantia da Qualidade no Projeto

Por já estar em andamento, o projeto já possuía uma cultura de desenvolvimento assimilada e um processo adotado. O projeto seguia algumas práticas de XP, mas, contrariando a política de desenvolvimento do C.E.S.A.R, não seguia à risca nem mesmo o "ProSCes Core". Esta foi uma dificuldade inicial enfrentada neste estudo de caso: projeto já iniciado sem garantia da qualidade, sem seleção do processo adequado para o projeto, sem que o processo colocado em prática estivesse documentado e ainda sem o seguimento do "ProSCes Core".

Durante o planejamento da qualidade, realizado em novembro, optou-se por documentar o processo a ser seguido em um plano do projeto, já inserindo alguns aspectos do "ProSCes Core" e do Guia XP-CMM2, como a elaboração do próprio plano do projeto e garantia da qualidade, plano de gerência de configuração, entre outros.

Este plano foi revisado pela alta gerência do C.E.S.A.R e representantes do projeto. O cliente seria convidado a revisar e aprovar o plano em um segundo momento, quando todos concordassem sobre o conteúdo do plano. No entanto, devido a problemas internos do cliente, o plano não pôde ser submetido à aprovação. Mesmo assim, foi considerado como base para acompanhamento e auditorias de garantia da qualidade.

6.7.3 As Auditorias Realizadas

Esta seção irá descrever os principais problemas encontrados nas auditorias realizadas no projeto e respectivas recomendações sugeridas pelo grupo de garantia da qualidade. Conforme já citado, a identificação dos problemas foi apoiada pelo Diagnóstico XP-CMM2, que realizou a interpretação de XP em relação ao CMM nível 2. Além destes aspectos de interpretação de processos, as auditorias checavam institucionalização, ou seja, se os processos definidos estavam sendo seguidos pela equipe.

As recomendações sugeridas para resolução dos problemas identificados tomaram como base o Guia XP-CMM2, com pequenas alterações visando contemplar as especificidades do projeto. Vale ressaltar que uma mesma recomendação pode resolver a vários desvios.

Os problemas descritos nesta seção são um subconjunto dos problemas identificados nas auditorias, caracterizado pelo escopo deste trabalho. Assim, problemas

identificados relacionados à instituição ou uso de ferramentas específicas – não relacionadas com XP ou com o nível 2 do CMM – não foram citados.

O resultado das auditorias relatado nas próximas seções poderá indicar um mesmo problema em mais de uma auditoria. Isto significa que ou o problema não foi resolvido entre uma auditoria e outra ou foi resolvido, mas há uma reincidência. Este resultado é esperado por qualquer projeto de software. Por isto é tão importante o trabalho de garantia da qualidade, para que problemas não sejam apenas identificados, mas também, acompanhados até fechamento.

6.7.3.1 Dezembro de 2002

Esta auditoria foi realizada com foco em planejamento e acompanhamento. Entrevistas foram realizadas com integrantes da equipe do projeto e a versão *draft* do Plano do Projeto foi analisada. A Tabela 6-2 relaciona os principais desvios e recomendações identificados relacionados ao escopo deste trabalho.

Prática	Desvio	Recomendação
RM.Ac2	O documento de requisitos, estabelecido como artefato do projeto no *draft* do Plano do Projeto, não foi elaborado e nem documentado. Desta forma, o *baseline* de requisitos não está claro para o projeto.	Elaborar e aprovar o documento de requisitos do projeto.
RM.Ac3, SPP.Ac4	O projeto não possui um Plano de Projeto aprovado pelos responsáveis.	Aprovar Plano do Projeto inserindo planejamento de recursos críticos do projeto, revisões formais, reuniões periódicas com toda a equipe do projeto.
SPP.Ac13	O Plano de Riscos não foi elaborado para o projeto.	Elaborar e acompanhar Plano de Riscos do projeto.
SPP.Ac11	Recursos críticos não foram planejados e estimados para o projeto.	Aprovar Plano do Projeto inserindo planejamento de recursos críticos do projeto, revisões formais, reuniões periódicas com toda a equipe do projeto.
SPTO.Ac13	Revisões formais não foram planejadas para o projeto.	Aprovar Plano do Projeto inserindo planejamento de recursos críticos do

		projeto, revisões formais, reuniões periódicas com toda a equipe do projeto.
SCM.Ac4	Não está estabelecido SCCB para itens de configuração que não é código.	Estabelecer SCCB para itens de configuração que não é código.
SCM.Ac10	Auditorias nas *baselines* do projeto não estão sendo realizadas.	Realizar periodicamente auditorias nas *baselines* do projeto.

Tabela 6-2 - Desvios e Recomendações da Auditoria de Dezembro de 2002

Durante o planejamento da garantia da qualidade, iniciado quando o grupo de garantia da qualidade foi alocado ao projeto, foi elaborado um plano do projeto, que deveria ter sido elaborado no início do projeto, conforme o ProSCes. Este plano passou a documentar requisitos não técnicos do projeto, como marcos, prazos, a requisição de prestação de contas para o cliente, entre outros. No entanto, este plano não foi aprovado pelo cliente até o momento desta auditoria. Sem a aprovação do plano, os compromissos e requisitos não técnicos do projeto não possuem concordância das partes envolvidas.

Neste momento, não havia sido elaborado um documento que descrevesse os requisitos técnicos do projeto. Os requisitos técnicos estavam documentados apenas na proposta técnica, ou contrato do projeto. O problema, é que este documento é estático e não é atualizado com mudanças realizadas no projeto. Este documento sofre alterações, apenas em alterações muito críticas nos compromissos acordados, como no escopo, prazo e custos.

Como parte da equipe se encontrava dentro do cliente para especificar os resultados, o trabalho realizado pela equipe se concentrava em atender demandas especificadas pelos usuários do sistema, de acordo com suas prioridades. No entanto, estas prioridades poderiam não estar contratadas, ou seja, não fazer parte do escopo do projeto.

Este problema impactava diretamente a KPA de gerência de requisitos, pois já que não existia uma *baseline* de requisitos, as mudanças também não poderiam ser controladas (desvio na atividade RM.Ac2, não formalizado na auditoria, pois a atividade RM.Ac1 precisaria ser tratada primeiro).

Vele considerar, que a forma como XP trata os requisitos era utilizada como base pela equipe: trabalhar sobre prioridade do cliente. No entanto, o contrato estabelecido pregava que um escopo determinado deveria ser cumprido até o final do projeto.

A auditoria recomendou que um Documento de Requisitos fosse elaborado, aprovado pelo cliente e gerenciado. Assim, a equipe deveria implementar os requisitos descritos no Documento de Requisitos, detalhando-os, conforme sugerido por XP.

Riscos não haviam sido identificados. Recursos críticos não haviam sido identificados e estimados.

Revisões formais não haviam sido planejadas: à medida que os produtos ficavam prontos, estes eram colocados em produção, sem revisão formal por parte do cliente. É importante ressaltar, que esta prática foi considerada um ponto fraco na auditoria por não estar aderente ao CMM, mas a equipe também não estava seguindo XP, que determina o envolvimento do cliente e testes de aceitação.

Inicialmente o projeto possuía muitos problemas de gerência de configuração, como: não saber a versão que se encontrava instalada no cliente, não saber o que constava em cada versão, retrabalho para solução de problemas já resolvidos, entre outros. Um gerente de configuração foi alocado *part time* ao projeto.

Neste momento, um plano de gerência de configuração acabava de ser elaborado e aprovado por representantes do projeto. A princípio, o plano considerava apenas como item de configuração o código fonte e o *build* do sistema. Auditorias nas *baselines* não eram realizadas.

6.7.3.2 Fevereiro de 2003

Esta auditoria foi realizada com foco em gerência de configuração. Entrevistas foram realizadas com integrantes da equipe do projeto e foram analisados em detalhes o plano de gerência de configuração, análise do repositório e da ferramenta de controle de mudanças do projeto. A Tabela 6-3 relaciona os principais desvios e recomendações identificados relacionados ao escopo deste trabalho.

Prática	Desvio	Recomendação
RM.Ac2	O baseline de requisitos não está definido para o projeto: não há um documento de requisitos do projeto,	Definir o *baseline* de requisitos. Ligá-lo às especificações passadas à equipe de desenvolvimento.

	mudanças aos requisitos não são controladas, a especificação das funcionalidades do projeto não estão relacionadas com os requisitos da proposta técnica aceita pelo cliente.	
SPTO.Ac10	Os riscos do projeto estão sendo reportados à gerência sênior do C.E.S.A.R, no entanto, não estão sendo acompanhados: a planilha de riscos não é alterada desde o momento em que foi colocada no repositório do projeto.	Acompanhar periodicamente os riscos do projeto.
SCM.Ac2	O plano de gerência de configuração tem sido alterado constantemente, no entanto, estas mudanças não estão sendo submetidas à revisão.	Elaborar critério para submissão dos planos à revisão e submeter o plano de gerência de configuração à revisão.
SCM.Ac2	O plano de gerência de configuração não está atualizado (há definições que estão em desuso no projeto) e muitos dos padrões estabelecidos não estão sendo seguidos pela equipe.	Atualizar o plano de gerência de configuração com as práticas e padrões adotados pelo projeto e permitidos pelo ProSCes.
SCM.Ac10	Uma auditoria foi realizada pelo grupo de gerência de configuração, no entanto, esta auditoria não foi eficaz, pois não identificou problemas como: nomeação incorreta dos itens de configuração e das *baselines*, falta de estabelecimento de *baselines*, falta de controle de mudanças de itens de configuração, não seguimento do padrão estabelecido para a estrutura do repositório e ferramenta de controle de mudanças.	Elaborar *checklist* para auditoria nas *baselines* do projeto.
SCM.Ac10	A auditoria realizada não teve os problemas identificados acompanhados	Registrar os desvios de auditoria encontrados e acompanhá-los até

	pelo gerente de configuração.	fechamento.
SCM.Ac4	Nem todos os artefatos importantes para o desenvolvimento do projeto foram identificados como itens de configuração, dentre eles: manual do usuário, documento de requisitos, especificação de casos de uso. Além disto, Itens de Configuração selecionados no PGC não estão no repositório do projeto.	Rever os itens de configuração e as baselines do projeto.

Tabela 6-3 - Desvios e Recomendações da Auditoria de Fevereiro de 2003

Neste momento, um sistema para gerenciar o escopo do projeto ainda não havia sido implantado. Uma versão inicial do Documento de Requisitos, proposta na auditoria de Dezembro de 2002 foi elaborada, no entanto, ainda não havia sido finalizada, aprovada e acompanhada em relação ao que se estava desenvolvendo no projeto.

Vale ressaltar que todo o trabalho realizado pela equipe Recife passava por um fluxo bem definido: a equipe Brasília registrava uma solicitação de mudança na ferramenta de controle de mudança para um *bug*, novo requisito ou alteração de um requisito já solicitado. A equipe Recife avaliava esta solicitação na reunião de planejamento da iteração, estimando-a e atribuindo-a à equipe.

O problema estava em gerenciar se as solicitações de mudança solicitadas e implementadas estavam realizando o escopo contratado do projeto. A equipe trabalhava sobre as prioridades do cliente, no entanto, conforme mencionado anteriormente, o projeto possuía um contrato firmado, que especificava diversas funcionalidades, e que deveria ser cumprido até o final do projeto.

Riscos foram identificados, mas não estavam sendo acompanhados.

O planejamento da gerência de configuração no projeto foi realizado pensando-se apenas no contexto da fábrica de software e não do projeto como um todo. A opção de não controlar os produtos elaborados pela parte do projeto que trabalha em Brasília foi acordada em reunião com o gerente do projeto e grupo de garantia da qualidade, visando facilitar a institucionalização do processo de gerência de configuração. Devido a isto,

poucos itens de configuração foram identificados no plano de gerência de configuração. Apenas os produtos relacionados ao *build* (código fonte, arquivos de interface e outros) estavam sendo gerenciados e controlados.

Além disto, apesar de bastante alterado, o plano de gerência de configuração possuía práticas não assimiladas pela equipe e não possuía um critério para ser submetido à nova aprovação. Assim, novos controles relacionados à gerência de configuração foram documentados e colocados em prática no projeto sem a devida aprovação do gerente do projeto e líderes.

Neste momento já foi possível identificar prejuízos pela não identificação de itens de configuração dos produtos criados pela equipe de Brasília. Assim, a auditoria sugeriu que os itens de configuração do projeto fossem replanejados.

As auditorias nas *baselines* realizadas pelo grupo de gerência de configuração não estavam sendo eficazes: poucos problemas foram identificados e estes não foram acompanhados.

6.7.3.3 Maio de 2003

A auditoria realizada teve como objetivo avaliar o entendimento da equipe e uso dos processos adotados pelo projeto, especialmente, o processo de planejamento e acompanhamento de projetos.

A auditoria se deu através de entrevistas e análise dos produtos de software gerados pela equipe. Três seções independentes de entrevistas foram realizadas, cada uma contando com a participação de integrantes da equipe. A primeira seção foi realizada com o gerente do projeto, a segunda com líderes das equipes de Brasília e Recife e a terceira com engenheiros de software e analistas de negócio.

Os produtos analisados nesta auditoria foram: *draft* do Plano do Projeto, XPlanner Desenvolvimento e Repositório do projeto. A Tabela 6-4 relaciona os principais desvios e recomendações identificados relacionados ao escopo deste trabalho.

Prática	Desvio	Recomendação
RM.Ac2	O documento que descreve os requisitos do projeto e define o escopo não está aprovado.	Aprovar o documento de requisitos.
RM.Ac3	Atualmente não há um processo para	Documentar o processo para alteração

	registrar, avaliar e aprovar as mudanças aos requisitos do projeto.	de requisitos no Plano do Projeto, seção de garantia da qualidade.
RM.Ac2, SPP.Ac4	O projeto não possui um plano do projeto aprovado e atualizado. Os documentos de planejamento existentes não estão na ferramenta de controle de versão do projeto.	Elaborar o plano do projeto e colocá-lo sobre gerência de configuração. O plano deve conter: • Métricas a serem coletadas; • Processo de desenvolvimento do projeto; • Escopo do projeto; • Processo de alteração de requisitos; • Cronograma do projeto; • Riscos do projeto; • Revisões formais em marcos; • Acompanhamento periódico da equipe; Aprovar o plano do projeto.
SPP.Ac13, SPTO.Ac10	Riscos não foram identificados pela nova gerência do projeto. Os riscos já levantados não estão sendo acompanhados.	Identificar novos riscos e acompanhá-los.
SCM.Ac2	O plano de gerência de configuração possui várias alterações ainda não aprovadas pelo atual gerente do projeto.	Aprovar o plano de gerência de configuração, contemplando o planejamento do controle da configuração dos produtos gerados por Brasília.
SCM.Ac4	Não há controle de configuração dos produtos gerados pela equipe de Brasília.	Aprovar o plano de gerência de configuração, contemplando o planejamento do controle da configuração dos produtos gerados por Brasília.

Tabela 6-4 - Desvios e Recomendações da Auditoria de Maio de 2003

No momento desta auditoria, o projeto estava em fase de replanejamento. O gerente do projeto foi substituído e o atual passou a exercer esta função em 02/04/2003. Além disto, sendo o cliente público, da esfera federal, após a posse do novo presidente, várias pessoas que atuavam no projeto do lado do cliente foram substituídas. Desta forma, a comunicação com o cliente piorou, dificultando a aprovação do Plano do Projeto e Documento de Requisitos. A auditoria recomendou que, minimamente, o plano fosse fechado para as atividades que estivessem sendo executadas, enquanto não havia uma definição sobre o envolvimento por parte do cliente.

O contrato estava sendo revisto. O gerente do projeto estava trabalhando intensamente na repactuação do contrato para, então, fechar o planejamento do projeto. A repactuação do contrato significa redefinir o escopo do projeto, a partir do que foi contratado. Para isto, foi realizado um levantamento do escopo implementado, o que ainda faltava ser implementado e uma nova proposta do que será feito até final do projeto.

Com a repactuação, o projeto deveria produzir um documento de requisitos com o último escopo contratado. Sobre estes requisitos, um sistema para controlar as mudanças deveria ser implementado. A recomendação do grupo de garantia da qualidade é que solicitações fossem registradas, avaliadas e acompanhadas até fechamento através da ferramenta de controle de mudanças adota.

O plano do projeto ainda se encontrava em versão não oficial, ou seja, não havia ainda sido passado por aprovação dos responsáveis. Como já citado, ainda assim, o plano foi considerado documento oficial do projeto pela equipe e grupo de garantia da qualidade.

Devido ao fato de o projeto ser de desenvolvimento de novas funcionalidades e manutenção, apesar da fase de replanejamento, a equipe continuava trabalhando normalmente, sobre os requisitos prioritários do cliente. Mesmo sem haver definição sobre o novo escopo, como o trabalho da equipe não parou, a auditoria recomendou que o documento de requisitos fosse aprovado o quanto antes, pelo menos sobre o escopo atual de trabalho.

O acompanhamento de riscos ainda não havia sido sistematizado pela equipe.

O plano de gerência de configuração deveria contemplar a gestão da configuração dos produtos gerados pela equipe Brasília e ser submetido à nova aprovação.

6.7.4 Implementação das KPAs no Projeto

Esta seção abordará as adaptações realizadas no processo do projeto, por KPA. Aqui são descritos os problemas encontrados e soluções empregadas durante todo o estudo de caso, sem explicitar a evolução obtida com o tempo. Isto facilitará o entendimento e a análise dos resultados finais obtidos.

Para cada KPA do nível 2, exceto gerência de subcontrato, são descritos:

- Práticas Adotadas: descreve brevemente as práticas adotadas pelo projeto para atender a KPA em questão. Estas práticas incluem as práticas de XP e as sugeridas pelo Guia XP-CMM2;
- Pendências: descreve pendências, ou seja, o que não foi ainda implementado no processo, segundo o Guia XP-CMM2;
- Análise: realiza uma análise das principais dificuldades e oportunidades para a implementação do Guia XP-CMM2 em relação a KPA em questão. Além disto, descreve brevemente ganhos e perdas obtidos pelo projeto com esta implementação.

6.7.4.1 Gerência de Requisitos

- **Práticas Adotadas**

Um documento contendo os requisitos de alto nível do projeto foi elaborado. Sobre estes requisitos, deveria ser implementada a gerência de escopo. Mudanças a estes requisitos deveriam ser controladas formalmente.

As estórias e tarefas de XP detalhavam os requisitos de alto nível e eram documentadas na ferramenta XPlanner [51], durante o planejamento da iteração, conforme XP.

Requisitos não técnicos e critérios de aceitação foram documentados no Plano do Projeto.

Desde a versão inicial, o Documento de Requisitos e Plano de Projeto foram colocados num repositório para gerenciamento das versões.

- **Pendências**

O Documento de Requisitos deveria ser aprovado, mas não foi devido a problemas do projeto já mencionados. Esta atividade está planejada para ser realizada.

Mudanças aos requisitos de alto nível do Documento de Requisitos deveriam passar por um controle formal. Isto não pôde ser testado no projeto, pois mudanças não ocorreram, já que a *baseline* de requisitos não estava estabelecida.

- **Análise**

O projeto apresentou grande dificuldade para estabelecer um ambiente de gestão de escopo. Esta dificuldade foi apresentada em decorrência de vários fatores: falta de representante do lado do cliente, troca de gerentes, replanejamento do escopo, cultura da equipe.

Desconsiderando os fatores externos, a cultura XP do projeto, pode ter contribuído para tornar mais difícil o gerenciamento do escopo. A equipe e o cliente estavam acostumados a trabalhar sobre as prioridades do cliente, vislumbrando o curto prazo do próximo *release* e não do contrato firmado.

Apesar disto, toda a equipe identificava a possibilidade de não cumprimento do contrato como um fator de alto risco para o projeto.

Por outro lado, a forma de trabalho sugerida por XP para documentar requisitos e priorizá-los foi de grande ajuda para o projeto. Especialmente por se tratar de desenvolvimento e manutenção. Toda a equipe julgava junta a prioridade de um *bug* e de uma nova implementação. A documentação na ferramenta XPlanner era realizada por todos. Todos entendiam todos os requisitos.

Os controles criados até o momento não impactaram o trabalho do dia a dia da equipe. O controle formal proposto para o escopo do projeto, apesar de ainda não completamente implementado, foi bem recebido pela equipe e organização.

6.7.4.2 Planejamento do Projeto

- **Práticas Adotadas**

Um plano do projeto foi elaborado contemplando o conteúdo de um plano XP, de um plano CMM e necessidades do C.E.S.A.R. Desta forma, o plano do projeto passou a definir: propósito, escopo, objetivos, descrição do cliente, organograma do projeto (com papéis da equipe e do cliente), responsabilidades, procedimentos, padrões e métodos, artefatos a serem entregues, instalações e ferramentas, treinamentos, planejamento de

releases, referencia a ferramenta XPlanner (que define o planejamento das próximas iterações), estimativas e planilha de riscos.

Uma planilha de riscos foi adotada. Nesta planilha os riscos são identificados, priorizados e planos de contingência e mitigação são definidos.

O plano do projeto foi submetido à análise e aprovação de líderes e alta gerência do C.E.S.A.R, para que os compromissos do projeto fossem acordados e firmados.

O processo de estimativa adotado para o projeto continuou a ser o sugerido por XP, que não é aderente ao nível 2 do CMM, conforme mencionado no Diagnóstico XP-CMM2.

- **Pendências**

O Plano do Projeto deveria ser aprovado formalmente, inclusive por representantes do cliente, mas não foi devido à problemas do projeto. Esta atividade está planejada para ser realizada.

Recursos críticos não foram estimados para o projeto. Vislumbrou-se identificar a massa de dados como um recurso crítico, já que o projeto trabalha com uma base muito grande de dados e consultas não devem ser lentas. Por passar por tantas prioridades, a identificação de recursos críticos foi preterida num primeiro momento.

- **Análise**

O plano do projeto contribuiu para melhorar a comunicação do projeto com a equipe e grupos externos ao projeto, como alta gerência, cliente e garantia da qualidade. Apesar de não haver aprovado formalmente o plano do projeto, representantes do cliente tiveram acesso ao plano. Informações que antes estavam na cabeça das pessoas, passaram a ser documentadas, como: planejamento das instalações, organograma, equipe, papéis e responsabilidades.

O plano passou a ser base para acompanhamento do projeto e auditoria do grupo de garantia da qualidade. O plano do projeto não alterou o dia a dia da equipe. A sua elaboração foi uma atividade a mais para o gerente do projeto, apenas.

6.7.4.3 Acompanhamento do Projeto

- **Práticas Adotadas**

O acompanhamento do projeto era bastante forte devido aos *releases* e iterações curtas e às reuniões diárias com toda a equipe. Esta KPA era implementada com a

atualização dos planos de *release* e iteração de acordo com mudanças no projeto, reuniões diárias para a comunicação das mudanças aos grupos afetados e acompanhamento das atividades técnicas.

Além disto, o projeto possuía acompanhamento das estimativas e do cronograma de atividades, onde a cada reunião de iteração se verificava o que foi cumprido na iteração passada e se planeja as atividades para a próxima iteração.

Um relatório mensal de progresso do projeto era enviado para a alta gerência do C.E.S.A.R e para o cliente. Este relatório contempla: dependências, riscos, atividades planejadas e realizadas, métricas. Desta forma, a alta gerência participa das mudanças aos compromissos com grupos externos e os riscos são acompanhados e reportados.

Revisões formais foram agendadas para serem conduzidas com representantes do cliente a cada *release*.

- **Pendências**

A cada mudança crítica o plano deveria ser alterado e submetido a uma nova revisão pelo cliente e alta gerência. Isto não ocorreu, já que nem mesmo a primeira versão do plano foi aprovada.

O relatório mensal do projeto não constituiu uma prática sistemática do projeto, pois em alguns meses não foi elaborado.

Nenhuma revisão formal ocorreu conforme planejado, apesar de que, em algumas implantações, o cliente era envolvido e tratava de aspectos do projeto com a equipe Brasília.

Os riscos não estavam sendo acompanhados sistematicamente. Recursos críticos não foram acompanhados já que não foram nem identificados e estimados.

- **Análise**

Da mesma forma que as KPAs gerência de requisitos e planejamento de projetos, as mudanças propostas para o acompanhamento trariam mais benefícios para grupos externos ao projeto que para o dia a dia da equipe.

O relatório de progresso deveria ser elaborado pelo gerente do projeto, em conjunto com informações passadas pela equipe nas reuniões de planejamento. Os grupos de garantia da qualidade e gerência de configuração também eram responsáveis por elaborar partes do relatório.

O cliente e alta gerência passaram a receber informações consolidadas e importantes sobre o progresso do projeto. O relatório também servia como informações para a equipe sobre riscos e progresso. Os grupos de garantia da qualidade e gerência de configuração reportavam seus resultados e eram acompanhados por estes relatórios.

Além do relatório, as revisões formais aumentariam a eficácia do acompanhamento por parte do cliente.

6.7.4.4 Garantia da Qualidade

- **Práticas Adotadas**

As atividades do Procedimento de Garantia da Qualidade do ProSCes, descrito na Seção 6.5.2 foram seguidas no projeto.

- **Pendências**

Apesar de comunicar os problemas do projeto, nenhum desvio identificado em auditoria foi oficialmente escalado para a alta gerência do C.E.S.A.R. Isto porque, foi acordado com a alta gerência, que o projeto precisaria de um tempo maior para se estruturar e organizar mediante todos os problemas enfrentados.

- **Análise**

As auditorias e revisões realizadas pelo grupo de garantia da qualidade foram bem recebidas pela equipe. O grupo contribuiu também para estabelecer padrões entre as equipes Brasília e Recife.

Além disto, o grupo de garantia da qualidade atuou fortemente como consultor de processos do projeto, acompanhando e sugerindo ações para resolução dos problemas.

A presença do grupo contribuiu de forma decisiva para que planos fossem elaborados e atualizados, visibilidade do projeto aumentasse para grupos externos ao projeto e equipe e problemas fossem, senão resolvidos imediatamente, tivessem a resolução planejada.

6.7.4.5 Gerência de Configuração

- **Práticas Adotadas**

Devido à natureza de desenvolvimento de novas funcionalidades e manutenção, o projeto enfrentava vários problemas no controle de versão do sistema.

Uma pessoa foi alocada para atuar em tempo parcial como gerente de configuração do projeto. Um plano de gerência da configuração foi elaborado para o

projeto descrevendo as *baselines*, itens de configuração, CCB (comitê que avalia as mudanças), entre outros. O plano foi aprovado pelo gerente do projeto e líderes e reaprovado quando mudanças impactantes eram realizadas.

Para não tornar custoso o processo de gerência de configuração dentro do projeto, foram adotadas ferramentas *freeware* para controle de versão (CVS [49]) e de mudanças (Bugzilla [46]).

O plano foi utilizado como base para controle das versões e mudanças dos itens de configuração do projeto.

Auditorias nas *baselines* passaram a ser realizadas, mas não de forma sistemática.

- **Pendências**

Não foi definido um critério para reaprovação do plano de gerência de configuração. Algumas vezes o plano apresentava diversas alterações, mas só era submetido à reaprovação, após identificação do problema pelo grupo de garantia da qualidade.

Inicialmente apenas o código foi tratado como item de configuração. No final deste experimento, o projeto estava em fase de planejamento para estender os itens de configuração do projeto.

- **Análise**

A KPA que obteve mais ganhos objetivos com a aplicação do Guia XP-CMM2 foi a gerência de configuração. Os problemas diminuíram consideravelmente. Toda a equipe considera, hoje, a gerência de configuração como um ponto forte do projeto.

No entanto, esta foi a KPA que mais impactou o dia a dia da equipe de desenvolvimento, que deveriam trabalhar sobre *baselines* estabelecidas, seguir padrões documentados no plano de gerência de configuração, realizar mudanças de forma controlada.

6.8 Aplicação do Guia XP-CMM2 em uma Empresa Incubada

Como já citado anteriormente, o C.E.S.A.R desenvolve projetos de software e cria empresas. A criação de empresas passa por um processo chamado incubação, onde o C.E.S.A.R provê recursos à empresa incubada. Esta empresa passará a ser uma empresa

do mercado, quando houver condições para se sustentar e gerar lucros. Isto, em geral, acontece com o aporte financeiro de um investidor.

No primeiro semestre de 2003, um investidor contratou um consultor para realizar um diagnóstico em uma das empresas incubadas do C.E.S.A.R para atestar a viabilidade de um possível investimento. Uma das conclusões do consultor foi que a empresa não possuía um processo de desenvolvimento de software estabelecido. No entanto, a empresa utilizava como base de desenvolvimento a metodologia XP.

Para solucionar o problema, o C.E.S.A.R solicitou que um diagnóstico do processo fosse realizado na empresa e que recomendações para os problemas identificados fossem propostas. O diagnóstico solicitado utilizou como base o nível 2 do CMM. Esta decisão foi tomada porque além de ser um modelo reconhecido mundialmente, o CMM foi utilizado na análise realizada pelo consultor do investidor.

Assim como no experimento relatado, o Diagnóstico XP-CMM2 contribuiu para a interpretação das práticas de XP quanto à aderência ao nível 2 do CMM. O Guia XP-CMM2 foi utilizado para propor soluções aos problemas identificados. Como era de se esperar, a empresa possuía algumas peculiaridades, de forma que nem todos os problemas e soluções estavam contemplados no Diagnóstico e Guia XP-CMM2.

Como empresa incubada do C.E.S.A.R, ela poderia adotar o ProSCes como processo de desenvolvimento, visando ter um processo mais estabelecido e formal. No entanto, um dos pré-requisitos para o trabalho era que a empresa continuasse a utilizar XP, pois já vinha obtendo bons resultados com a metodologia há um tempo. Além disto, os integrantes da empresa colocaram que a metodologia atendia às expectativas e cultura da empresa.

Ao final do diagnóstico realizado, um relatório foi elaborado, descrevendo o contexto do diagnóstico, método utilizado, pontos fortes, fracos e recomendações. Este relatório foi inicialmente validado pela empresa, que apontou pequenos problemas de entendimento e depois foi formalmente aceito com as recomendações propostas. A partir de então, o relatório foi enviado para a alta gerência do C.E.S.A.R. O **Apêndice A**, exibe o relatório final deste trabalho, excluindo detalhes da empresa, visando manter o sigilo requerido das informações.

A empresa passou a implementar as recomendações. O grupo de garantia da qualidade não acompanhou a implementação das recomendações, pois esta atividade não fazia parte do escopo do trabalho, no entanto, o grupo atuou esclarecendo dúvidas e propondo soluções.

Dentre as atividades já implementadas pela empresa estão: a elaboração de planos de projeto, garantia da qualidade e configuração; identificação e acompanhamento de riscos e recursos críticos computacionais; adoção de ferramentas para a gerência de versão e mudanças, entre outras.

Em reuniões, a empresa declarou que considera que as recomendações irão contribuir para seu trabalho e que não irão impactar negativamente a dinâmica de trabalho.

O resultado final foi entregue ao investidor para avaliação e continuidade dos trabalhos.

6.9 Considerações Finais

O Guia XP-CMM2 foi aplicado de forma sistemática em um projeto real. A implementação do Guia e análise de resultados foram realizados através da abordagem de uso de um procedimento de garantia da qualidade aderente ao nível 2 do CMM. A principal ferramenta para viabilidade da implantação e análise de resultados foram as auditorias do grupo de garantia da qualidade, realizadas através de análise de documentos e entrevistas aos membros da equipe do projeto.

Os objetivos deste estudo declarados na Seção 6.1, foram considerados atendidos:

- "Verificar se o Guia é utilizável no mundo real"

O guia foi aplicado a um projeto real e a uma empresa incubada. Nem todos os aspectos do Guia puderam ser testados devido a problema dos projetos, no entanto, a grande maioria foi adotada.

- "Verificar se os benefícios declarados do uso de XP podem ser alcançados com as modificações sugeridas pelo Guia" e "Verificar se um projeto XP perde a agilidade esperada se inserir práticas do CMM nível 2"

Como já mencionado, para as duas aplicações do Guia XP-CMM2, as conclusões obtidas, tanto pelo grupo de garantia da qualidade, quanto pela equipe do projeto é que o Guia não afetou negativamente a dinâmica de trabalho.

O formalismo adicionado ao projeto se deu essencialmente no nível gerencial e em pouco impactou o trabalho da equipe como um todo. O projeto não perdeu a dinâmica proposta por XP: as iterações curtas, *releases* freqüentes, reuniões diárias, integração contínua, programação em pares contribui para a comunicação e motivação da equipe. As mudanças às iterações não sofreram nenhum impacto. Mudanças aos *releases*, considerados escopo do projeto, passaram a ser feitas através de solicitação de mudanças, tratadas nas reuniões de planejamento das iterações.

A KPA que mais alterou o trabalho da equipe foi a de gerência de configuração. No entanto, o resultado positivo obtido com a aplicação das práticas sugeridas pelo Guia XP-CMM2 indicou que as mudanças realizadas foram consideradas necessárias e benéficas pela equipe.

- "Verificar se é possível conviver em um mesmo ambiente com XP e com o nível 2 do CMM" e "Benefícios de aplicar a abordagem do nível 2 do CMM em um projeto XP"

O Guia XP-CMM2, com práticas do CMM e de XP, propiciou vários benefícios ao projeto. O gerenciamento passou a ser mais eficaz e visível através da identificação de riscos, maior envolvimento da alta gerência, plano contemplando todos os principais aspectos do projeto e planejamento de revisões formais.

Os problemas de gerenciamento de configuração foram consideravelmente diminuídos e o processo se tornou mais visível para toda a equipe.

A gestão de escopo passou a ser uma preocupação da equipe.

- "Averiguar a dificuldade em assimilar práticas do nível 2 do CMM em um projeto com cultura XP"

A maior dificuldade na implantação do Guia XP-CMM2 foi a implementação de um sistema de gestão de escopo. Este problema foi aumentado por problemas internos do projeto, como a substituição do gerente do projeto, e relação com o cliente do governo.

A elaboração e uso de planos não caracterizaram *overhead* para a equipe. As práticas de gestão, apesar de nem todas implementadas, foram consideradas necessárias por todos os envolvidos no estudo.

7 Conclusões e Trabalhos Futuros

Este trabalho foi motivado especialmente pela busca de respostas em relação aos mundos das metodologias ágeis e dos modelos de qualidade: se eles poderiam conviver num mesmo ambiente e, em especial, se seria possível obter benefícios destes dois mundos. Para isto foram estudados a fundo a metodologia ágil mais conhecida e utilizada atualmente, XP, e o nível 2 do CMM.

Um Diagnóstico XP-CMM2 foi realizado, para identificar a aderência de XP ao nível 2 do CMM. Depois, para os problemas identificados, soluções foram propostas, resultando no Guia XP-CMM2. Finalmente, o Diagnóstico XP-CMM2 e o Guia XP-CMM2 foram utilizados em projetos reais.

O uso do Diagnóstico XP-CMM2 e do Guia XP-CMM2 em projetos reais mostrou que é possível se beneficiar dos mundos XP e nível 2 do CMM, que eles podem conviver em um mesmo ambiente, desde que ajustes sejam realizados, e que eles realmente se complementam.

As conclusões deste trabalho estão descritas neste capítulo, que está organizado da seguinte forma:

- Seção 7.1 – Principais Contribuições: relaciona as principais contribuições resultantes deste trabalho;
- Seção 7.2 – Dificuldades Encontradas: relata quais foram as principais dificuldades encontradas para a realização do trabalho;
- Seção 7.2.5 – Trabalhos Relacionados: descreve trabalhos relacionados a este;
- Seção 7.3 – Artigos Publicados: descreve os artigos publicados, frutos deste trabalho;
- Seção 7.4 – Trabalhos Futuros: descreve trabalhos que podem ser realizados tendo como base o presente trabalho.

7.1 Principais Contribuições

A primeira grande contribuição deste trabalho se deu na interpretação do atendimento de XP ao nível 2 do CMM, o que resultou no Diagnóstico XP-CMM2. O Diagnóstico XP-CMM2 foi realizado de forma aprofundada, objetivando seguir o mesmo rigor aplicado em uma avaliação oficial do SEI para determinar o nível de maturidade de uma organização. Assim, o Diagnóstico XP-CMM2 considerou os elementos de mais baixo nível do CMM, que são as práticas e subpráticas. Em relação a XP, o estudo também se deu de forma aprofundada. O diagnóstico considerou além das práticas, princípios e valores de XP, aspectos inseridos no dia-a-dia de projetos como as reuniões diárias. O Diagnóstico XP-CMM2 contribuiu para definir a distância entre o nível 2 do CMM e XP, até então somente comparados em níveis mais altos do CMM e de XP [31].

Outra contribuição foi o Guia XP-CMM2, que define um caminho para utilizar XP de forma a estar aderente ao nível 2 do CMM. Com este guia, empresas que possuem foco na entrega rápida de software e em produtividade, e que comumente possuem interesse em utilizar metodologias ágeis como XP, podem selecionar as práticas do nível 2 do CMM que desejam implementar, contribuindo para aumentar a confiança na metodologia utilizada. Empresas que possuem o interesse primordial de seguir modelos de qualidade amplamente reconhecidos mundialmente, como o CMM, poderão utilizar XP em projetos menores, menos críticos, ou em projetos que se espera uma agilidade maior. Juntamente com a diretriz para utilizar XP e CMM nível 2 em um mesmo ambiente, o Guia XP-CMM2 sugere templates e ferramentas para facilitar seu uso.

A aplicação do Diagnóstico XP-CMM2 e do Guia XP-CMM2 a projetos reais ofereceu a possibilidade de avaliar os resultados de se empregar as abordagens de desenvolvimento ágil – sendo representado por XP – e de modelos de qualidade – representado pelo nível 2 do CMM – em um mesmo ambiente. Esta aplicação mostrou os ganhos, as perdas e as dificuldades em se trabalhar com estes dois fundamentos.

7.2 Dificuldades Encontradas

Várias dificuldades foram encontradas e enfrentadas durante o desenvolvimento deste trabalho. As próximas sub-seções irão detalhar estas dificuldades.

7.2.1 Falta de Rigor de XP

XP defende a informalidade para o processo de desenvolvimento e também o aplica na sua própria definição. Não há a *baseline* XP, ou a versão mais atual de XP, como no RUP, por exemplo. XP é definida em livros, que muitas vezes se contradizem. Para citar um exemplo, Beck et al afirma que em "materiais velhos de XP"[2] costumava-se utilizar um fator de carga para apoiar o planejamento e estimativa, mas que isto não deve mais ser utilizado [24]. De fato, os "materiais velhos de XP" que citam este fator de carga são os primeiros livros de XP (como [25], [39]). Para saber que este fator de carga não é mais utilizado nas estimativas deve-se ler o último livro de XP publicado por Beck, no entanto este último livro não se propõe a descrever uma nova versão de XP, mas abordar de forma mais específica o planejamento num projeto XP.

As contradições, ou interpretações, de XP dificultaram a realização do Diagnóstico XP-CMM2, que foi elaborado de forma sistemática e o mais fiel possível ao CMM e a XP.

7.2.2 Trabalho Interpretativo

A elaboração do Diagnóstico XP-CMM2 e as soluções propostas pelo Guia XP-CMM2 são baseadas fundamentalmente em interpretação. O modelo CMM é um guia para melhoria de processos das organizações, elaborado de forma a ser aplicado em organizações de diferentes tamanhos e culturas [32]. Portanto, foi elaborado visando ser abrangente e genérico.

Além disto, o que se estava comparando – XP e CMM – possuem essências diferentes. As premissas são diferentes. Como exemplo, o CMM considera que qualidade do produto é obtida a partir da qualidade no processo. XP, em vários momentos refuta esta idéia. Além disto, o CMM e a engenharia de software de uma forma geral, consideram que o custo para realizar uma mudança aumenta consideravelmente com o passar do tempo. XP parte do princípio que isto não é verdade (conforme detalhado no Capítulo 2, na Seção 2.3.3, Figura 2-1 – Curva do Custo da Mudança).

[2] Do inglês *old stuff*

Os vocabulários de XP e CMM são diferentes, assim, para citar alguns exemplos, gerente do projeto para o CMM é *tracker* para XP, plano para XP é completamente diferente de plano para o CMM, XP não menciona alta gerência em seus projetos.

Decisões baseadas em interpretação são mais difíceis de serem realizadas que decisões baseadas em fatos. O embasamento deve ser maior, as considerações devem ser abrangentes, a argumentação deve ser documentada, para que justifique as decisões, conforme realizado nos Capítulos 4 e 5 deste trabalho. Interpretação de produtos essencialmente diferentes como XP e CMM são ainda mais difíceis. Neste contexto, é importante ressaltar a experiência e base utilizada pela aluna para realizar os julgamentos necessários: treinamento no método oficial do SCE [34], participação da equipe de implementação do nível 2 na empresa C.E.S.A.R [52], participação da equipe de avaliação oficial que reconheceu o nível 2 do C.E.S.A.R em Junho de 2003.

7.2.3 Descaracterização de XP

Um dos objetivos do Guia XP-CMM2 era propor soluções que não descaracterizassem XP. Caso contrário, provavelmente os benefícios da adoção da metodologia ágil XP não seriam obtidos. Há bastante literatura disponível [10] a respeito de soluções empregadas para atender o CMM, no entanto, grande parte delas poderiam ferir os objetivos e a essência de XP.

Um exemplo é a prática de estimativa de tamanho para os principais produtos de trabalho e atividades requerida pelo CMM (SPP.Ac9). Apesar de o Diagnóstico XP-CMM2 considerar que esta prática não é atendida por XP, o Guia XP-CMM2 não recomendou alterações. Isto porque a forma como as estimativas são realizadas em XP agregam diversos valores e princípios da metodologia, como assumir responsabilidade, coragem, fundamentada no *feedback*, opção pela simplicidade e outras. Além disto, o estudo de caso demonstrou a efetividade das estimativas como são propostas por XP.

A decisão de não sugerir uma solução para a não satisfação da prática SPP.Ac9 foi a mais difícil, demorada e estudada deste trabalho, já que iria ferir uma das práticas do CMM. No entanto, a identificação de outras soluções também constituíram um grau elevado de dificuldade.

7.2.4 Aplicação em um Projeto Real

Apesar de trabalhar em uma empresa que adota o CMM como fundamento para a realização de projetos de software e que possui interessados em XP, a aplicação de todo o Guia XP-CMM2 em um projeto real constituiu uma dificuldade deste trabalho.

Mesmo havendo um projeto selecionado para utilizar o Guia XP-CMM2, sua aplicação se deu com restrições. Primeiramente restrições políticas: *templates* e algumas práticas sugeridas não puderam ser utilizados, já que o projeto deveria seguir o processo padrão da organização. Além disto, o projeto selecionado enfrentou diversas dificuldades ao longo do estudo (conforme descrito no Capítulo 6), o que comprometeu seu poder de assimilar o Guia XP-CMM2.

Além disto, o objetivo de implementação de todo o Guia XP-CMM2 demandaria tempo para institucionalização. Como se sabe, inserir novos processos, mudar cultura, transformar ações dispersas em sistematização é um trabalho árduo e duradouro. O tempo dedicado ao estudo, apesar de relativamente longo, não foi suficiente para implantar e institucionalizar todo o Guia XP-CMM2.

7.2.5 Trabalhos Relacionados

Conforme mencionado no Capítulo 1, Mark Paulk publicou um artigo descrevendo como XP atenderia ao CMM [31]. Esta análise foi realizada tomando como base as metas do CMM, para todos os níveis do CMM. As metas são um componente de mais alto nível que as práticas e sub-práticas, utilizadas como base para a comparação do Diagnóstico XP-CMM2. Apesar de as práticas e subpráticas não serem elementos normativos do CMM (não são utilizados para determinar nível de maturidade, por exemplo), este trabalho optou por utilizá-los, pois, como já citado, eles são a base para investigação de dados em uma avaliação oficial. Assim, a utilização dos componentes práticas e subpráticas, aprofunda o estudo de comparação entre XP e CMM, além se assemelhar a uma avaliação oficial.

Assim como Paulk, Antonio Bonato publicou um artigo comparando XP ao CMM, em seus diversos níveis [1], tomando como base o componente “metas”. Bonato conclui que “XP atende a boa parte das KPAs do nível 2 e 3 do CMM, isto quando se faz uma leitura menos rigorosa destas KPAs, usando-se o bom senso”. Neste trabalho, a

essência da análise também se deu tomando como base as metas do CMM e não os elementos práticas e sub-práticas.

Jonas Martinsson, elaborou um trabalho de dissertação com o objetivo de "amadurecer XP" (tornar XP mais próximo do CMM) sugerindo alterações de forma a atender as metas das KPAs de todos os níveis do CMM [23]. Neste trabalho, Martinsson indica que apenas duas metas da KPA SQA deixam de ser atendidas no nível 2. Por isto, sugere pequenas alterações em XP, de forma a contemplar o papel de SQA. A diferença de resultados entre o trabalho de Martinsson e este trabalho pode ser justificada pela diferença entre os componentes utilizados para a comparação entre XP e CMM: no trabalho de Martinsson as metas e neste trabalho as práticas e sub-práticas.

Nawrocki et al, relacionaram XP e CMM, mas não de forma a torná-los complementares, como este trabalho. O trabalho define um modelo de maturidade para XP [17], indicando como amadurecer seu uso.

7.3 Artigos Publicados

Como frutos deste trabalho, dois artigos foram publicados relatando experiências adquiridas a partir do uso de XP relacionadas a fundamentos da engenharia de software e CMM.

O primeiro artigo, chamado "O Impacto do Uso de Práticas de Metodologias Ágeis de Desenvolvimento sobre a Gerência de Projetos", escrito por Renata Endriss Carneiro Campelo, Cibele de Ataíde Feitosa e Teresa Maria de Medeiros Maciel, foi publicado no XIII Congresso Internacional de Tecnologia de Software, (CITS), realizado no ano de 2002, em Curitiba – PR, Brasil. Este artigo relatou os benefícios obtidos pela gerência de um projeto real através do uso de XP [37].

O artigo "O Uso de Extreme Programming em uma Organização CMM Nível 2" descreveu resumidamente o estudo de caso relatado no Capítulo 6. O artigo foi apresentado no II Simpósio Brasileiro de Qualidade de Software, realizado em Fortaleza – CE, Brasil, em setembro de 2003 e foi escrito por Renata Endriss Carneiro Campelo, Fábio Gomes Silva (líder técnico do projeto) e Hermano Perrelli de Moura [38].

7.4 Trabalhos Futuros

Este trabalho pode ser estendido de várias formas visando aumentar a contribuição para as áreas de engenharia de software e qualidade de software e para a comunidade de software, que deseja se beneficiar dos mundos das metodologias ágeis e dos modelos de qualidade.

Alguns trabalhos sugeridos são:

- Realizar este trabalho utilizando o CMMI como modelo de referência. Esta opção não foi considerada para este trabalho porque o *framework* do CMMI (com seus modelos e método de avaliação) foi lançado após o início deste trabalho. Além disto, o SEI indica que o SW-CMM é um modelo que ainda será utilizado durante cerca de dois anos;

- Implementar uma ferramenta que suporte o Guia XP-CMM2. O Guia XP-CMM2 sugere *templates* e ferramentas, no entanto uma ferramenta que implementasse todos, ou a maioria dos aspectos do guia facilitaria bastante a sua adoção e, provavelmente, aumentaria a produtividade da equipe. Uma possibilidade é estender ferramentas já existentes, como a XPlanner [51];

- Realizar este trabalho para a KPA *Software Product Engineering* (SPE), do nível 3 do CMM. Esta KPA está diretamente relaciona ao desenvolvimento do produto final, o software. Assim, define práticas para elicitar, documentar e validar requisitos; realizar testes unitários, de integração de sistema e aceitação; projetar o software; implementar e outras. Esta KPA possui várias interseções com XP. O trabalho de realizar um diagnóstico e propor soluções para os pontos falhos poderia ser bastante útil para a comunidade XP identificar o que se espera das práticas de desenvolvimento do ponto de vista de um modelo de qualidade como o CMM;

- Da mesma forma que o item acima, a realização deste trabalho para a prática *Peer Review* (PR) do nível 3 do CMM poderia apresentar um grande ganho para os interessados em adotar CMM e XP. A prática programação em pares de XP muitas vezes é apontada como suficiente para atender a KPA PR. No entanto, um

estudo detalhado poderia contribuir para a melhoria da sistemática das revisões em pares de XP e o entendimento do que se espera de revisões em pares por modelos de qualidade como o CMM.

7.5 Considerações Finais

Cada vez mais a comunidade de software vem aceitando, entendendo e aplicando metodologias ágeis. Elas se mostram uma alternativa viável especialmente para grupos e organizações pequenas, por serem muito fáceis de implantar e de ter seus conceitos disseminados por toda a equipe.

Apesar disto, é notório a falta de tratamento de aspectos relevantes do desenvolvimento de software por grande parte destas metodologias. A combinação das metodologias ágeis com estes aspectos, seja com base em um modelo de qualidade como o CMM, seja com base nos resultados difundidos da engenharia de software, se mostra uma solução factível e viável para diversos ambientes de desenvolvimento.

Referências

[1] A. Bonato, “Extreme Programming e Qualidade de Software”, Congresso Brasileiro Extreme Programming Brasil (http://www.xispe.com.br/evento2002/index2.html, último acesso em 13/10/2003), Dezembro de 2002.

[2] A. Cockburn, Writing Effective Use Cases, Primeira Edição, Addison-Wesley, 2000.

[3] A. Leon, A Guide to Software Configuration Managemen, Artech House, Abril de 2000

[4] A.W. Morales, “Going to Extremes”, Software Development Magazine, Janeiro de 2002.

[5] C.H.P. Mello et al, ISO 9001:2000: Sistema de Gestão da Qualidade para Operações de Produção e Serviços, São Paulo, Atlas, 2002.

[6] CMMI for Software Engineering, Version 1.1, Continuous Representation (CMMI-SW, V1.1, Continuous), Technical Report CMU/SEI-2002-TR-028, Agosto de 2002.

[7] CMMI for Software Engineering, Version 1.1, Staged Representation (CMMI-SW, V1.1, Staged), Technical Report, CMU/SEI-2002-TR-029, Agosto de 2002.

[8] D. Bellin et al, The CRC Card Book, Addison-Wesley, Junho de 1997.

[9] D. Zubrow et al, Maturity Questionnaire, Technical Report, CMU/SEI-94-SR-7, Junho 1994.

[10] D.J. Reifer, “XP and the CMM”, IEEE Software, vol. 20, no. 3, Maio de 2003.

[11] F. Paul Clipp III, Total Quality Management, The National Management Association, 1990.

[12] G. Booch, Object Solutions – Managing the Object Oriented Project, Addison-Wesley, 1995.

[13] I. Graham, Requirements Engineering and Rapid Development: An Object-Oriented Approach, Addison-Wesley, Junho 1999

[14] I. Sommerville, Software Engineering, Fifth Edition, Addison-Wesley, 1995.

[15] IEEE Standards Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, 1993.

[16] J. Grenning, "Launching Extreme Programming at a Process-Intensive Company", IEEE Software, vol. 18, No. 6, Novembro 2001.

[17] J. Nawrocki et al, "Toward Maturity Model for eXtreme Programming", Poznan University of Technology, 60-965 Poznan, Poland.

[18] J. Rasmusson, "Introducing XP into Greenfield Projects: Lessons Learned", IEEE Software, vol. 20, no. 3, Junho de 2003.

[19] J.A. Highsmith, Agile Software Development Ecosystems, Addison-Wesley, 2002.

[20] J.H. Johnson, "Micro Projects Cause Constant Change", XP 2001 Conference, http://www.xp2003.org/conference/program.html (último acesso em 14/10/2003).

[21] J.R. Nawrocki et al, "Comparison of CMM Level 2 and eXtreme Programming", Quality Connection - 7th European Conference on Software Quality, Helsinki (Finland), Junho de 2002.

[22] J.T. Nosek, "The case for collaborative programming", Communications of the ACM, vol. 41, No. 3, 1998.

[23] Jonas Martinsson, Maturing Extreme Programming Through the CMM, Master's Thesis in Computer Science, Outubro de 2002, CODEN: LUNDFD6/NFCS/NFCS-5248/1—35/2002, Department of Computer Science, Lund University, Lund, Sweden.

[24] K. Beck et al., Planning Extreme Programming, Addison-Wesley, 2000.

[25] K. Beck, Exteme Programming Explained: Embrace change, Addison-Wesley, 1999.

[26] L. Williams et al., “Strengthening the case for pair programming”, IEEE Software, vol. 17, No. 4, 2000.

[27] M. Fowler et al, “Put Your Process on a Diet”, Software Development Magazine, Dezembro de 2000.

[28] M. Fowler et al., “The Agile Manifesto”, Software Development Magazine, Agosto de 2001.

[29] M. Fowler, UML Destilled: A Brief Guide to the Standard Object Modeling Language, 2nd Edition, Addison-Wesley, 1999.

[30] M.C. Paulk et al., The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, 1995.

[31] M.C. Paulk, “Extreme Programming from a CMM Perspective”, IEEE, vol. 18, Novembro 2001.

[32] M.C. Paulk, “Using the Software CMM in Small Organizations”, The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality, Portland, Oregon, 13-14 Outubro de 1998, pp. 350-361.

[33] M.C. Paulk, “Using the Software CMM With Good Judgment”, ASQ Software Quality Professional, vol. 1, no. 3, Junho de 1999.

[34] P. Byrnes et al, Software Capability Evaluation, version 3.0, Technical Report, CMU/SEI-96-TR-002, ESC-TR-96-002, Abril de 1996.

[35] P. Krunchten, The Rational Unified Process: an introduction, Addison-Wesley, 1998

[36] P. Schuh, Recovery, “Redemption, and Extreme Programming”, IEEE Software, vol. 18, No. 6, Novembro de 2001

[37] R. Endriss et al, “O Impacto do Uso de Práticas de Metodologias Ágeis de Desenvolvimento sobre a Gerência de Projetos”, II Simpósio Brasileiro de Qualidade de Software, Setembro de 2003, Fortaleza – CEARÁ, Brasil.

[38] R. Endriss et al, “O Uso de Extreme Programming em uma Organização CMM Nível 2”, Congresso Internacional de Tecnologia de Software 2002 (CIT’s 2002), Junho de 2002, Curitiba – Paraná, Brasil.

[39] R. Jeffries et al, Extreme Programming Installed, Addison-Wesley, 2000.

[40] R.S. Pressman, Software Engineering: A Practitioner's Approach, Fourth Edition, McGraw-Hill, 1997.

[41] Software Engineering Institute, CMM-Based Appraisal, Software Capability Evaluation, V3.0, Team Training Reference Materials Notebook, presented by Integrated System Diagnostics.

[42] The Standish Group, "Extreme CHAOS 2001", February 2001.

[43] W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, 1989.

[44] *Web site* Agile Alliance, http://www.agilealliance.org/ (último acesso em 13/10/2003).

[45] *Web site* An International Collaboration to Develop a Standard on Software Process Assessment, http://www.sei.cmu.edu/iso-15504/ (último acesso em 13/10/2003).

[46] *Web site* Bugzilla, http://bugzilla.mozilla.org/ (último acesso em 13/10/2003).

[47] *Web site* CMMI, http://www.sei.cmu.edu/cmmi/ (último acesso em 13/10/2003).

[48] *Web site* Crystal Methodologies, http://crystalmethodologies.org/ (último acesso em 13/10/2003).

[49] *Web site* CVS, , http://www.cvshome.org/ (último acesso em 13/10/2003).

[50] *Web site* da ferramenta Ant, http://ant.apache.org/ (último acesso em 13/10/2003).

[51] *Web site* da ferramenta XPlanner, http://www.xplanner.org/ (último acesso em 13/10/2003).

[52] Web site do C.E.S.A.R, http://www.cesar.org.br (último acesso em 13/10/2003).

[53] *Web site* Extreme Programming: A gentle introduction, http://www.extremeprogramming.org (último acesso em 13/10/2003).

[54] *Web site* IEEE, http://www.ieee.org/ (último acesso em 13/10/2003).

[55] *Web site* Integrated System Diagnostics Brasil, http://www.isdbrasil.com.br/ (último acesso em 13/10/2003).

[56] *Web site* ISO – International Organization for Standardization, http://www.iso.ch/, (último acesso em 13/10/2003).

[57] *Web site* Poznam University of Technology, http://www.cs.put.poznan.pl (último acesso em 13/10/2003).

[58] *Web site* Project Management Institute, http://www.pmi.org/ (último acesso em 13/10/2003).

[59] *Web site* Rational Software, http://www.rational.com/ (último acesso em 13/10/2003).

[60] *Web site* Software Engineering Institute, http://www.sei.cmu.edu (último acesso em 13/10/2003).

[61] *Web site* The Standish Group, http://www.standishgroup.com/ (último acesso em 13/10/2003).

[62] *Web site* XProgramming.com, http://www.xprogramming.com (último acesso em 13/10/2003).

Apêndice A – Relatório de Auditoria da Unidade de Negócio

Diagnóstico de Processos

[redacted]

1 Dados do Diagnóstico

Escopo:	[redacted]
Período:	[redacted]
Auditor:	[redacted]
Destinatários:	[redacted]

1. Objetivo

O diagnóstico realizado objetiva a avaliação da aderência dos processos e práticas adotados nos projetos [redacted], em relação às principais práticas

das seguintes áreas chaves de processo do nível 2 e 3 do CMM [30]: Gerenciamento de Requisitos, Planejamento de Projetos, Acompanhamento de Projetos, Gerência de Configuração e Engenharia do Produto de Software.

A área chave Gerência de Subcontrato não foi contemplada neste diagnóstico, pois os projetos não realizam subcontratam produtos e/ou serviços que impactem o desenvolvimento do software.

2. Contexto

O projeto [redacted] tem como objetivo [redacted]. A equipe é composta de três desenvolvedores, sendo um deles responsável por assumir os papéis de cliente (especificando os requisitos do sistema e realizando testes de aceitação), gerente do projeto e arquiteto. [redacted].

[redacted].

Os projetos utilizam como base a metodologia de desenvolvimento de software Extreme Programming (XP) [25].

3. Método

Este diagnóstico foi conduzido por Renata Endriss através de entrevistas com representantes das equipes dos projetos, análise de documentos produzidos pelo projeto e procedimentos documentados.

Os seguintes documentos foram utilizados como base para realização do diagnóstico:

[1] M.C. Paulk et al., The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley Pub Co, 1995

[2] K. Beck, Exteme Programming Explained: Embrace change, Addison-Wesley Pub Co, 1999

Alguns produtos gerados pelos projetos foram analisados, dentre eles:

[3] Ferramenta XPlanner do projeto

[4] Planilha de Acompanhamento

[5] Ferramenta Twiki do projeto

[6] Documento "Descrição Tecnológica [redacted]"

2 Resultados

2.1 Boas Práticas

Gerenciamento de Requisitos

2.1.1 Os requisitos são documentados através de estórias e tarefas, em cartões de papel (como prevê a metodologia XP) e na ferramenta XPlanner [3]. Requisitos são revisados por toda a equipe nas reuniões de planejamento de release e de iteração.

2.1.2 Mudanças aos requisitos não são implementadas sem controle: elas são avaliadas nas reuniões de planejamento de release e de iteração, por todos da equipe.

2.1.3 Planejamento do projeto e produtos gerados são elaborados sobre os requisitos do release e da iteração.

Planejamento e Acompanhamento de Projetos

2.1.4 O planejamento dos projetos é realizado através de um plano de releases, que contempla as datas dos releases, iterações do próximo release, requisitos (estórias) de cada iteração e estimativas. Além disto, em documentos diversos há descrições sobre o que é cada projeto.

2.1.5 Estimativas de esforço são realizadas por toda a equipe para cada tarefa da iteração, que possui uma granularidade de forma a maximizar a precisão das estimativas. As estimativas de esforço são acompanhadas diariamente e nas reuniões de planejamento da iteração.

2.1.6 Um cronograma é elaborado para as iterações, que possuem tempo fixo: três semanas. As atividades e responsáveis por executá-las são discriminadas para cada iteração.

2.1.7 Os projetos possuem forte acompanhamento através de reuniões diárias, iterações e releases curtos.

Gerência da Configuração

2.1.8 Os projetos utilizam o CVS como repositório para arquivos que compõem o software (código e arquivos XML).

Engenharia do Produto de Software

2.1.9 A arquitetura de alto nível é elaborada pelo arquiteto com o apoio dos demais membros da equipe. A arquitetura reflete os requisitos de alto nível do sistema.

2.1.10 O código é atualizado e melhorado através de constantes refactorings, realizados sobre a arquitetura do sistema.

2.1.11 Testes unitários são realizados pelo responsável de cada estória.

2.2 Desvios de Processo de Software

Gerenciamento de Requisitos

2.2.1 O escopo completo do projeto não está documentado. Apenas os requisitos funcionais da próxima entrega são documentados.

Risco: Não haver visibilidade sobre o escopo do projeto como um todo. Não haver entendimento comum e acordo entre os interessados no projeto sobre o escopo do projeto. Mudanças de mais alto nível não podem ser controladas e ter seu impacto avaliado se o escopo não é definido.

2.2.2 Não há um documento de referência para os requisitos não funcionais do sistema, critérios de aceitação dos requisitos, requisitos não técnicos.

Risco: O software final não contemplar os requisitos não funcionais do sistema. Não haver clareza sobre o atendimento aos requisitos.

2.2.3 Não há registro sobre as mudanças nos requisitos do projeto.

Risco: Não haver visibilidade sobre o progresso do projeto. Se o replanejamento e mudanças não são registradas, não há como ter visibilidade sobre os motivos que levam ao não cumprimento de compromissos estabelecidos, caso isto aconteça.

2.2.4 Mudanças não são refletidas em todos os documentos dos projetos.

Risco: Documentos desatualizados comprometem o acordo sobre compromissos do projeto, o controle sobre as mudanças, a linha de base para desenvolvimento, a integridade do sistema.

Planejamento e Acompanhamento de Projetos

2.2.5 Os projetos possuem um plano de releases, que contempla as datas dos releases, iterações do próximo release, requisitos (estórias) de cada iteração e estimativas de esforço. Este plano não contempla informações importantes de planejamento de projetos, como: propósito, escopo, ciclo de vida, procedimentos adotados, artefatos a serem gerados, estimativas, riscos, equipe.

Risco: Falta de visibilidade dos principais interessados a respeito do projeto como um todo. Falta de base para acompanhar o projeto. Conhecimento das informações do projeto apenas nas cabeças das pessoas, o que pode levar a falta de entendimento comum sobre o projeto.

2.2.6 Estimativas de esforço são realizadas, no entanto, estimativas de custo não são realizadas e acompanhadas.

2.2.7 Recursos críticos computacionais não foram formalizadas e não são acompanhados nos projetos.

Risco: O sistema não estar apto a entrar em execução quando estiver pronto.

2.2.8 Premissas utilizadas nas estimativas não são documentadas.

Risco: Impossibilidade de reconstruir as estimativas, caso seja necessário realizar replanejamento.

Gerência da Configuração

2.2.9 Apesar de utilizar um repositório para o código, não há gerência de configuração no projeto: não há plano de gerência de configuração descrevendo itens de configuração, baselines, momento de entrada dos itens nas baselines, impacto das mudanças às baselines, entre outros.

Risco: Falta de controle de mudanças, possibilidade de retrabalho, de falta de integridade entre os arquivos que compõem o sistema.

Engenharia do Produto de Software

2.2.10 A arquitetura do projeto está documentada em diversos documentos e não está sob gerência de configuração.

Risco: Inconsistência entre código e arquitetura, dificuldade para mantê-la íntegra, caso precise ser alterada.

2.2.11 Casos de testes não são identificados e documentados.

Risco: Testes não eficazes, produto que não atende aos requisitos.

2.2.12 Não está formalizada a forma de registro e acompanhamento dos problemas encontrados nos testes de aceitação. Os problemas são registrados na ferramenta Twiki ou é enviado por e-mail ao responsável ou virará uma estória para a próxima iteração.

Risco: Problemas identificados não serem corrigidos, não saber em que versão do produto o problema foi corrigido.

2.3 Oportunidades de Melhoria de Processo e Melhorias em Progresso

2.3.1 Formalizar o padrão de codificação utilizado no código.

2.3.2 Formalizar as revisões em pares. Atualmente a equipe realiza inspeções sobre código de forma ad-hoc.

2.3.3 Foi acordado recentemente que o membro da equipe que representa o cliente no projeto executa testes de aceitação no final das iterações, no entanto, esta prática ainda não está institucionalizada nos projetos.

4. Recomendações

Recomendação	Item(ns) Atendido(s)
Elaborar um documento com o planejamento macro dos projetos, contendo: propósito, ciclo de vida, procedimentos adotados, artefatos a serem gerados, estimativas, riscos, recursos críticos computacionais, equipe e responsabilidades, planejamento de releases e iterações, planejamento de testes, escopo, requisitos não funcionais, critérios de aceitação dos requisitos.	2.2.1, 2.2.7, 2.2.11, 2.2.2
Registrar as mudanças (motivo, impacto, histórico) realizadas no planejamento de releases, iterações e requisitos. Definir um procedimento para isto e referenciá-lo no Plano do Projeto.	2.2.3, 2.2.4
Definir procedimento para registro das mudanças e da realização de testes de aceitação e referenciá-lo no Plano do Projeto.	2.2.3, 2.2.4, 2.2.12
Estimar e acompanhar custos dos projetos.	2.2.6
Documentar as premissas das estimativas de esforço, custo e	2.2.8

recursos críticos computacionais.	
Planejar a gerência de configuração nos projetos, definindo: Responsabilidades sobre a gerência de configuração nos projetos Itens de configuração *Baselines* Momento de entrada dos itens de configuração nas *baselines* Procedimento para realizar mudanças aos itens de configuração e *baselines* SCCB (comitê que avalia as mudanças)	2.2.9
Documentar a arquitetura do sistema em um arquivo único, que seja referência para a equipe. Colocar este arquivo sob gerência de configuração.	2.2.10
Documentar como se darão os testes de aceitação, explicitando critérios de aceitação de testes e casos de testes.	2.2.11

Apêndice B - Template do Plano do Projeto

Template do Plano do Projeto

Versão: <xx.xx> | Data: <xx/xx/xxxx>

Conteúdo

1 Introdução

<Descreva a introdução deste documento, o que o documento apresenta, seu objetivo, escopo e metas. Além disto, em que momentos deverá ser aprovado. Exemplo abaixo>

Este documento tem como objetivo definir os aspectos essenciais para acompanhamento do projeto. Deverá ser aprovado por <incluir nomes dos responsáveis e cargos - alta gerência da organização (gerência sênior), representantes dos grupos afetados, se houverem, e representantes do cliente > no início do projeto e reaprovados nas seguintes situações:

- Desvio em 10% do cronograma ou custo do projeto
- A cada reunião de planejamento de *release*

1.1 Definições, Acrônimos e Abreviaturas

<Informe acrônimos, termos, abreviações na tabela abaixo>

Acrônimo	Descrição

1.2 Referências

[63] Documento de Requisitos, versão <xx.yy>, <dd/mm/aaaa>
[64] XPlanner do Projeto, <site>
[65] Plano de Gerência de Configuração, versão <xx.yy>, <dd/mm/aaaa>
[66] K. Beck et al., Planning Extreme Programming, Addison-Wesley, 2000.

2 Objetivo , Escopo e Metas do Projeto

<Descreva o objetivo, escopo e metas do projeto. Além disto, requisitos técnicos e não técnicos do projeto>

O objetivo deste projeto é <informe o objetivo do projeto>.

2.1 Escopo

Os requisitos detalhados do projeto e plano de entregas, estão descritos no Documento de Requisitos [63].

Os macros requisitos funcionais deste projeto são:

- < descreva os requisitos funcionais macros do sistema>

Os macros requisitos não funcionais deste projeto são:

- < descreva os requisitos não funcionais macros do sistema>

Não faz parte deste projeto:

- <descreva o escopo negativo, ou seja, o que não está contemplado por este projeto>

2.2 Critérios de Aceitação

<Descrever o critério de aceitação que será utilizado para validar que os produtos satisfazem os requisitos. Exemplos:

- Todos os testes de aceitação executados
- Todos os *deliverables* entregues
- Nenhum *bug* de severidade alta, que impede a execução do projeto
- Nenhum *bug*, o qual a estimativa de resolução seja de mais de 10 horas>

3 Ciclo de Vida do Projeto

O ciclo de vida do projeto é definido como iterativo e incremental. O projeto é dividido em marcos, chamados *releases*. Cada *release* é dividido em iterações seqüenciais, com <X> semanas de duração. O planejamento dos *releases* e iterações está documentado no Documento de Requisitos [63].

4 Organização do Projeto

Esta seção descreve a organização do projeto em termos de papéis e responsabilidades.

4.1 Organograma

<Adicione o organograma do projeto, descrevendo também o organograma do cliente, que irá interagir com o projeto>

4.2 Papéis e Responsabilidades

<Descreva os papéis e responsabilidades dos envolvidos no projeto>

4.3 Equipe

<Descreva a equipe do projeto e respectivos papéis>

5 Procedimentos, Métodos e Padrões

<Descreva todos os procedimentos, métodos e padrões adotados para desenvolver e/ou manter o software. Apontar para livros XP, Guia XP-CMM2, ... Exemplo abaixo>

Procedimento/método/padrão	Descrição	Referência
Padrão de codificação	Padrão utilizado para codificar *software*.	<indique qual o padrão, aponte para uma referência, ...>
Procedimento de Estimativa	Procedimento para realizar as estimativas de estórias e tarefas.	[66], cap. 12

6 Produtos

Esta seção descreve os produtos a serem desenvolvidos pelo projeto, indicando os produtos a serem entregues ao cliente. A gestão da configuração destes produtos é descrita no Plano de Gerência de Configuração do projeto [65].

<Insira outros produtos na tabela, conforme necessidades do projeto. Exemplo: manual do usuário, arquitetura, modelo de classes, modelo de dados, ... >

Produto	Descrição	Entregue
Plano do Projeto	Plano que descreve o planejamento	

	do projeto em termos de recursos computacionais críticos, equipe, objetivos do projeto, planejamento de instalações, entre outros. Produzido no MS Word.	
Documento de Requisitos	Descreve o planejamento de *releases* do projeto, indicando estórias (requisitos) a serem desenvolvidas por *release* e por iterações.	
Build do Sistema	*Build* do software rodando, com os requisitos descritos no Documento de Requisitos implementados.	X
Plano de Testes	Testes implementados na linguagem de desenvolvimento adotada.	X
Código fonte	Código fonte do sistema.	X

7 Cronograma

O cronograma de *releases* e iterações está descrito no Documento de Requisitos [63].

8 Custos

<Referencie as estimativas de custos do projeto e planejamento de pagamento. Além disto, comunique quaisquer compromissos relevantes relacionados aos custos do projeto.>

9 Recursos Críticos Computacionais

Esta seção descreve os recursos críticos computacionais identificados para o projeto e as estimativas realizadas. Estes recursos serão acompanhados a cada iteração através do Documento de Requisitos [63].

<Identifique os recursos críticos computacionais do projeto>

9.1 Procedimento de Estimativa de Recursos Críticos Computacionais

<Descreva o procedimento utilizado para estimar os recursos críticos computacionais ou faça referência ao documento que o descreva>

9.2 Estimativas de Recursos Críticos Computacionais

<Descreva as estimativas e premissas utilizadas para estimar os recursos críticos computacionais>

10 Instalações

<Esta seção descreve as instalações necessárias para execução do projeto>

10.1 Hardware

<Descreve o hardware necessário para execução do projeto>

Equipamento	Quantidade	Finalidade

10.2 Software

<Descreve o software necessário para execução do projeto>

Software	Quantidade	Finalidade

11 Acompanhamento do Projeto

<Esta seção descreve como o projeto irá ser acompanhado. A tabela abaixo sugere as reuniões e conteúdo conforme o Guia XP-CMM2.>

A tabela abaixo descreve as reuniões a serem realizadas no projeto e respectivos participantes e conteúdo.

Reunião	Descrição	Pauta
Stand up meetings	Reuniões diárias envolvendo toda a equipe de desenvolvimento. A ser realizada <informe horário e local da reunião>. A reunião deve durar em média 15min.	Problemas técnicos, dificuldades de implementação, problemas de configuração, pendências de garantia da qualidade, comunicação e andamento do projeto.
Reunião de planejamento da iteração	Reunião a ser realizada no primeiro dia da iteração para planejá-la. Participação de toda a equipe de desenvolvimento. Além disto, esta reunião avalia a iteração passada.	Da iteração passada: avalia métricas, estórias completadas, registro dos recursos críticos computacionais gastos, acompanhamento das estimativas. Da próxima iteração: estórias a serem implementadas, estimativas. Trata status das atividades de gerência de configuração e garantia da qualidade. Acompanhamento de riscos.
Reunião de planejamento do *release*	Reunião a ser realizada no início do desenvolvimento do próximo release para planejá-lo. Participação de toda a equipe de desenvolvimento. Participação do gerente sênior e representantes de alto nível do cliente. Além disto, esta reunião avalia o release passado.	Reavalia Plano do Projeto, riscos, Documento de Requisitos, progresso do projeto, acompanha custos, compromissos.

12 Riscos do Projeto

Esta seção referencia o documento que lista os riscos do projeto e descreve como eles serão acompanhados. A tabela abaixo relaciona os riscos identificados para o projeto.

Risco	Probabilidade[3]	Impacto[4]	Prioridade[5]	Ações de Contingência	Status[6]

[3] Probabilidade de o risco vir a ocorrer. Opções: Alta, Média ou Baixa

[4] Impacto do risco no projeto. Opções: Alta, Média ou Baixa

[5] Prioridade do risco. Deve ser baseado na probabilidade e impacto

[6] Status do risco e ações tomadas em relação a ele

Apêndice C - Template do Documento de Requisitos

O template do Documento de Requisitos foi elaborado no MS Excel em três planilhas.

A Figura 4 exibe a planilha que é a capa do Template do Documento de Requisitos.

Figura 4 - Capa do Template do Documento de Requisitos

A Figura 5 exibe a planilha que lista as estórias do projeto e para cada uma uma breve descrição, estimativa inicial, o *release* e a iteração que a estória participa, o *status* da estória (Em andamento, Não inicializada, Finalizada) e a estimativa final (complexidade da estória ao final da iteração).

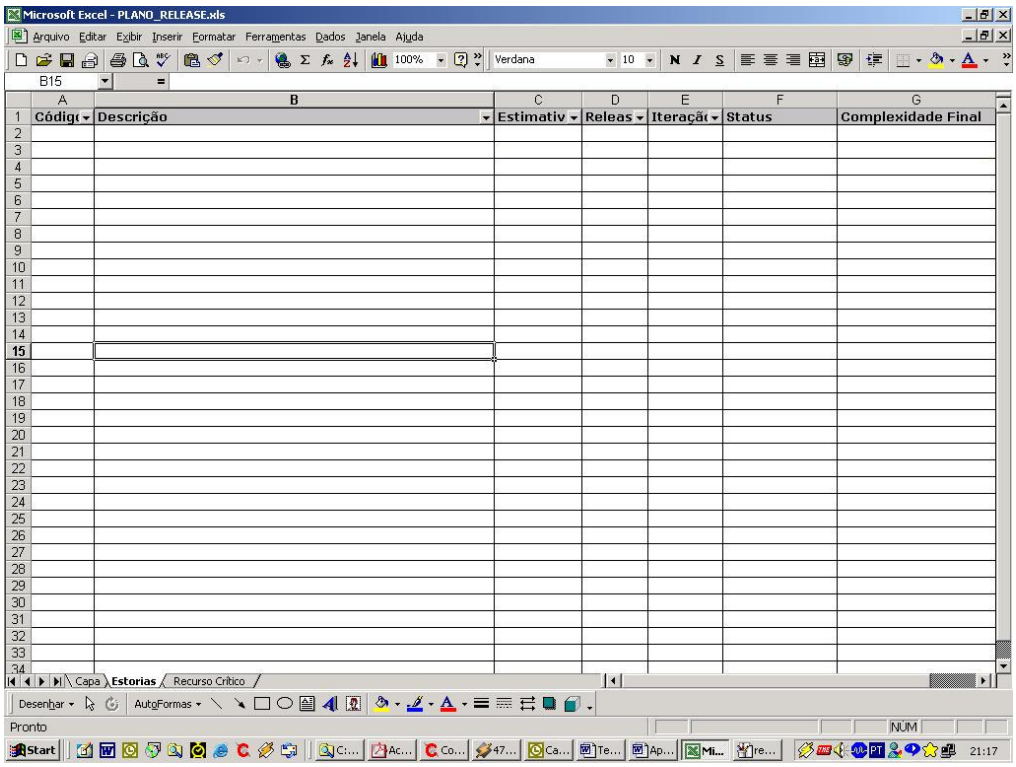

Figura 5 - Planilha Estórias do Template do Documento de Requisitos

A Figura 6 exibe o template para estimativa e acompanhamento dos recursos críticos computacionais. Para cada recurso crítico identificado, devem ser informados a estimativa inicial e o tamanho real ao final de cada iteração.

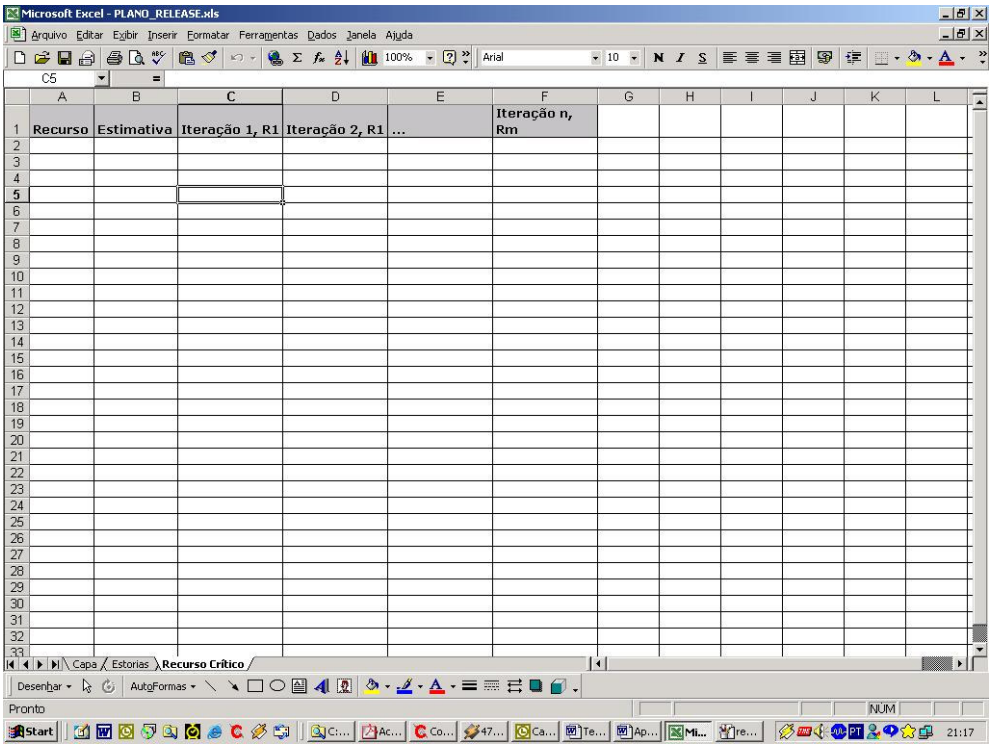

Figura 6 - Planilha Recursos Críticos do Template do Documento de Requisitos

Apêndice D - Template do Plano de Gerência de Configuração

Template do Plano de Gerência de Configuração

Versão: <xx.xx> | Data: <xx/xx/xxxx>

Conteúdo

1 Introdução

<Descreva a introdução deste documento, o que o documento apresenta, seu objetivo, escopo e metas. Além disto, em que momentos deverá ser aprovado. Exemplo abaixo>

Este documento tem como objetivo definir os aspectos essenciais para a gerência da configuração do projeto. Deverá ser aprovado por <incluir nomes dos responsáveis e cargos – gerente do projeto> no início do projeto e reaprovados a cada *release* e nas seguintes situações:

- Mudança nos itens de configuração do projeto, ferramentas e *baselines* do projeto

1.1 Definições, Acrônimos e Abreviaturas

<Informe acrônimos, termos, abreviações na tabela abaixo>

Acrônimo	Descrição
SCM	Gerência de Configuração de Software, do inglês *Software Configuration Management*
CR	Solicitação de mudança, do inglês *Change Request*
SCCB	Comitê de controle de mudanças de software, do inglês *Software Change Control Board*

1.2 Referências

[67] Plano do Projeto, versão <xx.yy>, <dd/mm/aaaa>

2 Ferramentas

<Descreva as ferramentas de configuração adotadas no projeto. A tabela abaixo sugere algumas ferramentas>

Ferramenta	Descrição
CVS – Concurrent Version System	Servidor da ferramenta de controle de versão.
WinCVS	Cliente da ferramenta de controle de versão.
Bugzilla	Ferramenta de controle de

	mudanças.

3 Baselines

<Descreva as *baselines* adotadas no projeto. Abaixo, uma sugestão>

Os itens de configuração selecionados, fazem parte de um subconjunto dos produtos relacionados no Plano do Projeto [65], que atendem a um ou mais dos seguintes critérios:

- Produtos entregues ao cliente ou a outra organização;
- Produtos considerados críticos para o projeto;
- Artefatos utilizados por mais de um grupo;
- Artefatos produzidos por terceiros, que serão incorporados ao projeto;
- Artefatos que, ao serem alterados, podem causar mudanças em outros artefatos associados.

Baseline	Label	Conteúdo (ICs)	SCCB
Iteração_X	ITERACAO_X	Documento de requisitos Código fonte Código de testes	

4 Controle de Versão

<Descreva as principais considerações a respeito do controle de versão no projeto e padrões adotados, se houverem>

4.1 Organização da Ferramenta de Controle de Versão

<Informe como a ferramenta de Controle de Versão estará organizada em termos de diretórios e seus conteúdos>

5 Controle de Mudança

<Descreva brevemente como solicitar mudanças utilizando a ferramenta de controle de mudanças e os padrões adotados, se houverem >

6 Releases

<Descreva o planejamento de *releases*. Indique os procedimentos para realização do *release.* "Exemplo: Os releases do projeto serão realizados através do envio por e-mail de um arquivo compactado, .zip". Abaixo, uma sugestão>

Os *releases* do projeto serão realizados de acordo com a tabela abaixo:

Release	Conteúdo
Release Número <X>	<Informe os itens de configuração que deverão fazer parte do *release*>

7 Auditorias de Gerência de Configuração

<Descreva brevemente como as auditorias de gerência de configuração serão realizadas no projeto. Abaixo, uma sugestão>

Auditorias em gerência de configuração serão realizados no período de <informar o período das auditorias>. A cada auditoria um relatório será gerado e enviado para todos da equipe através de e-mail. Um plano de ação para resolver os problemas identificados será elaborado em conjunto com o gerente do projeto. O acompanhamento do plano de ação é de responsabilidade de <informar o responsável>

8 Relatórios de Gerência de Configuração

<Descreva os relatórios providos pela gerência de configuração. Indique conteúdo, público alvo, periodicidade. Abaixo, uma sugestão>

Relatório	Descrição	Periodicidade	Público alvo
Status das CRs	Obtido a partir de consulta ao *bugzilla*, relaciona quantidade de CRs abertas, fechadas e pendentes	A cada iteração	Todos da equipe
Relatórios de auditoria	Relatório descrevendo os problemas identificados na gestão de configuração	A cada <informe o período>	Todos da equipe
Plano de	Relatório descrevendo o	A cada	Todos da

ação de auditoria	*status* das ações do Plano de Ação das auditorias de gerência de configuração	iteração	equipe