Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

Padronização da Documentação do Processo de Desenvolvimento de Software de Micro e Pequenas Empresas Utilizando o Método Scrum

Ricardo Camargo dos Santos

TCC-EP-102-2013

Maringá - Paraná
Brasil

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

Padronização da Documentação do Processo de Desenvolvimento de Software de Micro e Pequenas Empresas Utilizando o Método Scrum

Ricardo Camargo dos Santos

TCC-EP-104-2013

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Produção, do Centro de Tecnologia, da Universidade Estadual de Maringá.
Orientadora: *Prof(ª). Dra. Tânia F. Calvi Tait*

Maringá - Paraná
2013

DEDICATÓRIA

Dedico essa conquista aos meus pais, José Maria e Maria, e aos meus irmãos Renato e Rafaela, que são as pessoas que são a base de tudo que faço.

Epígrafe

“Do or do not, there is no try.”

Yoda.

AGRADECIMENTOS

Aos meus pais José Maria e Maria, sem a o apoio deles eu não teria chegado até aqui.

Aos meus irmãos Renato e Rafaela, que sempre me dão conselhos nas horas mais difíceis.

A minha professora e orientadora Tânia, que me orientou durante o ano de 2013 inteiro, sem ela o desenvolvimento desse trabalho não teria sido possível.

Aos meus amigos Vitor, Carlos, Mateus, Bruna e Oliver, que me acompanharam nesses 5 anos de faculdade, estudando para provas e realizando trabalhos.

A minha namorada Gabriela, que sempre me apoiou e incentivou em todos os momentos que precisei.

Aos meus companheiros da República Lanterna Verde, Vitor e Henrique, que conviveram comigo nesse ano de 2013 em que passamos bons e maus momentos juntos.

RESUMO

O mercado de desenvolvimento de software está cada vez mais competitivo, as micro e pequenas empresas passam cada vez mais dificuldades para competir com empresas de grande porte. Isso porque as grandes empresas, normalmente, possuem métodos padrão para desenvolvimento de software, além de uma grande quantidade de documentação gerada durante o desenvolvimento do mesmo. Neste trabalho estudou-se as micro e pequenas empresas, os métodos de desenvolvimento de software e documentação do processo de desenvolvimento. Após estudo de caso de uma empresa, definiu-se o método a ser utilizado para o desenvolvimento de software e definiu-se a documentação padrão a ser utilizada durante este processo. O método é composto pelas etapas pré-planejamento, desenvolvimento e pós-desenvolvimento.

Palavras-chave: métodos de desenvolvimento de software, micro e pequenas empresas, documentação de software.

Abstract

The software's development market is more and more competitive, as micro and small companies go through more and more difficulties to compete with bigger companies. That's because the bigger ones, usually, own standard methods of software production and also own a great amount of documentation produced in th e process. In this work, it was studied micro and small companies, the methods of the software's process of production and the documentation of this process. After a company's case study, it was defined the method to be used to the production of software and also the standard documentation to be used during the process. The method is composed by the following steps: pre planning, development and post development.

Keywords: software's development, micro and small companies, software's documentation.

Sumário

1.1. Justificativa 2
1.2. Definição e delimitação do problema 2
1.3. Objetivos 2
1.3.1. Objetivo geral 2
1.3.2. Objetivos específicos 2
2. REVISÃO DE LITERATURA 3
2.1. Padronização 3
2.2. Padrão de Qualidade 3
2.2.1. Modelo CMMI 3
2.2.2. Modelo MPS.BR 5
2.3. Documentação do desenvolvimento de software 6
2.4. Desenvolvimento de Software 7
2.4.1. Engenharia de software 8
2.4.1.1. Modelo Cascata 10
2.4.1.2. Prototipação 12
2.4.1.3. Modelo Espiral 14
2.5.1. Extreme Programing(XP) 16
2.6. Metodologia Scrum 21
2.7. Comparação Entre os Métodos de Desenvolvimento de Software 22
2.8. Micro e Pequena Empresa 25
3. METODOLOGIA 27
4. DESENVOLVIMENTO 29
4.1. Contexto da Empresa 29
4.2. Documentação de Software 29
4.2.1. Documentação de Requisito 30
4.2.2. Documentação do Plano de Projeto 30
4.2.3. Documentação da Solução Técnica 30
4.2.4. Documentação do Check-list de Verificação do Código Fonte 31
4.2.5. Documentação do Sumário de Teste 31
4.2.6. Documentação de Release Notes 31
4.3. Problemas Encontrados no Processo de Documentação do Desenvolvimento de Softwares e Soluções Propostas 35
4.4. Proposta de Padronização da Documentação de Software Utilizando a Metodologia SCRUM 35
4.5. Plano de Atividades e Documentações para o Desenvolvimento do Software 36
4.6. Contribuições Esperadas 39
5. Considerações Finais 40
5.1. Limitações 40
5.2. Trabalhos futuros 40
6. REFERÊNCIAS 41

LISTA DE ILUSTRAÇÕES

Figura 1 - O ciclo de vida do modelo cascata (adaptado SOMMERVILLE, 2007).....11
Figura 2 – Prototipação (PRESSMAN, 2006).....13
Figura 3 – Modelo em espiral do processo de software de Boehm (©IEEE, 1998) 15
Figura 4 - Metodologia XP (WELLS, 2002).....20
Figura 5 - Etapas Scrum (MOUNTAIN GOAT SOFTWARE, 2005) 22
Figura 6 - Proposta de Padronização da Documentação de Software Utilizando a Metodologia SCRUM 39

LISTA DE QUADROS

Quadro 1 - Níveis do CMMI 5
Quadro 2 - Comparação dos Métodos de Desenvolvimento de Software 23
Quadro 3 - Documentos de Software 32
Quadro 4 - Problemas Encontrados no Processo de Documentação do Desenvolvimento de Softwares e Soluções Propostas 35

LISTA DE ABREVIATURAS E SIGLAS

CMMI	*Capability Maturity Model Integration*
MPE	Micro e Pequenas Empresas
WBS	*Work Breakdown Structure*
MPS.BR	Melhoria de Processo do Software Brasileiro
EPP	Empresa de Pequeno Porte
PIB	Produto Interno Bruto
CASE	*Computer Aided Software Engineering*
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
XP	*Extreme Programing*

1. INTRODUÇÃO

As micro e pequenas empresas (MPEs) representam os principais pilares da economia brasileira, seja pela grande capacidade de geração de empregos, ou pelo grande número de MPEs distribuídas pelo território brasileiro.

Em termos numéricos, segundo KOTENSKI (2005, apud PEREIRA et al, 2010), as MPEs representam 25% do Produto Interno Bruto (PIB) do país, geram aproximadamente 14 milhões de empregos, ou seja, 60% dos empregos formais, e representam 99% dos 6 milhões de estabelecimentos formais existentes. As Mpes representam ainda 99,8% das empresas que são criadas a cada ano.

Apesar de representarem boa parte da economia brasileira, as MPEs enfrentam muitas dificuldades e por isso tem um alto índice de mortalidade nos primeiros anos. Com as MPEs da área de software não é diferente, enfrentam todas as dificuldades das demais, no entanto, ao estarem situadas na área de software, onde as inovações tecnológicas são cada vez mais rápidas e caras, as MPEs enfrentam desafios cada vez maiores para se manterem competitivas.

Uma das principais falhas das MPEs de software é que poucas ou nenhuma parte do projeto é documentada. Alguns dos problemas da falta de documentação adequada são: dificuldade para a atualização e manutenção, dificuldades de acesso do usuário e alto custo para o software (PEREIRA et al, 2010).

A documentação no processo de desenvolvimento de software já é muito utilizada nas empresas de maior porte. Segundo AMBLER (2001, apud PEREIRA et al, 2010) há duas razões básicas para documentar: auxiliar a comunicação durante o desenvolvimento do software e auxiliar o entendimento nas atividades de manutenção e atualização quando se fizerem necessárias.

Além disso, uma documentação bem feita durante o desenvolvimento de software auxiliará a empresa a manter os prazos de entrega propostos, fator essencial para a sobrevivência da organização.

Este trabalho analisa o método de documentação da empresa Software Maringá, e assim propõe um processo de documentação para MPEs, mostrando para elas as vantagens e a importância de uma documentação bem feita ao longo do desenvolvimento do software.

1.1. Justificativa

O presente trabalho foi realizado para que o processo de desenvolvimento de software de micro e pequenas empresas seja documentado de forma padrão.

A finalidade do estudo é mostrar para as MPEs que elas também devem documentar o desenvolvimento de softwares, abandonando a ideia de que elas não tem tempo para isso e que a documentação só é necessária para empresas de grande porte.

A escolha de MPEs devido a sua grande representação na economia brasileira e devido a sua precariedade no desenvolvimento de softwares.

1.2. Definição e delimitação do problema

Através do estudo de cada etapa do desenvolvimento de software das MPEs, foi determinada uma documentação padrão necessária para cada uma dessas fases.

A finalidade do estudo é implantar melhorias no processo de desenvolvimento de software das MPEs, deixando claro que elas são capazes de documentar o processo de desenvolvimento de software e que com isso se tornarão mais competitivas no mercado.

Além de mostrar que também podem fazer essa documentação, também são destacadas as melhorias trazidas por esse padrão de documentação.

1.3. Objetivos

1.3.1. Objetivo geral

Apresentar um modelo padrão para a documentação do desenvolvimento de software para micro e pequenas empresas.

1.3.2. Objetivos específicos

- Definir os recursos e atividades necessárias para a documentação;
- Levantar os possíveis benefícios trazidos pela documentação do desenvolvimento do software, como melhoria da qualidade do software, redução de custos, maior confiabilidade, redução de tempo de projeto e etc.

2. REVISÃO DE LITERATURA

2.1. Padronização

Segundo Megen (2002) a padronização é o caminho seguro para a produtividade e competitividade, pois é uma das premissas onde se assenta o moderno gerenciamento. É obtida em sua grande parte, de forma voluntária, e consiste de uma atividade sistemática de estabelecer, por intermédio de discussões entre pessoas, o procedimento mais adequado, definindo-o como padrão a ser cumprido.

Podemos definir de forma mais técnica a padronização como o método de operação mais eficiente possível, ou seja, evitar desperdícios e utilizar insumos como materiais, trabalhadores e máquinas, de forma racional.

Na maior parte das micro e pequenas empresas, os processos não estão padronizados. Se for um processo executado por vários colaboradores cada um faz do seu jeito, se for executado por apenas um, a maneira de fazer só está clara para ele, fica registrada apenas em sua memória e nada mais (PEREIRA et al, 2010).

Segundo (IMAI 1996, apud ALVES, 2011) além de garantir a qualidade, a padronização do trabalho é a forma mais eficaz de executar o trabalho.

Para Meegen (2002) na busca da qualidade, a padronização é uma ferramenta gerencial que possibilita a transmissão de informações e dos conhecimentos adquiridos. Através da utilização contínua de padrões estabelecidos, ela assegura a performance dos sistemas produtivos, seus processos e operações, permitindo uma maior produtividade e qualidade.

A padronização pode trazer vários benefícios para a empresa, como por exemplo, preservação do conhecimento de como fazer, estabilidade dos processos, certeza de previsibilidade, base de treinamento e base para melhoria.

2.2. Padrão de Qualidade

2.2.1. Modelo CMMI

O CMMI (*Capability Maturity Model Integration*) consiste das melhores práticas relativas às atividades de desenvolvimento e manutenção aplicadas a produtos e serviços. Ele possui práticas que cobrem o ciclo de vida do produto desde a concepção até a entrega e manutenção, e se concentra no trabalho necessário para construção e manutenção do produto em sua totalidade (Guia CMMI para Desenvolvimento Versão 1.2, 2006).

Os CMMI foca na melhoria de processo em uma organização. Contendo os elementos essenciais de processos efetivos o CMMI descreve um caminho de melhoria evolutiva desde processos imaturos, ou ad hoc, até processos maduros, disciplinados, com qualidade e eficácia melhoradas.

O modelo CMMI utiliza-se de níveis para descrever um caminho evolutivo recomendado para uma organização que deseja melhorar os processos utilizados para desenvolver e manter seus produtos e serviços. Os níveis também podem resultar de classificações obtidas por meio de avaliações 7 realizadas em organizações compreendendo a empresa toda (normalmente pequenas), ou grupos menores, tais como um grupo de projetos ou uma divisão de uma empresa.

O CMMI apresenta dois caminhos para melhoria. Um caminho permite que as organizações melhorem de forma incremental os processos correspondentes a uma ou mais áreas de processos individualmente selecionadas pela organização (representação contínua). O outro caminho permite que as organizações melhorem um conjunto de processos inter-relacionados e, de forma incremental, tratem sucessivos conjuntos de áreas de processo (representação por estágios).

Para a representação contínua, emprega-se a expressão “nível de capacidade” e para a representação por estágios, emprega-se a expressão “nível de maturidade”.

Independentemente da representação escolhida, o conceito de níveis é o mesmo. Os níveis caracterizam melhorias a partir de um estado em que processos estão mal definidos em direção a um estado que utilize informações quantitativas a fim de determinar e gerenciar melhorias necessárias para satisfazer aos objetivos estratégicos da organização.

Para alcançar um determinado nível, uma organização deve satisfazer a todas as metas associadas à área de processo ou ao conjunto de áreas de processos que constituem o alvo para melhoria, independentemente de se tratar de um nível de capacidade ou de um nível de maturidade.

Essas dimensões do CMMI, capacidade e maturidade, são utilizadas tanto para benchmarking e atividades de avaliação, quanto para orientar os esforços de melhoria da organização. · Níveis de capacidade, associados à representação contínua, aplicam-se à melhoria de processo da organização em áreas de processos individuais. Esses níveis são um meio para melhorar,

de forma incremental, os processos correspondentes a uma determinada área de processo. Há seis níveis de capacidade, numerados de 0 a 5.

Níveis de maturidade, associados à representação por estágios, aplicam-se à melhoria de processo da organização em um conjunto de áreas de processo. Esses níveis auxiliam na previsão dos resultados de futuros projetos. Há cinco níveis de maturidade, numerados de 1 a 5.

Nível	Representação Contínua	Representação por Estágios
0	Incompleto	Não se aplica
1	Executado	Inicial
2	Gerenciado	Gerenciado
3	Definido	Definido
4	Gerenciado Quantitativamente	Gerenciado Quantitativamente
5	Em otimização	Em otimização

Quadro 1 - Níveis do CMMI

Existem algumas diferenças entre os níveis das duas representações, mas para as duas representações os níveis irão determinar o grau de padronização, controle, indicadores de desempenho e organização de um processo.

2.2.2. Modelo MPS.BR

O programa MPS.BR (Melhoria de Processo do Software Brasileiro) tem como uma de suas metas, definir e aprimorar um modelo de melhoria e avaliação de processo de software e serviços, tendo como foco às micro, pequenas e médias empresas. A maior parte das MPE's não possui nenhum tipo de certificação e o modelo CMMI acaba não se adequando as suas necessidades, assim o MPS.BR voltou-se para atender as necessidades dessas empresas e tem como foco ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software e serviços (Guia Geral MPS de Software, 2012).

O MPS.BR é dividido em quatro componentes: Modelo de Referências MPS para Software (MR-MPS-SW), Modelo de Referências MPS para Serviços (MR-MPS-SV), Método de Avaliação (MA-MPS) e Modelo de Negócios (MN-MPS). Além desses quatro componentes a MPS.BR possui dois guias para as organizações, sendo eles: Guia de Aquisição e Guia de Implementação. O Guia de Aquisição fornece boas práticas para organizações que pretendem adquirir um software ou serviço. Já o guia de Implementação sugere formas de implementar cada um dos níveis da MR-MPS-SW.

O Modelo de Referências MPS para Software contém os requisitos que os processos das organizações devem atender para estar em conformidade com o MR-MPS-SW. Este modelo contém as definições de níveis de maturidade, processos e atributos do processo. Os níveis de maturidade do MR-MPS-SW são divididos em nove, sendo eles: G (Parcialmente Gerenciado), F (Gerenciado), E (Parcialmente Definido), D (Largamente Definido), C (Definido), B (Gerenciado Quantitativamente), A (Em Otimização). A escala de maturidade se inicia no nível G e progride até o nível A. Cada um dos níveis de maturidade possui processos que indicam onde a organização deve colocar o esforço de melhoria. Para se alcançar um determinado nível de maturidade é necessário que os processos atendam aos requisitos e propósitos daquele nível. (Guia Geral MPS de Software, 2012)

O Modelo de Referências MPS para Serviços contém os requisitos que os processos das organizações devem atender para estar em conformidade com o MR-MPS-SV. Este modelo contém as definições de níveis de maturidade, processos e atributos do processo. Os níveis de maturidade do MR-MPS-SV estão em conformidade com os requisitos de modelos de referência de processo da Norma Internacional ISO/IEC 15504-2.

O Modelo de Negócio MN-MPS descreve regras de negócios para implementação do MR-MPS-SW e MR-MPS-SV pelas Instituições Implementadoras e regras de avaliação utilizadas pelas Instituições Avaliadoras.

O Guia de Avaliação contém o processo e o método de avaliação MA-MPS, os requisitos para os avaliadores e Instituições Avaliadoras. O processo e o método de avaliação MA-MPS estão em conformidade com a Norma Internacional ISO/IEC 15504-2.

2.3. Documentação do desenvolvimento de software

A documentação do desenvolvimento de software pode ser definida como um conjunto de manuais gerais e técnicos, podendo ser organizado em forma de textos e comentários, utilizando ferramentas do tipo dicionários, diagramas e fluxogramas, gráficos, desenhos, dentre outros.

Documentar o desenvolvimento de software é uma atividade essencial desse processo. É através da documentação que a evolução do software é registrada. Com isso criam-se bases necessárias para as etapas posteriores do processo de software, como por exemplo, treinamento, utilização e manutenção do software (COELHO ,2009).

Diversos tipos de documentos são gerados ao longo do processo de software, tais como propostas, planos de projetos, especificações de requisitos e de projeto, dentre outros. Por isso é necessário que a organização tenha que desprender um tempo precioso na documentação, designando pessoas habilitadas especificamente para esta tarefa e escolhendo as ferramentas que possibilitem a documentação de cada etapa do processo.

Devido a essa quantidade de tempo e de recursos que é utilizada pela documentação, muitas empresas documentam após o término do projeto, ou a fazem de forma parcial, ou ainda como na maioria dos casos de micro e pequenas empresas, não documentam. Essas empresas focam apenas na qualidade do software em si, mas não enxergam que uma documentação de qualidade gera um aumento na qualidade do software (PEREIRA et al, 2010).

Mas para que essa documentação seja de qualidade, ela deve ser feita durante o processo de desenvolvimento do software, possibilitando assim que atividades como avaliação e manutenção do software possam ser realizadas com sucesso. Problemas como, custos do projeto, dificuldades de manutenção e imprecisão dos serviços fornecidos, podem ser sanados com a utilização da documentação.

2.4. Desenvolvimento de Software

Howard Baetjer JR (apud Pressman, 2006) fez os seguintes comentários sobre o processo de desenvolvimento de software:

> Desde que o software, como todo capital, é conhecimento incorporado, e como esse conhecimento está incialmente disperso, tácito, latente e incompleto na sua totalidade, o desenvolvimento de software é um processo de aprendizado social. O processo é um diálogo no qual o conhecimento, que deve se transformar em software, é reunido e incorporado ao software. O processo fornece interação entre usuários e projetistas, entre usuários e ferramentas em desenvolvimento e entre projetistas e ferramentas em desenvolvimento (tecnologia). É um processo iterativo no qual a própria ferramenta serve como meio de comunicação, com cada nova rodada de diálogo explicitando mais conhecimento útil do pessoal envolvido (PRESSMAN, 2006,P. 16).

De maneira básica, o processo de software pode ser definido como um roteiro que deve ser seguido para criar um produto de alta qualidade dentro de um prazo.

Segundo Pressman (2006), do ponto de vista técnico, o processo de software é definido como um arcabouço para as tarefas que são necessárias para construir softwares de alta qualidade.

Dentro do desenvolvimento de software, não utilizamos apenas o processo de software, utilizamos métodos técnicos e ferramentas automatizadas para desenvolver o software, entrando então no conceito de engenharia de software.

Segundo (BAUER, apud PRESSMAN, 2006) engenharia de software é a criação e a utilização de sólidos princípios de engenharia afim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais.

Pressman (2006) define o seguinte arcabouço de processo genérico para o desenvolvimento de software. Esse arcabouço é aplicável à grande maioria dos projetos de software:

- Comunicação: Essa atividade abrange o levantamento de requisitos e outras atividades relacionadas, mas para isso é necessário uma alta comunicação e colaboração com o cliente.
- Planejamento: Essa atividade irá estabelecer um plano para o trabalho de engenharia de software que se segue. Deve descrever as tarefas a serem realizadas, os riscos, os recursos necessários, os produtos de trabalho a serem produzidos de acordo com o cronograma.
- Modelagem: Essa atividade inclui a criação de modelos que irão auxiliar o cliente e o desenvolvedor a entender melhor os requisitos do software e o projeto que vai satisfazer esses requisitos.
- Construção: Combina a geração de código e os testes necessários para revelar erros no código.
- Implantação: O software, seja esse completo ou parcial, é entregue ao cliente, que avalia o produto entregue e fornece um feedback.

Essas cinco atividades podem ser usadas desde o desenvolvimento de softwares pequenos até o desenvolvimento de grandes softwares. O que irá mudar serão os detalhes do processo de software, mas as atividades de arcabouço permanecem as mesmas.

2.4.1. Engenharia de software

Pressman (2006) define a engenharia de software como um conjunto de etapas que envolve métodos, ferramentas e os procedimentos básicos para o desenvolvimento de software. Essas

etapas muitas vezes são chamadas de paradigmas da engenharia de software. Para Sommerville (2007) a engenharia de software, ou engenharia de sistemas, é a atividade de especificação, projeto, implementação, validação, implantação e manutenção do software.

De acordo com Pressman (2006) os métodos de engenharia de software proporcionam os detalhes de "como fazer" para construir o software. Segundo Sommerville (2007) os métodos de engenharia de software, ou processos de software, é um conjunto de atividades que leva a produção de um produto de software. Essas atividades podem envolver o desenvolvimento de software propriamente dito, utilizando uma linguagem de programação qualquer, ou desenvolver um software através da modificação ou ampliação de um software já existente. Para Pressman (2006) os métodos de software proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem planejamento e estimativa de projeto, análise de requisito de software e de sistemas, projeto de estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção. Os métodos de desenvolvimento geralmente introduzem uma notação gráfica ou orientada a linguagem especial e introduzem um conjunto de critérios para a qualidade do software.

Segundo Pressman (2006) as ferramentas de engenharia de software proporcionam apoio automatizado ou semi-automatizado aos métodos. Atualmente existem ferramentas de para apoiar cada um dos métodos de engenharia de software. A ferramenta mais utilizada para apoiar esses métodos é a ferramenta CASE (*Computer Aided Software Engineering*). A ferramenta CASE consiste em um conjunto de ferramentas integradas, de forma que a informação criada por uma ferramenta possa ser utilizada por outra.

Os procedimentos da engenharia de software são como um elo de ligação que mantém juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador. Os procedimentos definem a sequência em que os métodos serão aplicados, os produtos que se exigem que sejam entregues (documentos, relatórios, formulários, etc.) os controles que ajudam a coordenar as mudanças, e os marcos de referência que possibilitam aos gerentes de software avaliar o progresso.

A engenharia de software compreende um conjunto de etapas que envolvem métodos, ferramentas e procedimentos. Pressman (2006) cita essas etapas como paradigmas de engenharia de software. A escolha desse paradigma é feita com base na natureza do projeto e

da aplicação, os métodos e as ferramentas a serem usados, os controles e os produtos que precisam ser entregues. Sommerville (2007) define esse conjunto de etapas como modelos de processo de software. Um modelo de processo de software é uma representação abstrata de um processo de software, que pode ser usado para explicar diferentes abordagens para o desenvolvimento de software. Os modelos de processo de software, ou paradigmas de engenharia de software são: Modelo Cascata, Prototipação, Modelo Espiral.

2.4.1.1. Modelo Cascata

O primeiro modelo de processo de desenvolvimento de software publicado originou-se de processos de engenharia de sistema (ROYCE 1970, apud SOMMERVILLE 2007). Esse modelo requer uma abordagem sistemática, sequencial ao desenvolvimento de software, que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção. Devido a esse encadeamento de uma fase com a outra esse modelo recebe o nome de cascata. Esse modelo abrange as seguintes atividades:

- Análise e definição de requisitos: Os serviços, restrições e objetivos do sistema são definidos através de consultas aos usuários do sistema. Esses três itens devem ser definidos detalhadamente para que se entenda melhor o software a ser desenvolvido, essas informações levantadas com os usuários servem como uma especificação de sistema.
- Projeto de sistema e software: Os requisitos são divididos em requisitos de sistema de hardware ou de software. Os requisitos de software possibilitam ao analista entender a natureza do programa a ser construído, além de possibilitar o entendimento do domínio do problema esses requisitos devem conter informações como desempenho do software e interfaces exigidas. Os requisitos de hardware são os requisitos necessários para rodar o sistema com excelência.
- Implementação e teste de unidade: O projeto deve ser traduzido numa forma legível por máquina. Essa etapa executa essa tarefa, os códigos são gerados por unidades de programa. Essas unidades serão testadas para verificar se cada unidade atende à sua especificação.
- Integração e teste de sistema: As unidades individuais de programa são integradas e testadas como um sistema completo para garantir que os requisitos de software foram atendidos. Após os testes o sistema de software é liberado para o cliente.
- Operação e manutenção: O sistema é instalado e colocado em operação. A manutenção envolve correção de erros não detectados nos estágios anteriores do ciclo de vida, no

aprimoramento da implementação das unidades de sistema e na ampliação de serviços de sistema à medida que novos requisitos são identificados

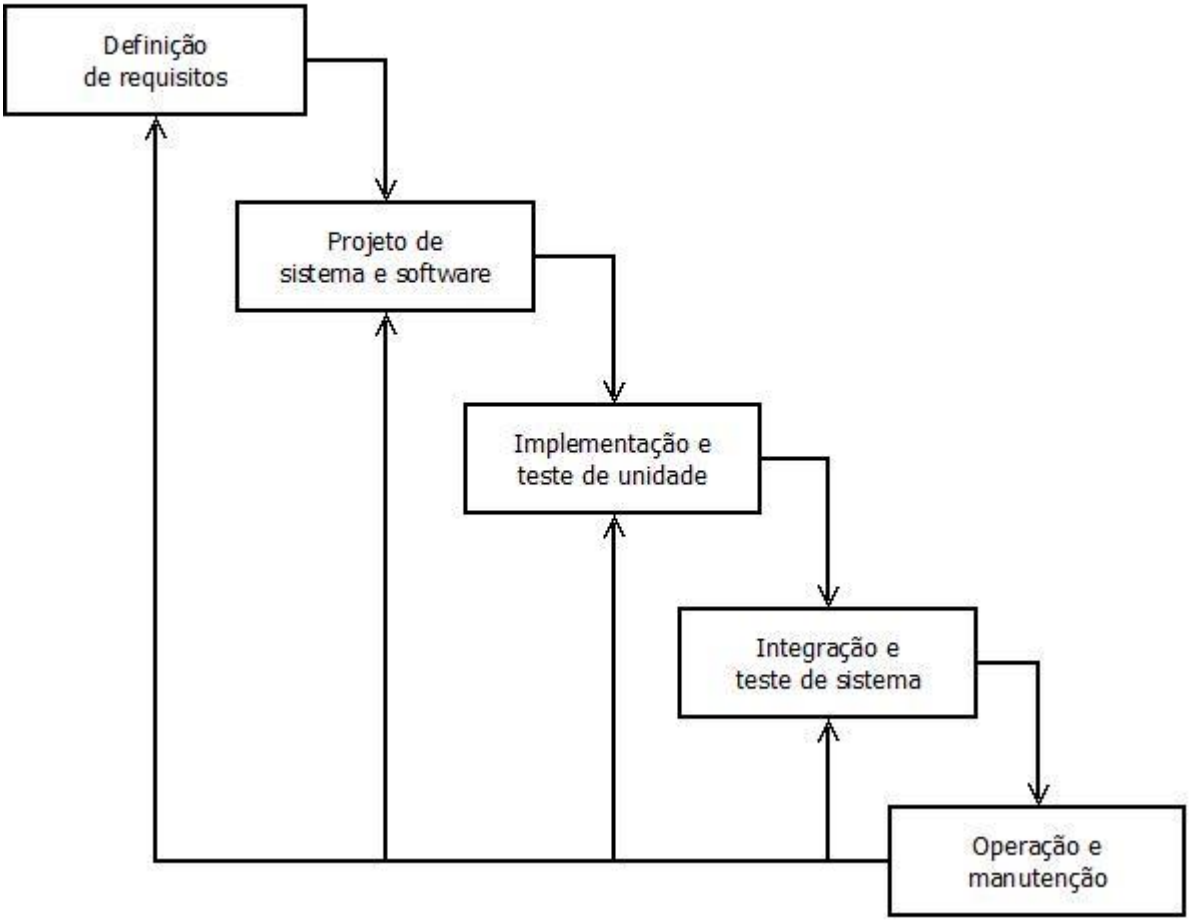

Figura 1 - O ciclo de vida do modelo cascata (adaptado SOMMERVILLE, 2007)

Na teoria, o resultado de cada fase consiste de um ou mais documentos que devem ser aprovados. Esse ciclo de vida é o mais antigo e o mais amplamente usado na engenharia de software, porém segundo Pressman (2007) no decorrer da última década, as críticas ao ciclo fizeram com que até mesmo seus ativos defensores questionassem sua aplicabilidade em todas as situações. Alguns dos problemas desse ciclo de vida são:

- Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Alguma iteração sempre ocorre e traz problemas na aplicação do ciclo.

- Muitas vezes é difícil para o cliente declarar todas as exigências explicitamente. O ciclo de vida cascata exige isso e tem dificuldade de acomodar a incerteza natural que existe no começo dos projetos.

- O cliente deve ter paciência. Uma versão de trabalho do programa não estará disponível até um ponto tardio do cronograma do projeto. Um erro grosseiro, se não for detectado até que o programa de trabalho seja revisto, pode ser desastroso.

As vantagens do modelo em cascata consistem na documentação produzida em cada fase e sua base para os outros modelos de desenvolvimento de software.

2.4.1.2. Prototipação

Segundo Pressman(2006) a prototipação é um processo que capacita o desenvolvedor a criar um modelo de software que será implementado. O modelo pode assumir uma das três formas:

- Um protótipo em papel ou baseado em modelo PC que retrata a interação entre homem-máquina de uma forma que capacite o usuário a entender a forma que a interação ocorrerá.

- Um protótipo de trabalho que implementa algum subconjunto da função exigida do software desejado.

- Um programa existente que executa parte ou toda função desejada, mas que tem outras características que serão melhoradas em um novo esforço de desenvolvimento.

Figura 2 – Prototipação (PRESSMAN, 2006)

A Figura 2 representa a sequência de eventos para a prototipação. A primeira tarefa como nas outras abordagens é a de coleta de requisitos. Nessa etapa o cliente e o desenvolvedor reúnem-se para definir os objetivos globais para o software, identificam as exigências conhecidas e esboçam as áreas que uma definição adicional é obrigatória. Ocorre então a elaboração de um "projeto rápido" O projeto rápido concentra-se na representação daqueles aspectos do softwares que serão visíveis ao usuário. O projeto rápido leva à construção de um protótipo que é avaliado pelo cliente/usuário e é usado para refinar os requisitos para o software a ser desenvolvido. O protótipo ao mesmo tempo é capaz de satisfazer as necessidades do cliente e mostrar ao desenvolvedor aquilo que precisa ser feito.

Assim como os outros modelos, a prototipação também possui pontos negativos, sendo eles:

- O cliente vê aquilo que parece ser uma versão de trabalho do software, sem saber que aquela é apenas uma demonstração para que o desenvolvedor entenda melhor as necessidades do cliente, e acaba exigindo alguns acertos sejam aplicados no protótipo para torna-lo usável. E muitas vezes a gerência de desenvolvimento do software cede.

- Devido a essas exigências de acertos no protótipo para que o cliente possa utilizá-lo no trabalho, muitas vezes o desenvolvedor acaba implementando linguagem de programação ou sistema operacional inadequado, o que pode levar o desenvolvedor a familiarizar-se com essas opções e esquecer o porquê de elas serem inadequadas.

Ainda que possam ocorrer problemas a prototipação é um paradigma eficiente da engenharia de software. Desenvolvedor e cliente devem concordar que o protótipo será construído apenas como um mecanismo de definir requisitos.

2.4.1.3. Modelo Espiral

Segundo Sommerville (2007) o modelo em espiral do processo de software foi originalmente proposto por Boehm. Em vez de representar o processo de software como uma sequência de atividades com algum retorno entre uma atividade e outra, o processo é representado como uma espiral. Cada loop na espiral representa uma fase do processo de software. Dessa forma o loop mais interno pode estar relacionado à viabilidade do sistema; o próximo loop, à definição de requisitos; o próximo ao projeto de sistema e assim por diante.

Cada “loop” da espiral está dividido em quatro setores:

- Definição de objetivos: Os objetivos específicos dessa fase do projeto são definidos. As restrições sobre o processo e o produto são identificadas e um plano detalhado de gerenciamento é elaborado. Os riscos de projeto são identificados. Dependendo desses riscos, estratégias alternativas podem ser planejadas.

- Avaliação e redução de riscos: Para cada risco de projeto identificado, uma análise detalhada é realizada. Providências são tomadas para reduzir o risco. Por exemplo, se houver risco de que os requisitos não sejam apropriados, um protótipo do sistema deve ser desenvolvido.

- Desenvolvimento e validação: Após a validação de risco, um modelo de desenvolvimento par ao sistema é selecionado. Por exemplo, se os riscos da interface com o usuário forem dominantes, um modelo de desenvolvimento apropriado pode ser a prototipação.

- Planejamento: O projeto é revisado e uma decisão é tomada para prosseguimento ao próximo loop da espiral. Se a decisão for pelo prosseguimento, serão elaborados planos para a próxima fase do projeto.

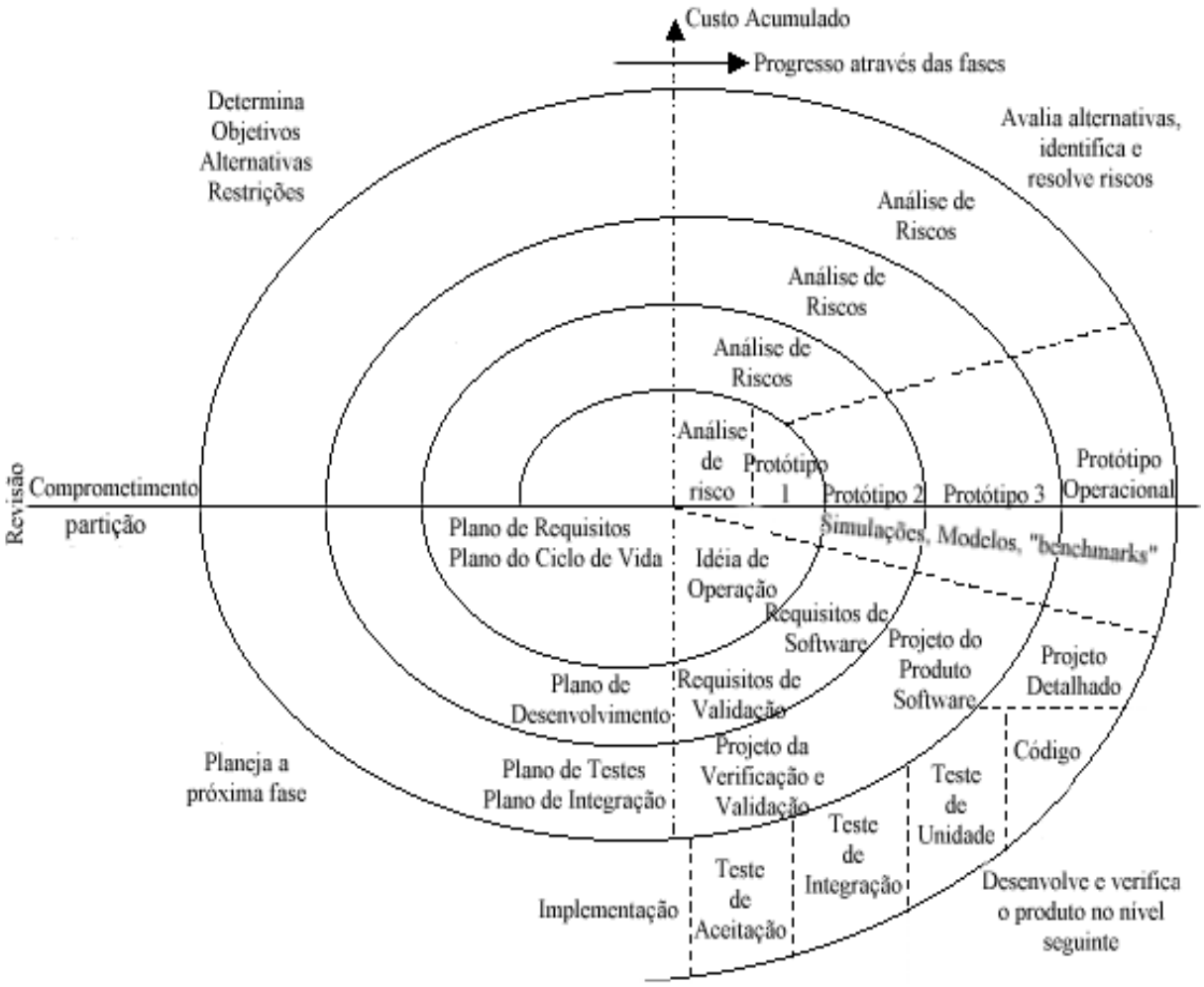

Figura 3 – Modelo em espiral do processo de software de Boehm (©IEEE, 1998)

A principal diferença entre o modelo em espiral e os outros modelos do processo de software é o reconhecimento explícito do risco no modelo espiral. Sommerville (2007) define risco como algo que pode dar errado. Os riscos podem causar problemas no projeto, tal como ultrapassar o cronograma e os custos, por isso, a minimização dos riscos é uma atividade de gerenciamento de projeto muito importante.

O ciclo da espiral começa com a elaboração dos objetivos, como desempenho e funcionalidade. Caminhos alternativos para alcançar esses objetivos são propostos, respeitando sempre às restrições do sistema. Cada alternativa é avaliada em relação a cada

objetivo e as fontes de risco do projeto são identificadas. O próximo passo é resolver esses riscos por meio de atividades de coleta de informações, tais como análise mais detalhada, prototipação e simulação. Após a avaliação dos riscos, é realizada uma parte do desenvolvimento, seguida pela atividade de planejamento para a próxima fase do processo.

2.5. Métodos Ágeis de Desenvolvimento de Software

As metodologias ágeis para desenvolvimento de software são uma resposta às chamadas metodologias tradicionais. Mesmo com a evolução das técnicas, dos computadores e das ferramentas nos últimos anos, a produção de software confiável, correto e entregue dentro do prazo e dos custos estipulados é muito difícil. Por esta razão, muitas organizações, principalmente as de pequeno porte, acabam por não usar nenhum processo, o que pode levar a efeitos desastrosos em termos de qualidade de software. Com isso essas empresas acabam escolhendo a utilização de métodos ágeis de desenvolvimento, essas metodologias não são orientadas a documentação nem apenas a codificação. (SOMMERVILLE, 2007)

Na sua maioria, as metodologias ágeis não possuem nada de novo, o que as diferencia das metodologias tradicionais são o enfoque e os valores. O enfoque dessas metodologias é nas pessoas, e não em processos ou algoritmos. Além disso, essas metodologias se preocupam mais com a implementação do que com a documentação. Uma outra característica das metodologias ágeis é que elas são adaptativas e não preditivas. Assim elas se adaptam a novos fatores decorrentes do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do projeto.

Para a metodologia ser considerada ágil ela deve aceitar a mudança ao invés de tentar prever o futuro. O problema não são as mudanças, porque elas sempre vão acontecer no decorrer do desenvolvimento do projeto, mas sim como receber, avaliar e responder às mudanças.

Enquanto as metodologias ágeis variam em termos de práticas e ênfases, elas compartilham de algumas características, como desenvolvimento iterativo e incremental, comunicação e redução de produtos intermediários, como documentação extensiva. Desta forma a possibilidade de atender aos requisitos do cliente são maiores, mesmo que esses requisitos mudem.

2.5.1. Extreme Programing(XP)

O *extreme programing*(XP) é talvez o mais conhecido e mais usado dos métodos ágeis. No *extreme programing*, todos os requisitos são expressos como cenários, que são implementados diretamente como uma série de tarefas (SOMMERVILLE, 2007).

As principais características dessa metodologia são:

- Feedback Constante;
- Abordagem Incremental
- A comunicação entre as pessoas é encorajada

A metodologia XP enfatiza o desenvolvimento rápido do projeto de forma a garantir a satisfação do cliente. Em vez de entregar tudo de uma só vez em uma data distante, essa metodologia entrega o software que você precisa quando você precisa. A metodologia XP também encoraja os desenvolvedores a responderem as mudanças que os clientes pediram, mesmo que seja no final do ciclo de desenvolvimento.

As regras, práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de software para os desenvolvedores, que segundo Beck(1999) são conduzidos por quatro valores:

- Comunicação: a finalidade deste princípio é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projeto também é encorajada.

- Simplicidade: Visa permitir a criação de códigos simples que não devem possuir funções desnecessárias. Código simples quer dizer implementar o software com o menor número de classes e método possível. Outra ideia importante da simplicidade é implementar apenas requisitos atuais, evitando-se adicionar funcionalidades, mesmo que essas possam ser importantes no futuro. A ideia do XP é que é melhor fazer algo mais simples hoje e pagar um pouco mais amanhã do que implementar algo complicado hoje que talvez não venha a ser usado, sempre considerando que requisitos são mutáveis.

- Feedback: Feedback constante significa que o programador terá informações constantes do código e do cliente. A informação do código é dada pelos testes constantes, que indicam os erros tanto individuais, quanto do software integrado. Em relação ao cliente significa que ele terá frequentemente uma parte do software totalmente funcional para avaliar. Com isso o cliente constantemente sugere novas

características e informações aos desenvolvedores. Eventuais erros e não conformidades são identificadas rapidamente e corrigidos nas próximas versões. Desta forma, a tendência é que o produto final esteja de acordo com as expectativas do cliente.

- Coragem: A coragem se faz necessária para implantar os outros três valores. Por exemplo, é preciso coragem para identificar uma oportunidade de simplificar e experimentá-la, também é preciso de coragem para obter o feedback constante do cliente.

Além dessas características, a metodologia XP baseia-se nas 12 práticas descritas por Beck(1999):

- Planejamento: consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto. A XP baseia-se em requisitos atuais para desenvolvimento de software, não em requisitos futuros. Beck(1999) diz que devemos tornar possível iniciar com um plano simples e refiná-lo com o passar do tempo. Além disso, a metodologia XP procura evitar problemas de relacionamento entre a área de negócios (clientes) e a área de desenvolvimento. As duas áreas devem cooperar para o sucesso do projeto, e cada uma deve focar em partes específicas do projeto. Desta forma, enquanto a área de negócios deve decidir sobre o escopo, a composição das versões e as datas de entrega, os desenvolvedores devem se preocupar com o processo de desenvolvimento, o cronograma, estimativas de prazo e de custo para que o software corresponda as expectativas dos clientes.

- Entregas frequentes: visa a construção de um software simples, e conforme os requisitos surgem, há a atualização do software. Cada versão entregue deve ter o menor tamanho possível, contendo os requisitos atuais de maior valor para o negócio. O ideal é que novas versões sejam entregues a cada mês, ou no máximo a cada dois meses, aumentando a possibilidade de feedback rápido do cliente. Essas entregas em curtos períodos evitam surpresas caso o software seja entregue após muito tempo, melhora as avaliações e o feedback do cliente, aumentando assim a possibilidade do software final estar de acordo com as expectativas do cliente.

- Metáfora: descrever o software sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do software.

- Projeto Simples: o programa desenvolvido pelo método XP deve ser o mais simples possível, satisfazendo sempre os requisitos atuais dos clientes, sem se preocupar com os requisitos futuros. Os requisitos futuros só devem ser adicionados quando eles realmente existirem.

- Testes: a metodologia XP focaliza a validação do projeto durante todo o processo de desenvolvimento. Os programadores desenvolvem o software criando primeiramente os testes.

- Programação em pares: a implementação do código é feita em dupla, ou seja, dois desenvolvedores trabalham em um único computador. O desenvolvedor que está com o controle do teclado e do mouse implementa o código, enquanto o outro observa continuamente o trabalho que está sendo feito, procurando identificar erros sintáticos e semânticos e pensando em como melhorar o código que está sendo implementado. Os papéis devem ser alterados continuamente. A vantagem da programação em dupla é que os desenvolvedores estarão continuamente aprendendo um com o outro.

- Refatoração: focaliza o aperfeiçoamento do projeto de software e está presente em todo o desenvolvimento. A refatoração só deve ser feita quando for necessário, ou seja, quando um desenvolvedor da dupla, ou os dois, perceber que é possível simplificar o módulo atual sem perder nenhuma funcionalidade.

- Propriedade Coletiva: o código do projeto pertence a todos os membros da equipe. Isso significa que qualquer pessoa que percebe que pode adicionar valor a um código, pode fazê-lo, mesmo que não tenha sido ele que tenha desenvolvido o código. Mas para acrescentar esse valor a pessoa deve fazer todos os testes necessários para garantir que o código não perderá nenhuma funcionalidade. Isso é possível porque na XP todos são responsáveis pelo software inteiro, a grande vantagem dessa ideia é que caso algum programador saia do meio de um projeto, a equipe consegue continuar o projeto sem muitos problemas, já que todos tem o conhecimento de todas as partes do software que está sendo desenvolvido, mesmo que não seja de forma detalhada.

- Integração Contínua: integrar e construir o sistema de software várias vezes por dia, mantendo os programadores em sintonia, além de possibilitar processos rápidos. Integrar apenas um conjunto de modificações de cada vez, assim fica óbvio quem deve fazer as correções quando os testes falham: a última equipe que integrou o código novo ao software.

- 40 horas de Trabalho Semanal: a XP assume que não se deve fazer horas extras constantemente. Caso seja necessário trabalhar mais do que 40 horas pela segunda semana consecutiva, existe um problema sério no projeto que deve ser resolvido não com um aumento de horas trabalhadas, mas com melhor planejamento, por exemplo. Caso seja necessário, os planos devem ser alterados, ao invés de sobrecarregar as pessoas.

- Cliente presente: a participação do cliente durante todo o desenvolvimento do projeto é fundamental. O cliente deve estar sempre disponível para sanar todas as dúvidas de requisitos, evitando atrasos e até mesmo construções erradas. Beck(1999) propõe a ideia de manter o cliente como parte integrante da equipe de desenvolvimento.

Figura 4 - Metodologia XP (WELLS, 2002)

2.6. Metodologia Scrum

A ideia principal da metodologia Scrum é que o desenvolvimento de softwares envolve muitas variáveis técnicas e do ambiente, como requisitos, recursos e tecnologias, que podem mudar durante o processo. Isso torna o processo de desenvolvimento complexo, requerendo flexibilidade para acompanhar as mudanças. Devido a essa necessidade de flexibilidade a metodologia Scrum é baseada em algumas ideias da teoria de controle de processos industriais para o desenvolvimento de software, introduzindo assim as ideias de produtividade e adaptabilidade além da flexibilidade. O objetivo da metodologia Scrum é fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto.

A metodologia Scrum é baseada seguintes princípios: equipes pequenas, requisitos pouco estáveis ou desconhecidos e iterações curtas para promover a visibilidade para o desenvolvimento.

A Scrum divide o desenvolvimento em iterações (*sprints*) de trinta dias. Equipes pequenas, de até dez pessoas, são formadas por projetistas, programadores, engenheiros e gerentes de qualidade. Estas equipes trabalham em cima dos requisitos (funcionalidade) definidos no início de cada Sprint. A equipe é responsável pelo desenvolvimento desta funcionalidade.

Na metodologia Scrum existem reuniões de acompanhamento diárias. Nessas reuniões, que são preferencialmente de curta duração (aproximadamente quinze minutos), são discutidos pontos como o que foi feito desde a última reunião e o que precisa ser feito até a próxima. As dificuldades encontradas e fatores de impedimentos (*bottlenecks*) são identificados e resolvidos.

O ciclo de vida da metodologia Scrum é baseado em três fases principais:

- Pré-planejamento: os requisitos são descritos em um documento chamado backlog. Posteriormente eles são priorizados e são feitas estimativas de esforço para o desenvolvimento de cada requisito. O planejamento inclui também, a definição da equipe de desenvolvimento, as ferramentas a serem usados, os riscos do projeto e as necessidades de treinamento. Finalmente é proposta uma arquitetura de

desenvolvimento. Eventuais alterações nos requisitos descritos no backlog são identificadas, assim como seus possíveis riscos.

- Desenvolvimento: as variáveis técnicas e do ambiente identificadas previamente são observadas e controladas durante essa fase. Diferente das outras metodologias que só consideram essas variáveis no início do projeto, a Scrum controla essas variáveis continuamente o que aumenta a flexibilidade para acompanhar as mudanças. Nesta fase o software é desenvolvido em ciclos (sprints) em que novas funcionalidade são adicionadas. Cada Sprint é desenvolvido de uma forma tradicional, primeiramente faz-se uma análise, em seguida o projeto, implementação e testes. Cada Sprint é planejado para durar de uma semana a um mês.

- Pós-planejamento: após a fase de desenvolvimento são feitas reuniões para analisar o progresso do projeto e demonstrar o software atual para os clientes. Nesta fase são feitas as etapas de integração, testes finais e documentação.

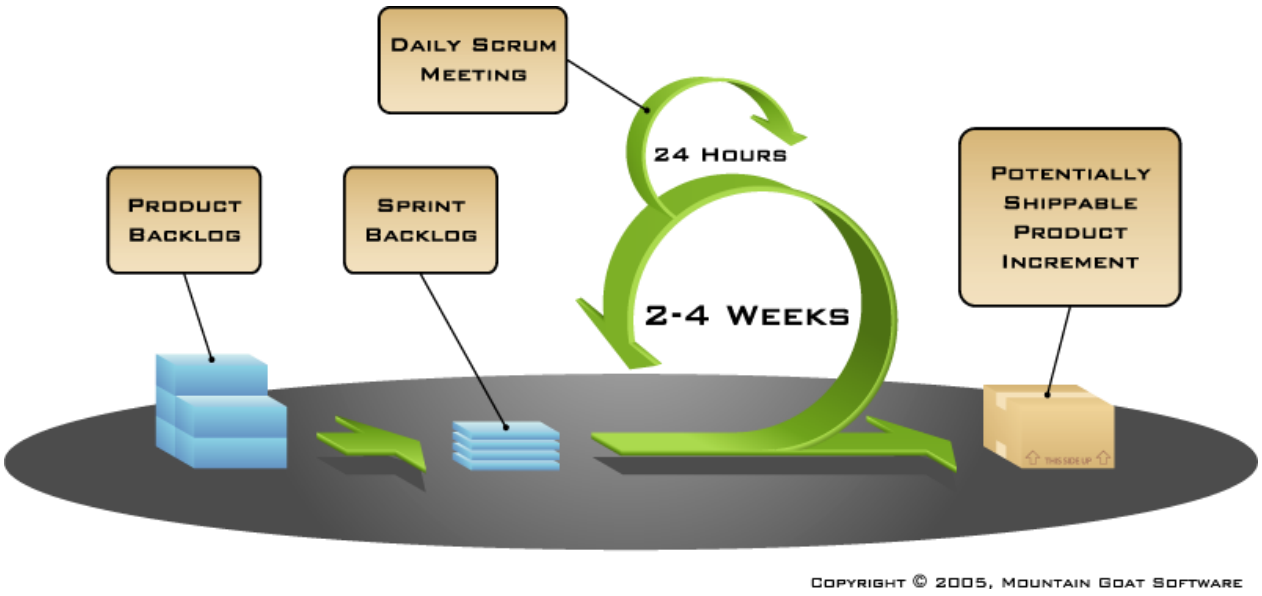

Figura 5 - Etapas Scrum (MOUNTAIN GOAT SOFTWARE, 2005)

2.7. Comparação Entre os Métodos de Desenvolvimento de Software

No quadro 2 é apresentada uma comparação entre os métodos de desenvolvimento de software. Essa comparação foi feita com base em atributos comuns aos modelos, que são abordados de formas diferentes. Destaca-se na comparação o item documentação, que é utilizada de forma diferente em cada método.

Os critérios de comparação foram escolhidos de modo a destacar o método que melhor se adapta a documentação e ao processo de software de micro e pequenas empresas.

	Planejamento	Desenvolvimento Ágil	Flexibilidade	Documentação	Feedback do Cliente	Gerenciamento de Riscos
Modelo Cascata	A etapa de planejamento é fundamental para o sucesso desse modelo, o fato de que uma fase só começa quando a outra termina obriga a um planejamento bem detalhado	Não é um método ágil, requer muito planejamento e uma equipe um tanto quanto grande trabalhando no projeto	Modelo pouco flexível, não reage muito bem a mudanças, uma mudança no escopo do projeto geraria um grande atraso nos prazos e aumento nos custos	Documentação muito detalhada do desenvolvimento do software, ao final de cada etapa uma documentação é gerada	Nível de feedback muito baixo, o cliente só terá algum contato com o software nas etapas finais	Os riscos devem ser detalhados na etapa de planejamento, devem ser bem gerenciados para que não ocorram atrasos no projeto
Prototipação	A etapa de planejamento consiste apenas no levantamento de requisitos	Não é um método ágil de desenvolvimento, o enfoque do método se encontra no feedback rápido do cliente ao visualizar o protótipo	Reage bem a mudanças no início do processo, visto que o cliente entra em contato com o protótipo muito rapidamente	A documentação quase não é utilizada nesse método	Feedback constante, o protótipo é gerado o quanto antes para que o cliente veja se é aquilo mesmo que ele precisa	Os riscos não são trabalhados, visto que a etapa de planejamento é focada apenas no levantamento de requisitos
Modelo Espiral	A etapa de planejamento é muito usada nesse método, além do planejamento no início do projeto, planeja-se o início de cada etapa do processo	Não é um método ágil de desenvolvimento, devido a essa necessidade de planejamento ao início de cada etapa	Devido ao alto nível de gerenciamento de riscos, esse processo reage bem a mudanças, sempre possuindo caminhos alternativos para cada risco	Documentação utilizada ao final de cada etapa do processo, ao final de cada etapa a documentação é revisada e então planeja-se o prosseguimento das etapas do processo	O feedback não é muito constante, ocorre quando um protótipo é gerado, antes disso o cliente não tem contato com o software que está sendo desenvolvido	O gerenciamento de riscos é o ponto forte desse processo, os riscos são avaliados no início do projeto e ao final de cada etapa do processo

Quadro 2 - Comparação dos Métodos de Desenvolvimento de Software

Continuação do Quadro 2.

	Planejamento	Desenvolvimento Ágil	Flexibilidade	Documentação	Feedback do Cliente	Gerenciamento de Riscos
Extreme Programing (XP)	A etapa de planejamento preocupa-se apenas com os requisitos atuais do cliente, deixando de lado os possíveis requisitos futuros, enquanto a área de negócios se preocupa com o escopo do projeto, os desenvolvedores se preocupam com cronograma, estimativas de custo e prazos	A agilidade do método é o seu ponto forte, visa entregar o software que o cliente precisa quando ele precisa	Reage bem as mudanças, nem sempre as mudanças são feitas quando pedidas, mas é garantido que elas serão feitas até o fim do processo	A documentação pode ser utilizada a cada entrega, mas geralmente não é devido a necessidade de agilidade no desenvolvimento	Feedback constante, entrega-se sempre o menor software possível atendendo as necessidades do cliente para que se tenha o feedback o quanto antes, isso garante que ao final do processo o software estará de acordo com as necessidades do cliente	Gerenciamento de riscos é pouco trabalhado nesse método, o enfoque do método é na implementação e não no planejamento.
Scrum	A etapa de planejamento é curta, define-se os requisitos, tecnologias necessárias, esforço necessário, definição das equipes de desenvolvimento, ferramentas a serem usadas, riscos e eventuais necessidade de treinamento da equipe	Método de desenvolvimento ágil, trabalha com equipes pequenas e com entregas constantes	Reage muito bem a mudanças, é um modelo bastante utilizado para requisitos não muito bem definidos ou até mesmo desconhecidos, essa flexibilidade é o ponto forte do modelo scrum	Documentação constante, ao final de cada sprint é gerada uma documentação	Feedback constante, o cliente entra em contato com o software bem cedo, gerando feedback ao final de cada Sprint	Os riscos são levantados na etapa de planejamento, porém não é uma etapa chave do processo devido a sua flexibilidade

2.8. Micro e Pequena Empresa

As micro e pequenas empresas podem ser classificadas por diversos critérios, sabendo-se que os mais utilizados são o faturamento anual bruto e o número de funcionários.

Adotando-se o faturamento anual bruto, deve-se observar o Regime Tributário Especial, criado pela Lei complementar número 123 de 14 de dezembro de 2006, que estabelece que a micro empresa tenha um rendimento anual bruto entre R$240.000,00 e R$2,4 milhões de Reais (SEBRAE, 2005).

Com relação ao número de funcionários a classificação mais utilizada é adotada pelo Serviço Brasileiro de Apoio às Micro e Pequenas Empresas – SEBRAE, conforme segue: Micro empresa: Organização atuante na área de comércio e serviços com até 9 funcionários; Pequena empresa: Organização atuante na área de comércio e serviços, com 10 a 49 funcionários.

Para Anholom et al (2007) as micro e pequenas empresas têm características que podem conduzi-las tanto ao sucesso como ao fracasso. Ainda, diz que elas estão mais sujeitas a mudanças de mercado devido principalmente a algumas de suas características mais específicas, conforme é apresentado a seguir.

- Falta de Planejamento Estratégico: Não importa o porte da empresa, ela sempre irá necessitar de uma visão de negócio, previsão de vendas, distribuição de recursos, dente outros;
- Pedidos Acima da Capacidade Produtiva: É comum que as micro e pequenas empresas aceitem pedidos maiores do que a sua capacidade de produção e este fator, aliado a falta de previsão, acarreta na incapacidade de cumprir o cronograma de entregas;
- Falta de Políticas de Melhoria Contínua: Ainda que, às vezes, a empresa evolua para um patamar superior, seja de qualidade ou mesmo de produção, os empresários dificilmente estabelecem um plano de melhoria contínua;
- Poucos Recursos Financeiros: Normalmente a micro ou a pequena empresa trabalha com trabalhos de menor porte influenciando diretamente na sua renda.

Anholom et al (2007) também cita características dessas empresas, que se bem exploradas, podem conduzi-las ao sucesso. Dentre estas características pode-se citar:

- Grande Flexibilidade: A maioria dos projetos que as micro e pequenas empresas trabalham são de pequeno porte, favorecendo a capacidade de mudanças no projeto;

- Comunicação Mais Efetiva: Com um número reduzido de funcionários e um espaço mais próximo para a realização das tarefas tem-se um ambiente de fácil comunicação;
- Contato Mais Próximo com o Cliente: A proximidade com o cliente facilita a comunicação com o mesmo, proporcionando assim um entendimento melhor dos requisitos do cliente, o que gera um aumento na qualidade de produtos e serviços;
- Estrutura Hierárquica bem Definida: Normalmente é administrada pelo dono e que, em muitos casos, ainda exerce outras funções, tendo maior contato com todos os processos.

Com base nas características das micro e pequenas empresas pode-se estimar sua grande importância na geração de oportunidades de negócio, mas para que isso seja aproveitado, é necessário investir na parte da gestão dessas empresas.

3. METODOLOGIA

Segundo Silva e Menezes (2005) pode-se classificar a pesquisa quanto a sua natureza como uma pesquisa aplicada, já que objetiva gerar conhecimentos para a aplicação prática e dirigida à solução de problemas, envolvendo interesses reais. Quanto à abordagem, a pesquisa é qualitativa, ou seja, uma relação dinâmica entre o mundo real e o subjetivo que não pode ser traduzido em números. O estudo é realizado através de análises e interpretações, não se aplicando ferramentas estatísticas. Os processos e os seus significados são os focos principais.

Do ponto de vista dos objetivos, a pesquisa é exploratória, visa proporcionar maior familiaridade com o problema com vistas a torna-lo explícito ou a construir hipóteses.

Quanto aos procedimentos técnicos, o trabalho é um estudo de caso, pois envolve um estudo profundo que permite detalhado conhecimento.

Os passos identificados para a realização do trabalho foram:

- Revisão bibliográfica dos conceitos relacionados (padronização, desenvolvimento de software, documentação e micro e pequenas empresas);
- Caracterizar o ambiente de estudo através de um estudo de caso;
- Levantar as informações do processo de software e do processo de documentação por meio de um estudo de caso.
- Definir o melhor tipo de documentação a ser utilizada;
- Estudar a documentação escolhida;
- Elaborar um processo padrão para documentar o processo de software;

Os instrumentos a serem utilizados:

- Questionário;
- Entrevistas;
- Observação;
- Levantamento de documentos.

O questionário foi aplicado junto a empresa durante o processo de levantamento dos documentos de software. O processo de observação ocorreu junto com as entrevistas com os

responsáveis pelos documentos gerados. O levantamento de documentos foi o resultado obtido através do questionário, das entrevistas e da observação.

4. DESENVOLVIMENTO

4.1. Contexto da Empresa

A empresa estudada preferiu não ser identificada, para facilitar a identificação da empresa daremos o nome de SM (Software Maringá). A empresa SM consiste em uma empresa prestadora de serviços de software, e desenvolve softwares em diversas plataformas, como Java, Delphi, Progress e .Net.

A SM foi fundada no ano 2000 e tem sua sede situada na cidade de Maringá, estado do Paraná, possuindo também filial em Presidente Prudente, estado de São Paulo, e em Hyderabad – Índia. O estudo do caso foi realizado na SM de sede, que conta com aproximadamente 110 colaboradores.

Podendo ser caracterizada como uma EPP (Empresa de Pequeno Porte), a SM tem como maior mercado consumidor o estado do Paraná, este mercado está se expandindo para outras regiões, porém elas ainda não são tão significativas quanto a região sul.

A SM possui certificação CMMI nível 2, isso indica que suas tarefas são bem organizadas, padronizadas de acordo com o órgão CMMI, possuindo também diversos indicadores de desempenho que mostram a qualidade do seu serviço. Hoje devido ao grau de padronização de seus processos a SM busca a certificação CMMI nível 3.

4.2. Documentação de Software

O quadro 3 caracteriza todas as documentações utilizadas no processo de desenvolvimento de software da SM. Para o levantamento dessa documentação utilizou-se de um questionário, que foi respondido pelo responsável pela geração de cada documento. As perguntas foram elaboradas para levantar as principais informações dos documentos, como por exemplo, qual a finalidade do documento, que tipo de informação o documento deve conter, qual a importância deste documento. O questionário consistia das seguintes perguntas:

- Qual o nome do documento?
- Quem é o responsável por fazer este documento?
- Em qual etapa do desenvolvimento de software o documento é desenvolvido?
- Como o documento é desenvolvido?

- Qual a importância deste documento?
- Qual as informações que o documento deve conter?
- Qual a finalidade do documento desenvolvido?

4.2.1. Documentação de Requisito

A documentação de requisito consiste em documentar os requisitos do cliente, esta documentação será feita pelo analista de negócios. Este documento é elaborado durante a fase de levantamento de requisitos, geralmente 15 dias antes do software entrar em desenvolvimento.

A documentação de requisitos é feita através do manual de requisitos, que consiste em um conjunto de práticas e perguntas pré-definidas para auxiliar o analista de negócios a levantar os requisitos junto ao cliente. A etapa de levantamento de requisitos é fundamental para um software de qualidade, porém é bem difícil levantar requisitos do cliente, por isso a necessidade da utilização do manual de requisitos.

O conteúdo da documentação de requisitos consiste nas necessidades e expectativas do cliente com relação ao software a ser desenvolvido. Este documento servirá de guia para todo o processo de desenvolvimento do software e auxiliará na definição do escopo do projeto.

4.2.2. Documentação do Plano de Projeto

A documentação do plano de projeto consiste em documentar tudo o que foi definido na etapa de planejamento do projeto, informações como WBS (*Work Breakdown Structure*) do projeto, cronograma, requisitos, riscos, premissas estarão dentro deste documento.

O responsável pela documentação do plano de projeto é o gerente do projeto, este documento é muito importante para evitar mudanças de escopo do projeto, deixar documentado o prazo limite para a entrega do software, o custo para o desenvolvimento do software e como será o processo de desenvolvimento deste software.

4.2.3. Documentação da Solução Técnica

O responsável pela documentação da Solução Técnica é o Scrum Master. O Scrum Master ficará responsável por reunir a equipe de desenvolvimento do software e discutir a melhor

solução para o desenvolvimento do software, ou seja, a melhor técnica de desenvolvimento a ser utilizada para aquele projeto, levando em conta custos, tecnologias utilizadas e etc. Essa reunião do Scrum Master com sua equipe também será importante para que todos saibam exatamente o que será desenvolvido. A documentação da solução técnica deve conter o método escolhido pela equipe para o desenvolvimento do projeto.

4.2.4. Documentação do Check-list de Verificação do Código Fonte

A verificação do código fonte consiste na leitura dos códigos das funções desenvolvidas, uma auditoria é marcada e uma equipe é definida para essa verificação do código fonte. Cada membro da equipe terá uma função, uma pessoa ficará responsável por ler os códigos, outra por organizar a equipe, outra por responder a um questionário pré-definido que irá garantir a qualidade e conformidade das funções. A documentação do Check-list de Verificação do Código Fonte consiste em documentar as respostas das perguntas, resultados apresentados pelos códigos desenvolvidos, se estão ou não conforme com o que era pra ser desenvolvido. O responsável por essa documentação é o programador designado para aquela auditoria que é realizada durante a etapa de desenvolvimento do software.

4.2.5. Documentação do Sumário de Teste

Diversos tipos de teste serão aplicados para garantir a qualidade do software a ser desenvolvido, a documentação do sumário de teste consiste em documentar os resultados e um resumo do histórico desses testes. O responsável por essa documentação é o analista de testes, e essa documentação será realizada a cada versão do produto a ser liberada. A documentação do sumário de teste irá mostrar ao Gerente de Configuração a situação da versão a ser liberada.

4.2.6. Documentação de Release Notes

Após a implementação do software, o cliente poderá solicitar algumas mudanças e a documentação de release notes será feita para armazenar as informações dessas mudanças. Cabe ao Gerente de Configuração elaborar a documentação de Release Notes, a documentação deve conter dados como: Lista de funcionalidades inseridas ou retiradas do produto, mudanças de interface feitas, o que realmente foi alterado e o que foi solicitado pelo cliente para se alterar.

Nome do Documento	Responsável	Etapa	Como	Importância	Conteúdo	Objetivo
Requisito	**Analista de Negócio**	Levantamento de Requisitos, 15 dias antes de entrar para o desenvolvimento.	**Elicitando os requisitos com o cliente, o levantamento de requisitos é feito através do Manual de Requisitos. O Manual de Requisitos consiste em perguntas e práticas pré-definidas para o levantamento de requisitos.**	**Será o guia para todo o processo de desenvolvimento de software**	**Necessidades e expectativas do cliente; regras de negócio e visão da funcionalidade no ambiente do cliente.**	**Documentar as expectativas do cliente. Auxiliar na definição do escopo do projeto**
Plano de Projeto	**Gerente de Projeto**	**Planejamento do Projeto**	**Com as necessidade do cliente, restrições e premissas do projeto, riscos, escopo e cronograma, recursos e ambiente do trabalho.**	**Plano definido para o desenvolvimento do projeto**	**Processo definido do projeto, datas de entregas, indicadores que serão monitorados, premissas e restrições e os riscos do projeto**	**Documentar as expectativas do cliente. Auxiliar na definição do escopo do projeto**

Quadro 3 - Documentos de Software

Continuação do Quadro 3.

Nome do Documento	Responsável	Etapa	Como	Importância	Conteúdo	Objetivo
Solução Técnica	Scrum Master	Reunião de SP2 do ciclo Scrum	Utilizando os requisitos e conhecendo as regras de negócio do sistema	Busca encontrar a melhor forma de desenvolver o software requisitado, levando em consideração custo e benefício; estimular o reuso de componentes e descoberta de novas tecnologias. Também visa gerar equilíbrio no conhecimento do time, devido as discussões realizadas em busca da melhor solução.	Definição técnica de como será desenvolvido o requisito	Estipular a melhor maneira para o desenvolvimento do projeto
Checklist de Verificação do Fonte	Programador	Após desenvolvimento	Através de Auditorias e Verificações do código fonte. O código fonte desenvolvido por um programador será revisto por outros programadores nas auditorias	Para garantir a qualidade e padrões estipulados pela empresa.	Itens críticos e padrões que devem conter no código fonte e na funcionalidade com o resultado se passou ou não na verificação	Para listar todas as perguntas que devem ser respondidas pelo programador.
Sumário de Teste	Após os teste na versão do produto	Utilizando os resultados dos testes que passaram e não passaram; quantidade de casos de teste aplicados.	Utilizando os resultados dos testes que passaram e não passaram; quantidade de casos de teste aplicados.	Divulgação da situação da versão para o Gerente de Configuração; Histórico do resumo dos testes aplicados; formalização do aval do teste para liberar para o cliente	Número de casos de teste, quais funcionalidades não passaram, responsável pelo teste, criticidade dos bugs.	Documentar os resultados dos testes para liberação da versão

Continuação do Quadro 3.

Nome do Documento	Responsável	Etapa	Como	Importância	Conteúdo	Objetivo
Release Notes	Gerente de Configuração	Após a implementação do produto	Gerado pela ferramenta Hudson	Evidencia todas as funcionalidades que entraram na versão do produto; Possibilidade de comparar o que estava planejado para entrar e o que realmente entrou. Gerenciar as mudanças feitas nas versões	Lista de funcionalidades que foram desenvolvidas, entregues e colocadas no produto.	Divulgação de novas funcionalidades para a área de Teste e para os clientes

4.3. Problemas Encontrados no Processo de Documentação do Desenvolvimento de Softwares e Soluções Propostas

O processo de documentação de desenvolvimento de software, é um processo árduo e que exige esforços de todos os membros da empresa, desde programadores até a alta gerência. Diversas dificuldades serão enfrentadas e vários problemas irão surgir a medida que o processo de documentação é implantado e executado. O quadro 4 mostra os principais problemas encontrados durante o processo de documentação e propõe soluções para esses problemas. Os problemas foram levantados através de entrevistas com os colaboradores da empresa estudada.

Problemas Encontrados	Soluções Propostas
Falta de Pessoal	Buscar contratar estagiários que fiquem responsáveis apenas pela parte do preenchimento dos documentos, evitando assim que os programadores tenham que parar suas atividades
Resistência do pessoal a fazer a documentação de forma correta	Conscientização do pessoal através de palestras, mostrando a importância e os resultados que serão obtidos com a documentação
Falta de conhecimento necessário para fazer a documentação de forma correta	Treinamento constante do pessoal, para que eles se mantenham informados de como preencher os documentos de forma correta.
Rotatividade constante da equipe	Manter a política de treinamento contínuos, assim os novos membros da equipe saberão como fazer o processo de documentação e os membros mais antigos se manterão atualizados sobre a forma correta de preencher os documentos.

Quadro 4 - Problemas Encontrados no Processo de Documentação do Desenvolvimento de Softwares e Soluções Propostas

4.4. Proposta de Padronização da Documentação de Software Utilizando a Metodologia SCRUM

A proposta de padronização da documentação software irá utilizar o modelo SCRUM. O modelo SCRUM é muito utilizado em micro e pequenas empresas por ser um método de desenvolvimento ágil, bastante versátil, que exige um contingente baixo de mão de obra, possui fases de desenvolvimento bem definidas, além de possuir entregas constantes, que gera

um feedback maior do cliente, diminuindo assim, o risco de desenvolvimento de um software que não esteja dentro das expectativas do cliente.

4.5. Plano de Atividades e Documentações para o Desenvolvimento do Software

A primeira etapa do método SCRUM consiste no pré-planejamento, nesta etapa será desenvolvido o plano do projeto, que deve conter as seguintes informações:

- Requisitos do cliente
- Regras de negócio
- Equipe de desenvolvimento
- Ferramentas a serem utilizadas
- Cronograma do projeto
- Escopo do projeto
- Riscos do projeto

Para se iniciar a elaboração do plano de projeto, deve-se levantar os requisitos junto aos clientes. Com os requisitos já levantados, deve-se definir a equipe que irá trabalhar no projeto, para que se tenham definido os recursos disponíveis para a realização do projeto. Após a definição da equipe será construído um cronograma detalhado do projeto, com todas as atividades a serem desenvolvidas, datas inicias e finais para cada uma dessas atividades e a definição dos recursos necessários para a realização das mesmas.

A próxima tarefa a ser realizada será o desenvolvimento do escopo do projeto, o escopo do projeto deverá conter: os requisitos que foram levantados, regras de negócios, estimativas de custo, riscos do projeto, WBS do projeto e qualquer outra informação que se ache necessária para que o cliente consiga visualizar de maneira clara o projeto que será desenvolvido.

Nesta etapa devem ser gerados dois documentos distintos, sendo eles o documento de requisitos e o documento do plano de projeto. O documento de requisitos deverá ser elaborado pelo analista de negócios. Muitas vezes em micro e pequenas empresas não

existe uma definição clara dos papéis de cada integrante da equipe. Neste caso o responsável pelo documento de requisitos será o membro que definiu os requisitos junto ao cliente. Este documento terá de conter as necessidades e expectativas do cliente, regras de negócio e visão da funcionalidade no ambiente do cliente. O documento de requisitos será muito importante para auxiliar na elaboração do escopo do projeto, e também deixará documentado as expectativas do cliente.

O documento do plano de projeto deverá ser feito pelo gerente do projeto, esse documento deve conter: o cronograma detalhado de desenvolvimento do projeto (WBS do Projeto), distribuição de tarefas, alocação de recursos para cada tarefa, datas de entrega, restrições e premissas do projeto e o escopo do projeto. O documento do plano de projeto será de grande importância para evitar mudanças no escopo do projeto, essas mudanças na maior parte das vezes geram: mudanças de custo do projeto e atrasos nos prazos de entrega. Tendo documentado o escopo inicial do projeto, o cliente terá consciência que se exigir alguma mudança grande do escopo, os prazos de entrega e custos também poderão ser alterados.

Assim que esses dois documentos forem finalizados, a segunda fase da etapa de desenvolvimento do método SCRUM pode ser iniciada.

A segunda etapa do método SCRUM consiste no desenvolvimento do software em si, ou seja, gerar os códigos do software. Esta fase é constituída de sprints que duram de uma semana a um mês, durante esses sprints serão desenvolvidas diversas funções do software para que ao final de cada sprint possa ser realizada uma entrega de parte do software ao cliente. Diversos tipos de testes serão realizados para garantir a qualidade destes sprints, portanto será necessário gerar um documento de sumário de testes. O documento de sumário de testes deverá conter o histórico completo dos testes realizados naquele Sprint, dados como tipo de testes e resultados dos testes deverão constar neste documento. Este documento irá facilitar a visualização da situação do pacote a ser entregue ao cliente, se ele está de acordo ou não com as expectativas e se poderá ou não ser entregue ao cliente. O documento de sumário de testes deverá ser realizado a cada Sprint, garantindo assim que todos os testes de todos os sprints estejam documentados. O responsável pelo documento de sumário de testes é o analista de testes.

Assim que o primeiro pacote é entregue ao cliente, temos o início da terceira fase do método scrum, esta etapa consiste na integração do software. O *feedback* do cliente em relação ao que está sendo entregue começa nessa fase, nessa etapa serão requisitadas mudanças no que foi entregue ao cliente, mudanças como: acréscimo de novas funções, retirada de funções, mudanças de interfaces e etc. A cada mudança solicitada será gerado um novo release no programa, nesse release existirão novas funções, ou funções deixarão de existir. Para o gerenciamento desses releases será necessário um documento de release notes, nele deverá constar todas as alterações feitas nos releases, evidenciando assim as novas funções implementadas, para que elas possam ser devidamente testadas antes de serem entregues ao cliente.

A Figura 6 é um esquema que representa as atividades que foram descritas, bem como apresenta os documentos que serão gerados em cada uma das três etapas do processo.

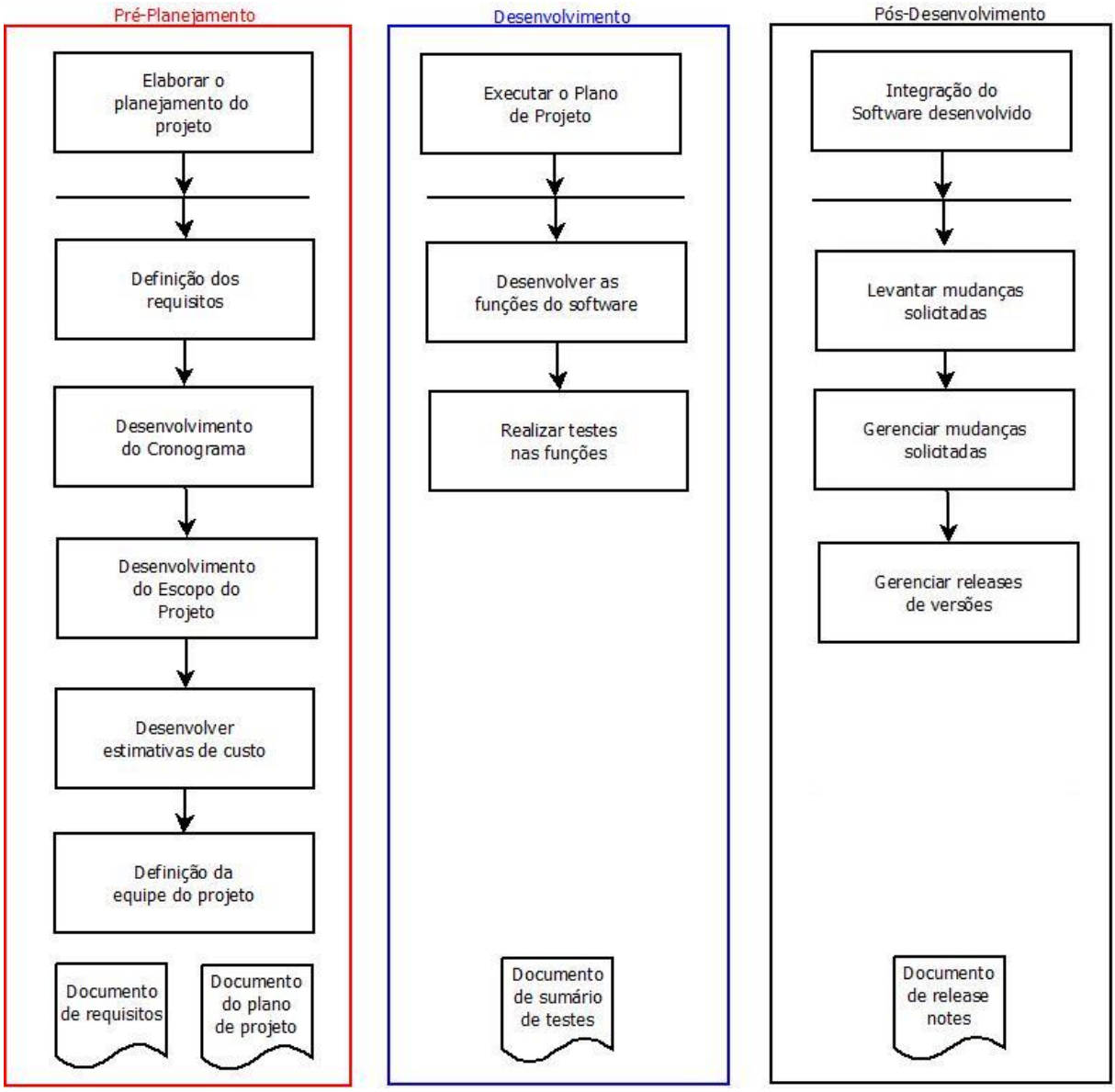

Figura 6 - Proposta de Padronização da Documentação de Software Utilizando a Metodologia SCRUM

4.6. Contribuições Esperadas

- Melhor qualidade do software desenvolvido;
- Facilitar o gerenciamento do processo de desenvolvimento do software;
- Melhor gerenciamento dos releases;
- Padronização das atividades a serem desenvolvidas;
- Aplicabilidade de indicadores de desempenho;

5. Considerações Finais

A padronização no processo de documentação do desenvolvimento de software foi o objetivo traçado a se desenvolver neste projeto. Pesquisando diversos métodos de desenvolvimento e estudando a SM foi possível selecionar um método de desenvolvimento aplicável a micro e pequenas empresas e mostrar que a documentação também pode ser feita nestas empresas.

Com a realização deste trabalho identificou-se que os métodos de desenvolvimento de software não são muito concretos, eles sempre se adaptarão ao projeto a ser desenvolvido. Também verificou-se que muito pouco se fala na literatura sobre documentação, ela apenas começou a ser citada, porém ainda não é trabalhada com um nível de detalhe muito grande.

Notou-se que com um planejamento bem realizado é possível que micro e pequenas empresas documentem seu processo de desenvolvimento de software, o que as tornará mais competitivas no mercado. Outros fatores identificados foram a grande importância das micro e pequenas empresas na economia nacional e o grande índice de mortalidade dessas empresas.

Após o estudo de caso, ficou claro que a documentação e padronização das atividades de desenvolvimento de software está se tornando um fator chave para o sucesso das empresas de software no mercado.

5.1. Limitações

A principal limitação encontrada foi a falta de informação na literatura sobre as documentações de software existente.

5.2. Trabalhos futuros

Realizar estudos de casos em diferentes empresas, para levantar mais informações sobre as documentações existentes e assim poder detalhar os documentos de softwares, fazendo com que esse assunto comece a surgir na literatura.

6. REFERÊNCIAS

ALVES, Lucas de Oliveira. **Padronização da Linha de Produção de Hidrolavadoras em uma Empresa Metal Mecânica**, 2011. Disponível em: < http://www.dep.uem.br/tcc/arquivos/TG-EP-57-11.pdf> Acesso em 05 abr, 2013.

ANHOLOM, Rosley. MORRETI, Diego de Carvalho. PINTO, Jefferson de Souza. ZOQUI, Eugênio José. **Características administrativas de micro e pequenas empresas: confronto entre as peculiaridades apresentadas pela literatura e observadas na prática ao longo da implantação de programas de melhorias,** 2005. Disponível em: < https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCgQFjAA&url=http%3A%2F%2Fwww.simpep.feb.unesp.br%2Fanais%2Fanais_12%2Fcopiar.php%3Farquivo%3DAnholon_R_Caracteristicas%2520administrativas%2520de%2520MPEs.&ei=AipnUr_EK6mY2wXF2oGwDA&usg=AFQjCNHooV8YTL2X0JZMZFGtfedachm9wg&sig2=7XwMTrpQzR5B8xPH0uTg7w&bvm=bv.55123115,d.b2I>. Acesso em 16 ago, 2013.

Beck, K., **Programação Extrema Explicada**, Bookman, (1999).

CMMI, **Guia CMMI para Desenvolvimento Versão 1.2**. Guia disponível em: <http://www.sei.cmu.edu/library/assets/whitepapers/CMMI-DEV_1-2_Portuguese.pdf>. Acesso em 10 ago, 2013.

COELHO, Hilda Simone. **Documentação de software: Uma necessidade.** Artigo disponível em: < http://www.periodicos.letras.ufmg.br/index.php/textolivre/article/view/24/24 > Acesso em 03 abr, 2013.

MEEGEN, Rene Alberto Van. **Análise crítica da utilização da padronização no sistema de melhoria dos centros de distribuição domiciliária dos correios.** 2002**.** Dissertação (Mestrado). Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/10468/000366410.pdf?sequence=1>. Acesso em 05 abr, 2013.

MOUNTAIN, Goat Software, 2005. Disponível em <http://www.mountaingoatsoftware.com/> Acesso em 24 nov, 2013.

MPSBR, **Guia Geral MPS de Software**. Guia disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012.pdf>. Acesso em 10 ago, 2013.

NUNES, Vanessa B. SOARES, Andrea O. FALBO, Ricardo A. **Apoio à Documentação em um Ambiente de Desenvolvimento de Software**. Artigo disponível em: <http://www.inf.ufes.br/~falbo/download/pub/2004-IDEAS-1.pdf>. Acesso em 05 abr, 2013.

PEREIRA, Rodolfo Miranda. TAIT, Tania Fatima Calvi. BRUZAROSCO, Donizete Carlos. **O uso e desenvolvimento de softwares em micro e pequenas empresas,** 2010. Disponível em: < http://periodicos.uem.br/ojs/index.php/RevTecnol/article/viewFile/8725/6844>. Acesso em 16 mar, 2013.

PRESSMAN, R. S. **Engenharia de Software**, 6 ed. São Paulo: McGraw-Hill, 2006.

SEBRAE.**Serviço Brasileiro de Apoio à Micro e Pequena Empresa. Legislação Básica da Micro e Pequena Empresa.** 2005. Disponível em < http://www.sebrae.com.br>. Acesso em: 24 de nov, 2013.

SILVA, Edna Lúcia da. MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação,** 2005. Disponível em <ftp://ftp.unilins.edu.br/brenoortega/metodologia/metodologia_de_pesquisa.pdf> Acesso em: 24 de nov, 2013

SOMMERVILLE, Ian. **Engenharia de Software**, 8 ed. São Paulo: Addison-Wesley,2007.
WELLS, Don. ***Extreme Programing.*** Disponível em: <http://www.extremeprogramming.org/>. Acesso em 24 nov, 2013.

YIN, Robert K. Estudo de Caso, planejamento e métodos. 2.ed. São Paulo: Bookman, 2001.

Universidade Estadual de Maringá
Departamento de Engenharia de Produção
Av. Colombo 5790, Maringá-PR CEP 87020-900
Tel: (044) 3011-4196/3011-5833 Fax: (044) 3011-4196