UNIVERSIDADE DO SUL DE SANTA CATARINA

DIOGO ANDRÉ LOFF

APLICAÇÃO DE MÉTODOS ÁGEIS DE ENGENHARIA DE SOFTWARE EM UM SISTEMA CAÓTICO

Araranguá

2010

UNIVERSIDADE DO SUL DE SANTA CATARINA

DIOGO ANDRÉ LOFF

APLICAÇÃO DE MÉTODOS ÁGEIS DE ENGENHARIA DE SOFTWARE EM UM SISTEMA CAÓTICO

Monografia apresentada ao Curso de Especialização em Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina, como requisito parcial para obtenção do titulo de Especialista.

Orientador: Prof. M. Sc. Nelson Abu Samra Rahal Junior

Araranguá

2010

AGRADECIMENTOS

Agradecemos a todos que nos ajudaram no levantamento de material e deram apoio ao desenvolvimento deste trabalho.

Agradecemos em especial ao orientador Nelson Abu que deu apoio no desenvolvimento do projeto nos ajudando com seu conhecimento e a empresa Logosystem Sistemas que permitiu a implantação deste projeto no seu dia-dia.

RESUMO

O objetivo deste trabalho acadêmico é demonstrar que as boas praticas no desenvolvimento e gestão de software podem dar resultados significativos independente do tamanho da empresa ou número de colaboradores, tendo o controle do atendimento e desenvolvimento de novos projetos como problema a ser resolvido. Para o desenvolvimento do projeto foram realizadas pesquisas exploratórias (a busca de estudo em livros e artigos) e de campo (visitas a empresa Logosystem e entrevistas com membros do núcleo de base tecnológica de Criciúma filiado a ACIC (Associação do Comércio e Indústria de Criciúma). Estas pesquisas vieram contribuir para validação do trabalho desenvolvido possibilitando uma analise e viabilidade de implantação do projeto no dia-a-dia de uma indústria de software. Por ser um projeto voltado a necessidade das indústrias de software, proporcionamos as empresas um melhor planejamento e controle de seus projetos, com a minimização de atrasos no desenvolvimento de produtos e maior qualidade no ambiente da empresa a atendimento aos clientes.

Palavras Chaves: Software. Caos. Boas Praticas. Desenvolvimento. Atendimento.

ABSTRACT

The purpose of this academic work is to demonstrate that good practices in the development and management software can give significant results regardless of company size or number of employees, taking control of the care and development of new projects as a problem to be solved. To develop the project have been performed exploratory research (search for study in books and articles) and field (Logosystem company visits and interviews with members of core technology base Criciúma affiliated with ACIC (Association of Commerce and Industry of Criciúma) . These studies have contributed to the validation work of enabling a feasibility analysis and project implementation in day-to-day life of a software industry. Because it is a project focused on the need for software industries, we provide companies with better planning and control of their projects, to minimize delays in product development and higher quality environment in the company's customer service.

Key Words: Software. Chaos. Best Practice. Development. Attendance.

LISTA DE ILUSTRAÇÕES

SUMARIO

1 DELINEAMENTO DA PESQUISA

Segundo PRESSMAN (2006), no atual cenário da indústria de software os principais problemas relacionados ao setor de desenvolvimento e atendimento estão ligados á falta de documentação do software e comunicação. Hoje existem vários métodos de trabalho voltados ao desenvolvimento de software, métodos tradicionais e métodos ágeis. Após pesquisa realizada com algumas empresas de software da região de Criciúma/SC foi possivel constatar que os métodos tradicionais costumam ser utilizados em poucas empresas, pois segundo entrevistados, este tipo de abordagem torna-se complexo e demanda grande organização da empresa. Já os métodos ágeis, grande parte dos entrevistados estão implantando ou pensam em implantar, pois vêem como uma ferramenta para auxiliar a organização.

Nesse contexto, um dos maiores obstáculos a ser superado é a cultura das pessoas envolvidas. A implantação de um sistema de trabalho novo costuma mexer diretamente com a zona de conforto de cada envolvido podendo gerar descontentamento.

Umas das formas de resolver esse problema seria a criação de um sistema de comunicação transparente que tem como objetivo preparar os envolvidos para o inicio do novo método de trabalho e assim fazer com que as pessoas busquem informações e dêem possíveis sugestões.

A fim de contextualizar os problemas citados acima este estudo ira também mostrar de forma bem resumida alguns dos métodos de trabalho utilizados no desenvolvimento de software.

1.1 JUSTIFICATIVA

Hoje existe grande resistencia na organização e na normatização de métodos de trabalho em empresas que vivem o caos. Este projeto visa demonstrar que é possível melhorar todo o processo, desde o desenvolvimento e qualidade de software ao atendimento a cliente. A melhoria do ambiante de trabalho e na

qualidade do produto conseguidas, justificam o tema proposto uma vez que utilizando uma metodologia é possível gerenciar de forma prática e eficaz os processos do dia-a-dia de uma indústria de software.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Melhorar o atendimento a cliente e desenvolvimento de novos produtos em uma indústria de software.

1.2.2 Objetivos específicos

Identificar problemas no atendimento a cliente e desenvolvimento de novos produtos;

Identificar indicadores de desempenho que a empresa utiliza;

Apontar possíveis melhorias nos problemas identificados;

Analisar métodos ágeis de engenharia de software existentes;

Escolher e implantar um método ágil de engenharia de software;

Analisar impacto do novo sistema de trabalho sobre o sistema atual.

1.3 ABRANGÊNCIA

A solução compreende a implantação de um método ágil para ajudar na organização de uma empresa de software que atualmente vivencia o caos, desta forma criar uma cultura de organização para o desenvolvimento de novos produtos dentro desta empresa.

1.4 METODOLOGIA

O trabalho será realizado em uma empresa de software para onde se pretende implantar um método de trabalho que permita uma maior organização. Serão feitas pesquisas bibliográficas e de campo junto à empresa e clientes para analisar a melhor forma de implantação do projeto.

1.5 ESTRUTURA DO TRABALHO

O presente trabalho encontra-se dividido em sete capítulos.

O primeiro capítulo trata das pesquisas feitas para verificar a viabilidade do projeto, assim como seus objetivos, abrangência e metodologia a ser trabalhada.

O segundo capítulo apresenta um histórico sobre o setor de TIC (tecnologia da informação e telecomunicação) em Santa Catarina e como surgiu o Software e a Engenharia de Software.

O terceiro capitulo abrange como são os métodos de desenvolvimento de software tradicionais e ágeis de forma genérica e quais são os componentes do mesmo.

No quarto capítulo, apresenta-se como funciona uma indústria de software que costuma trabalhar no caos.

O quinto capítulo trata da implantação do método ágil na empresa utilizada como caso de uso neste trabalho.

O sexto capitulo, trata da conclusão deste trabalho, onde será descrito se foram obtidos melhorias no processo de trabalho implantado na empresa utilizada como estudo de caso.

2 HISTORICO

2.1 O SETOR DE TIC EM SANTA CATARINA

Segundo IEL/SC (2007) o setor de TIC (tecnologia da informação e comunicação) surgiu no final da década de 60 com a empresa CETIL, na cidade de Blumenau. O CETIL foi criado inicialmente para ser o Centro de Processamento de dados das indústrias têxteis da região, mas transformou-se no maior *bureau* de serviços do pais, tendo abrangência em todo território nacional.

Com o surgimento da microinformática, o CETIL passa a perder espaço no mercado para empresas de software voltadas para computadores pessoais. Grande parte das empresas surgidas na época foram criadas por ex-funcionários do próprio CETIL, como a WK Informática, a PC Auxiliar e a Fácil Informática. Com isto é criado o primeiro pólo de TIC com ênfase em software na região de Blumenau.

Em 1984, foi criada em Florianópolis, a Fundação CERTI, que estabelece em 1986 uma incubadora de empresas de base tecnológica, atualmente conhecida por CELTA (Centro Empresarial para Laboração de Tecnologias Avançadas). Por meio desta incubadora foram criadas diversas empresas da área de TIC que contribuíram para a formação de um pólo de tecnologia em Florianópolis.

Após o surgimento destes dois grandes pólos o setor passa a se organizar, e assim surgem as primeiras associações de empresas de tecnologia.

- Em 1986 é criada a ACATE em Florianópolis;
- O BLUSOFT é fundado em 1992, em Blumenau;
- Em 1995 é formalizada a SOFTVILLE em Joinville.

Com a criação destas associações o setor passa a se fortalecer e aumenta significativamente a competitividade das empresas de tecnologia catarinenses. O setor de TIC tem um grande crescimento a partir do apoio do programa SOFTEX, cujo objetivo era ampliar as exportações brasileiras de software, são criados três núcleos: Blumenau (1992), Joinville (1993) e Florianópolis (1994).

Com isto, Santa Catarina passou a ser o único estado do Brasil com três núcleos SOFTEX, demonstrando a força do setor de TIC catarinense. Um pouco mais adiante também foram criados diversos centros de tecnologia e incubadoras localizados em algumas regiões de Santa Catarina, como, o TEKNOPARK em Rio do Sul (1997), o MIDI-Tecnológico em Florianópolis (1998), o MIDI-VILLE em Joinville (1999), o MIDI-SUL em Criciúma (2001) e o MIDI-OESTE em Chapecó (2002).

Após o surgimento destes centros de tecnologia e associações de empresas, Santa Catarina começa a ser vista pelo governo estadual e federal como um grande pólo tecnológico, com isto passam a surgir investimentos e linhas de crédito para fortalecer ainda mais este setor do estado.

2.2 SOFTWARE

Segundo PRESSMAN (1995) durante as três primeiras décadas da era do computador, o desafio principal era desenvolver um hardware que permitisse um processamento e armazenamento de dados por um menor custo. Ao longo da década de 1980 isto foi possível graças a avanços significativos na área de microeletrônica. Hoje o mesmo problema se depara no desenvolvimento de software, melhorar a qualidade e baixar custos.

Nas ultimas décadas o software ultrapassou o hardware, ele passa a ser um diferencial para dirigir um negocio. A gama de informações que um sistema de gestão pode disponibilizar diferenciam uma empresa de seus concorrentes, então o software pode fazer a diferença.

2.2.1 Desenvolvimento de Software

Para PRESSMAN (1995) no inicio da era do computador, os sistemas baseados em computador eram desenvolvidos pela administração orientada a hardware. Os gerentes de projetos estavam no hardware, pois este era o item de maior custo no desenvolvimento de um sistema. Para controlar os custos dos hardware os gerentes instituíram formulas, métodos e padrões técnicos, que eram cuidadosamente colocados em pratica na analise e projeto antes que alguma coisa fosse construída.

No inicio a programação era vista como uma “forma de arte”. Existiam poucos métodos formais e quase não existia mão de obra. O programador freqüentemente aprendia seu oficio na tentativa e erro. O desenvolvimento de software era virtualmente indisciplinado e muitos profissionais adoravam isso.

Hoje a distribuição dos custos no desenvolvimento de sistemas baseados em computador mudou drasticamente. O Software é o item de maior custo e o hardware passa a ser secundário, e com isto surgem os seguintes questionamentos:

- Por que demora tanto tempo para que os programas sejam concluídos?
- Por que os custos são tão elevados?
- Por que não descobrimos todos os erros antes de entregarmos o software aos nosso clientes?
- Por que temos dificuldade em medir o progresso enquanto o software esta sendo desenvolvido?

Estas são algumas perguntas que levaram a adoção de métodos e praticas que criaram a engenharia de software.

2.2.2 Características do Software

Segundo PRESSMAN (1995) o software é um sistema lógico, e não físico. Portanto, o software tem características que são consideravelmente diferentes das do hardware.

2.2.2.1 Desenvolvimento de software e manufatura

Segundo PRESSMAN (1995) existem algumas semelhanças entre o desenvolvimento de software e a manufatura, em ambas atividades a alta qualidade é conseguida mediante a um bom projeto, mas durante o processo de manufatura pode introduzir problemas de qualidade, enquanto que no software estes problemas inexistem (ou podem ser facilmente corrigidos). Ambas atividades dependente de pessoas, mas o fator de qualidade no produto final da manufatura pode ser uma maquina, enquanto no software são as pessoas.

2.2.2.2 Software não se "desgasta".

Abaixo temos a Figura 1, que mostra o índice de falhas em função do tempo para um produto manufaturado. A curva indica que no inicio do ciclo de vida do produto existe um grande numero de falhas, (estas são geralmente atribuídas a erros de projeto e falhas durante o processo de criação) os defeitos são corrigidos e durante um certo período de tempo o produto não apresenta mais problemas. A medida que o tempo passa o índice de falhas eleva-se novamente devido ao desgaste do produto (ocorrido pelo ambiente onde esta inserido, pelo uso constante, etc).

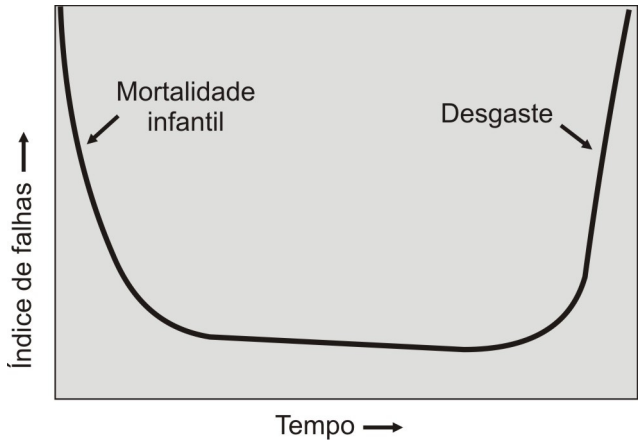


Figura 1 – Curva de falhas para o hardware.
Fonte: Pressman (1995, p.14).

O software não é sensível ao ambiente onde esta inserido, o que indica que seu desgaste não é como de um produto manufaturado. Na Figura 2 é possível observar como seria curva de desgaste do software. Defeitos não descobertos provocaram alto índice de falhas logo no começo de sua vida, porem, esses são corrigidos (partindo do pressuposto que estas correções não introduzam novas falhas), e a curva achata-se. Esta curva é uma forma grosseira de demonstrar o desgaste do software, pois segundo ela o software não se desgasta, que não é uma verdade.

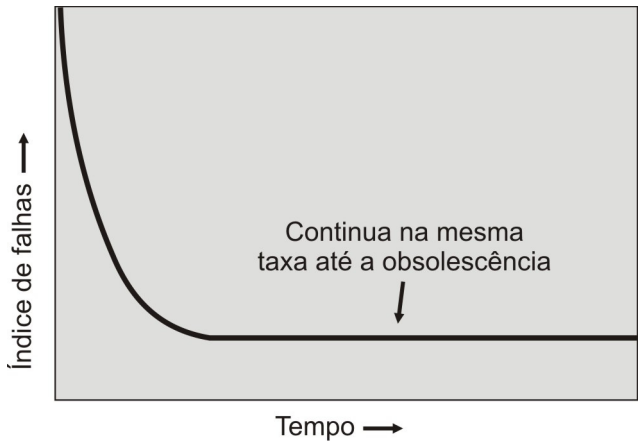


Figura 2 – Curva de falhas do software (idealizado).
Fonte: Pressman (1995, p.15).

A aparente contradição encontrada pode ser explicada pela Figura 3. Durante o ciclo de vida do software ele vai sofrendo modificações (manutenção),

mudanças na legislação, novas funcionalidades, etc. Quando são realizadas estas modificações é provável que uma série de falhas iram aparecer, fazendo com que a curva de índice de falhas apresente picos, e antes que a curva estabilize são solicitadas novas modificações tornando um clico continuo e aumentando o número de possíveis falhas, deste modo o software vai se deteriorando.

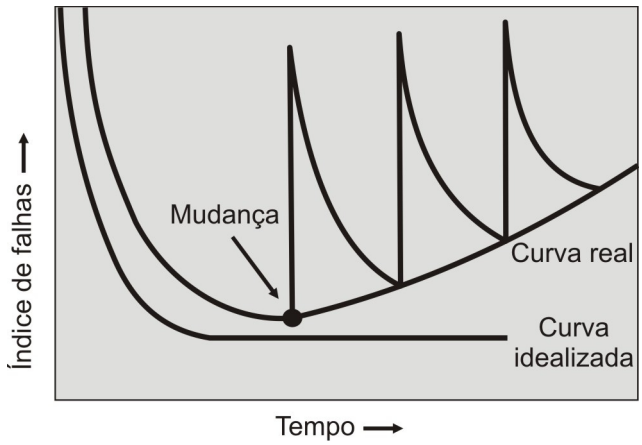


Figura 3 – Curva de falhas real para o software.
Fonte: Pressman (1995, p.15).

2.2.3 Aplicações do Software

Para PRESSMAN (1995) software é uma ferramenta que pode ser aplicada as mais diversas situações que possuam um processo a ser seguido, este processo dentro de um software pode ser chamado de algoritmo. Software abrange desde um sistema comercial, que possui uma série de regras de negocio e banco de dados altamente estruturado, até o controle de maquinas automatizadas, onde pode possuir um software especifico para executar uma única função. Logo em seguida temos de forma macro as aplicações em potenciais.

Software Básico: é uma coleção de aplicativos que dão suporte a outros programas. Compiladores, editores, componentes do sistema operacional, *drives*, são exemplos de software básico. Este tipo de software exige estruturas complexas que permitam o uso intenso por múltiplos usuários, escalonamento, operações

concorrentes, compartilhamento de recursos e sofisticada administração dos processos.

Software de Tempo Real: como o nome diz, este software controla eventos do mundo real. O software de tempo real pode coletar dados de sistemas externos, formatar informações de outros ambientes, enviar dados para outras aplicações, monitorar e coordenar outros componentes. O sistema de tempo real deve responder conforme suas restrições, pois caso não responda no tempo determinado seus resultados podem ser desastrosos. Este tipo de software é utilizado em sistemas que devem ser altamente confiáveis, por exemplo, sistema para controle de reator de usina nuclear.

Software Comercial: esta é a maior área de software existente, são aplicações destinadas à operacionalização e gestão de empresas. Esta família de software pode ser dividida em diversos seguimentos, como, varejo, indústria, contabilidade, RH, etc. Estas aplicações demonstram grande importância nas organizações, pois seu uso se torna indispensável para a tomada de decisão.

Software Cientifico e de Engenharia: e de Engenharia: são softwares basicamente formados por algoritmos para processamento de números. Estes são aplicados nos mais diversos estudos, vão desde astronomia a engenharia de automóveis. Muitos destes softwares utilizam sistemas de tempo real e sistemas básicos como parte de seus componentes.

Software Embutido (*embedded software*): são produtos inteligentes, e estão nos mais diversos equipamentos eletrônicos do nosso dia-dia. São utilizados muitas vezes para desempenhas funções especificas como, menu de funções de uma televisão ou controle de um microondas. A industria de eletrônicos esta a cada dia desenvolvendo novos softwares embutidos com cada vez mais recursos.

Software de Computador Pessoal: estes softwares são planilhas eletrônicas, editores de texto, gerenciadores de e-mails, jogos (este seguimento esta a cada dia mais forte), aplicativos multimídia (som e imagem), redes de relacionamento. Neste seguimento costumam surgir as interfaces mais inovadoras para seres humanos.

> ..., o software de computador pessoal continua a representar os mais inovadores projetos de interface com seres humanos de toda a indústria de software. PRESSMAN (1995).

Software de Inteligência Artificial (*Artifieial Intelligency - AI*): são software de algoritmos não numéricos utilizados para resolver problemas que não sejam compatíveis à computação e analise direta. Nesta área existem sistemas utilizados para reconhecimento de padrões simples (imagem e voz), jogos, demonstração de teoremas e reconhecimento de padrões complexos (este se utiliza de uma estrutura chamada *redes neurais artificiais*, onde o software é capaz de reconhecer padrões através da aprendizagem, utilizando-se de "experiências passadas").

2.3 ENGENHARIA DE SOFTWARE

Para PETERS (1999) atualmente a engenharia de software vem cada vez mais conquistando o seu espaço no mundo. Com o surgimento continuo de novas ferramentas, tecnologias, linguagens de programação, a própria evolução de conhecimentos dos stakholders (interessados, usuários, cliente) que a cada dia solicitam novos requisitos de software, a ES (Engenharia de Software) se torna indispensável para a tomada de decisão e para o sucesso do projeto.

Segundo PRESSMAN (1995) a engenharia de software é uma série de métodos, ferramentas e procedimentos que permitem o desenvolvimento de software de forma sustentável e racional.

Métodos: indicam os detalhes de como construir um software e introduzem um conjunto de critérios para a qualidade de software. Os métodos constituem um conjunto de tarefas como: planejamento, estimativa, analise de requisitos, desenvolvimento, validação, testes e manutenção.

Ferramentas: As ferramentas de engenharia de software proporcionam o apoio aos métodos. Atualmente existem ferramentas para todos os métodos informados anteriormente.

Procedimentos: Estes fazem a ligação entre os métodos e as ferramentas. Os procedimentos indicam a seqüência em que os métodos serão executados e os produtos que deverão ser entregues (relatórios, documentos, formulários, etc.), nesta seqüência de métodos entram os controles (ferramentas) que garantem a qualidade e que os métodos sejam executados como previsto.

2.3.1 Modelos de Ciclo de Vida do Software

Com base em PESSMAN (1995), PETERS (1999) e SOMMERVILE (2003) os modelos de ciclo de vida do software fornecem visão madura de como deveria ser a seqüência ideal de execução dos métodos propostos pela engenharia de software. Os ciclos como, cascata, incremental, espiral e prototipação são processos que possuem uma maturidade alcançada com base de conhecimento e experiências já vivenciadas pela indústria de software. Abaixo segue uma breve explanação sobre os ciclos de vida citados.

> A modelagem de processos de software alcançou uma fase de maturidade com base no conhecimento e experiência adquirida na produção de grande-escala de projetos software. PETERS (1999).

2.3.1.1 Cascata

Para PETERS (1999) o modelo cascata é o mais antigo de todos, em seu formato original descreve uma seqüência de atividades do ciclo de vida de um software, iniciando pela concepção e finalizando numa possível substituição (Figura

4). A concepção e os requisitos têm por objetivo nos mostrarem *o que* conhecemos da solução que pretendemos desenvolver. No projeto e implementação os desenvolvedores passam a se preocupar *como* o sistema será construído de forma que funcione e possua qualidade. Já nas ultimas etapas do modelo (Teste, Liberação, Manutenção e Substituição) o ponto mais importante é a *operação* do sistema. Após cada atividade do ciclo é fornecido um feedback para as fases anteriores, com isto possíveis problema encontrados podem ser corrigidos e evitados em outros projetos.

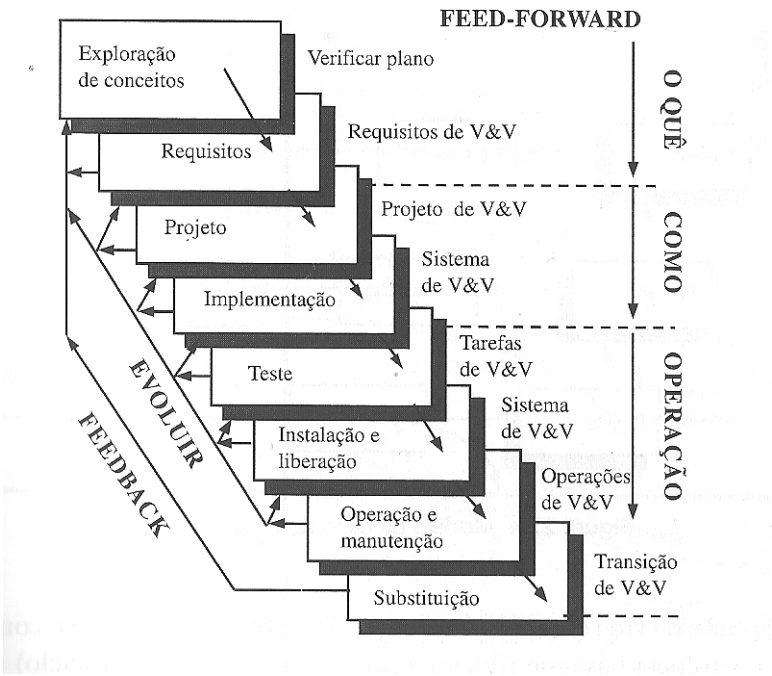


Figura 4 – Modelo de processo de software em cascata.
Fonte: Peters (1999, p.41).

Este modelo possui uma grande vantagem que é identificação e produção de um conjunto de documentos com as especificações e resultados de cada fase do ciclo de vida, isto permite um acompanhamento do andamento do projeto de forma clara, mas em contrapartida este modelo possui algumas desvantagens.

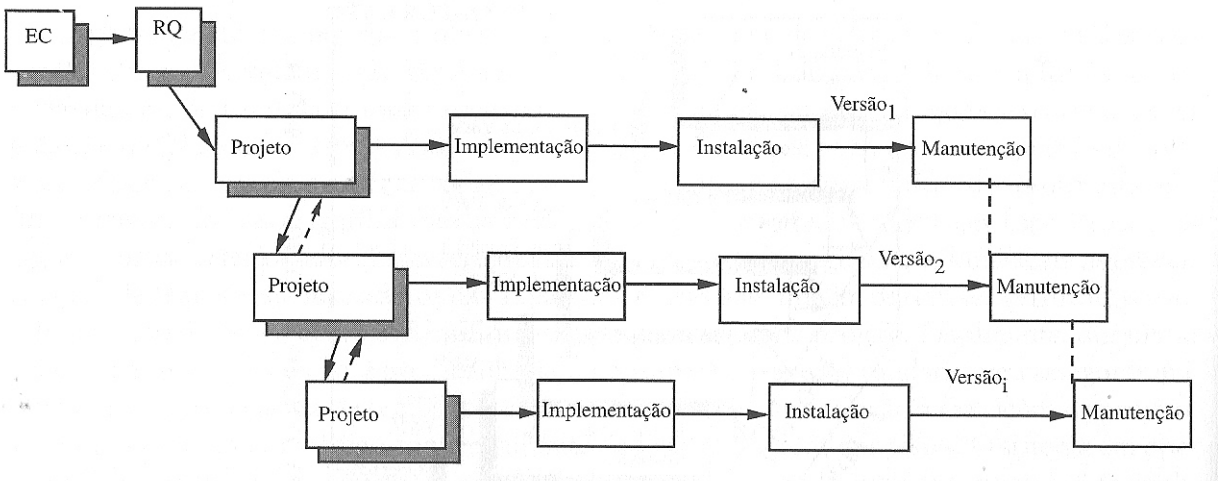
- O cliente terá que esperar até a entrega do projeto para ver como o sistema funciona, dependendo do tamanho e complexidade a entrega pode levar um tempo considerável.
- Com este modelo não é capaz de efetuar engenharia reversa em um sistema existente. Engenharia reversa é um processo do qual é

analisado e identificado componentes e funcionalidade de um sistema existente.

- O cliente deve especificar todas as características na primeira fase do ciclo, dependendo da complexidade raramente o cliente conseguirá repassar todas as características desejadas no projeto.

2.3.1.2 Incremental

O modelo incremental é semelhante ao cascata (Figura 5), segundo PETERS (1999) este surgiu para minimizar o retrabalho gerado no modelo cascata, este retrabalho é causado pelo fato de o cliente não conseguir especificar tudo que deseja e no momento da entrega o projeto não atende suas necessidades.
Neste modelo os requisitos e especificações são primeiramente identificados, e após isto as demais etapas são executadas em cada momento que é necessário o lançamento de uma nova versão do software. Este modelo parte do principio irreal de que os requisitos e especificações permanecem estáveis, porem, os requisitos tendem a evoluir devido às novas necessidades e experiências com o cliente e também com o surgimento de novas tecnologias.



Fonte: Peters (1999, p.42).

No modelo incremental podem ser definidas pequenas entregas de software para o cliente (versões), em cada uma delas o usuário final pode realizar sua validação e desta forma possíveis problemas pode ser corrigidos e entregues na próxima versão. Este processo de desenvolvimento possui diversas vantagens.

- O cliente não precisa esperar que todo o software seja finalizado para que possa utilizado. Já na primeira entrega é definido o básico para tornar o sistema operacional e desta forma o cliente já poderá operá-lo.
- Com base na validação realizada pelo cliente, é possível com a experiência obtida realizar possíveis melhorias para as próximas liberações.
- O risco de fracasso do projeto é minimizado devido a constante iteração com o cliente.
- Como as funções mais importantes são entregues nas primeiras liberações e durante o processo é provável que surjam incrementos, mas mesmas, os clientes tentem a não encontrarem falhas nas partes mais importantes, pois devido à evolução das mesmas elas serão testadas em cada nova liberação.

2.3.1.3 Prototipação

O modelo de prototipação (Figura 6) surgiu com o intuito de possibilitar um desenvolvimento rápido de produtos de software e também para mostrar ao cliente como ficaria o software desejado com base nos requisitos e objetivos definidos no inicio do projeto. Para PETERS (1999) os protótipos vão desde um esboço realizado em uma simples folha de papel a um software que demonstre como ficaria o produto final. Para situações onde há grande interação homem-máquina seria ideal a abordagem de prototipação para sucesso do projeto.

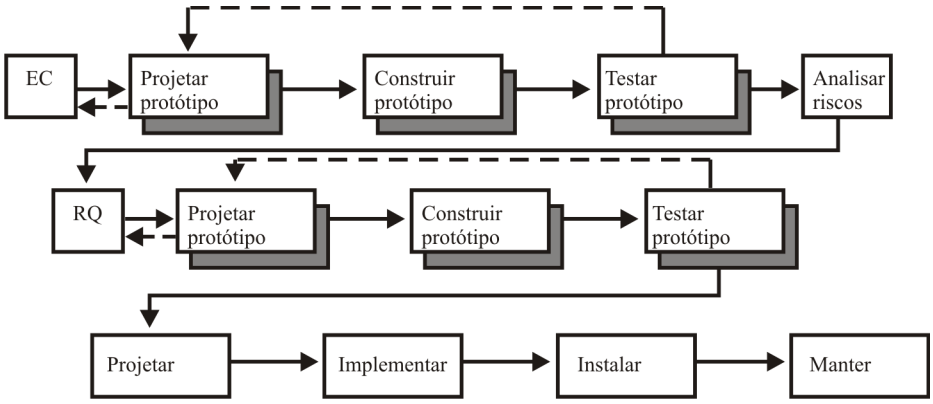


Figura 6 – Ciclo de vida de software de prototipação.
Fonte: Peters (1999, p.47).

Uma das principais vantagens do modelo de prototipação é a possibilidade de obter feedback dos clientes, com isto é possível minimizar risco de que projeto falhe. Também após cada feedback os requisitos e conceitos do sistema são alterados antes de iniciar a fase de construção do projeto.

2.3.1.4 Espiral

O modelo espiral foi desenvolvido com o intuito de utilizar as mesmas características do modelo cascata juntamente com o modelo de prototipação, acrescentados de um novo elemento a analise de riscos. Este modelo também não funciona de forma seqüencial com um retorno de uma atividade para outra, seu funcionamento o é em forma de espiral (Figura 7). Segundo PETERS (1999) cada etapa do processo de desenvolvimento pode estar atrelada a um loop da espiral, sendo assim, primeiro loop pode ser a analise de requisitos, o próximo pode ser o projeto do sistema, e assim por diante. A espiral é divida em quatro grandes setores, Definição de Objetivos, Analise de Riscos, Desenvolvimento e validação e Planejamento.

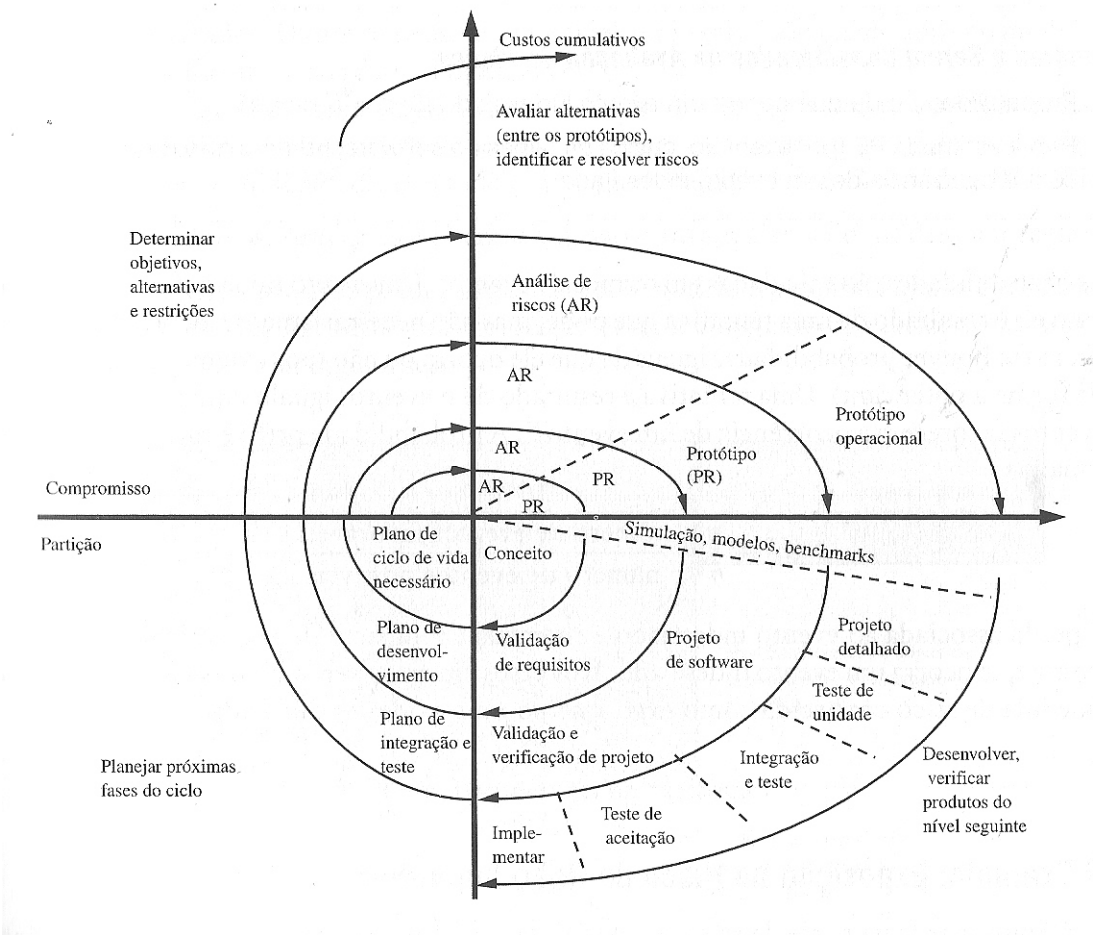


Figura 7 – Modelo de ciclo de vida em espiral.
Fonte: Peters (1999, p.43).

A distinção deste modelo com os demais é a explicita consideração dos ricos que podem evolver o projeto. A analise de riscos é importante para o gerenciamento de projetos, pois com ela é possível realizar uma tomada de decisão de forma planejada caso algo de errado, como estouro do prazo, aumento de custos, problemas com a tecnologia adotada, entre outros. O ciclo de vida espiral inicia com a elaboração dos objetivos e requisitos para o sistema, após esta etapa entra a analise de riscos, onde para todos os objetivos acertados no setor anterior são colocados possíveis restrições e estas são enumeradas em possibilidade de acontece e se acontecer qual o grau do problema. Na próxima etapa iniciasse o desenvolvimento do projeto, criação de protótipos, desenvolvimento do software, testes, liberação, após esta etapa realizasse o planejamento onde será definido o inicio da próxima fase do processo.

3 MODELOS TRADICIONAIS E AGEIS

3.1 MODELOS TRADICIONAIS

Segundo PRESSMAN (2006) os modelos tradicionais ou prescritivos, surgiram para colocar ordem no caos do desenvolvimento de software, e estes tem fornecido um roteiro razoavelmente efetivo para as equipes de desenvolvimento de software.

> Modelos prescritivos de processo definem um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalhos que são necessários para fazer engenharia de software com alta qualidade. Esses modelos de processo não são perfeitos, mas efetivamente fornecem um roteiro útil para o trabalho de engenharia de software. PRESSMAN (2006).

Os modelos citados a seguir fornecem uma visão genérica do processo de software e de quais artefatos devem ser gerados, mas o segredo esta na adaptação de todos os modelos para a realidade de desenvolvimento de seu projeto. Segue alguns modelos tradicionais.

- Cascata;
- Modelos Incrementais: Incremental e RAD;
- Modelos Evolucionários: Prototipação, Espiral, Concorrente;
- Modelos Especializados: Baseado em Componentes, Métodos Formais, Orientados a Aspecto;
- Processo Unificado (UP, RUP).

Alguns destes modelos já foram vistos no capitulo 2, pois fazem parte da concepção de engenharia de software, então será feito somente um breve descritivo do Processo Unificado que é uma tentativa de unificação das melhores praticas adotadas em cada modelo convencional.

3.1.1 Processo Unificado (UP, RUP)

Para PRESSMAN (2006) o processo unificado é uma tentativa de utilização das melhores praticas contidas em cada modelo convencional. Este processo reconhece a importância da comunicação com o cliente e o papel fundamental da arquitetura de software para possibilitar evolução e reuso. O processo unificado possui um fluxo que é iterativo e incremental, dando a sensação evolucionaria ao processo de desenvolvimento que é fundamental no mundo moderno de software.

O processo unificado surgiu no inicio da década de 1990, quando o desenvolvimento orientado a objetos estava em impasse de qual seria o melhor método para este tipo de desenvolvimento de software. Então James Rumbaugh, Grady Booch e Ivar Jacobson, iniciaram a criação de um “processo unificado” que reuniria o melhor de todos os processos existentes até o momento e adotaria características adicionais propostas por outros especialistas do ramo de OO (Orientação a Objetos), estas novas características deram origem a UML (*Unified Modeling Language* ou Linguagem Unificada de Modelagem), que contem uma notação robusta para a modelagem de projetos orientados a objetos.

3.1.1.1 Fases do processo unificado

O processo unificado possui cinco atividades genéricas, Concepção, Elaboração, Construção, Transição e Produção. Na figura 8 podemos ver as fases do processo unificado.

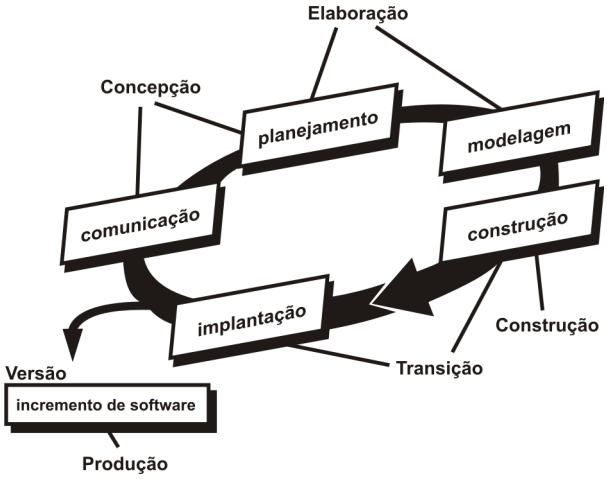


Figura 8 – Processo Unificado.
Fonte: Pressman (2006, p.52).

Na fase de *concepção*, é dado ênfase para a comunicação com o cliente e para o planejamento do projeto. Os requisitos de negócios para o software são recolhidos junto ao cliente e usuários finais e um plano de projeto é descrito. Os requisitos fundamentais são descritos em forma de casos de uso. Casos de uso, em linhas gerais, descrevem uma seqüência de ações que são realizadas por um ator (uma maquina, um usuário ou um sistema), estes permitem identificar o escopo e fornecem base para o planejamento do projeto. Após recolhido os requisitos é feito um planejamento, este identifica os recursos, avalia riscos, define cronograma e estabelece as bases para as próximas fases do projeto.

A fase de *elaboração* envolve o planejamento com o cliente e a atividade de modelagem de processos. Nesta fase os casos de uso são refinados e expandidos para cinco visões diferentes (modelo de casos de uso, modelo de analise, modelo de projeto, modelo de implementação e o modelo de implantação), em alguns casos também é criado um protótipo que demonstra uma viabilidade arquitetural do projeto, não fornecendo todas as características e funções do sistema. Na fase de elaboração o plano de projeto é revisto a todo momento para garantir que o escopo, riscos, entrega e custos permaneçam condizentes, aqui ainda são permitidas alterações no plano de projeto.

Na fase de *construção*, usando o modelo arquitetural como entrada os componentes de software são desenvolvidos ou adquiridos, desta forma os casos de uso dos usuários finais transformam-se em codificação. A cada componente desenvolvido são realizados testes unitários, ao passo que cada componente integra com outro, testes são realizados com base nesta integração.

A fase de *transição* abrange o final da atividade de construção e o inicio da implantação. Nesta fase o software é liberado para os usuários final onde todas as funcionalidades são testadas e através do *feedbak* o usuário repassa defeitos e alterações necessárias. Também nesta fase são criados manuais de usuário, guias de perguntas frequentes e procedimentos de instalação, ao final desta fase os incrementos de software necessários são realizados e assim é entregue uma versão final para o cliente.

A fase de *produção* é o final da implantação onde é realizado um acompanhamento junto aos usuários, é fornecido suporte para o ambiente de operação. Os defeitos e solicitações são recolhidos, submetidos e avaliados.

Ao final de cada fase são recolhidos os produtos que trabalho que são todos os artefatos (documentos, protótipos, software) gerados, na figura 9 é possível visualizar os artefatos necessários para cada uma das 4 fases. No PU ao passo que um incremento é analisado, construído e entregue, outros incrementos podem estar sendo analisados, construídos e entregues, isto indica que as fases do PU trabalha de forma concorrente e não sequencial.

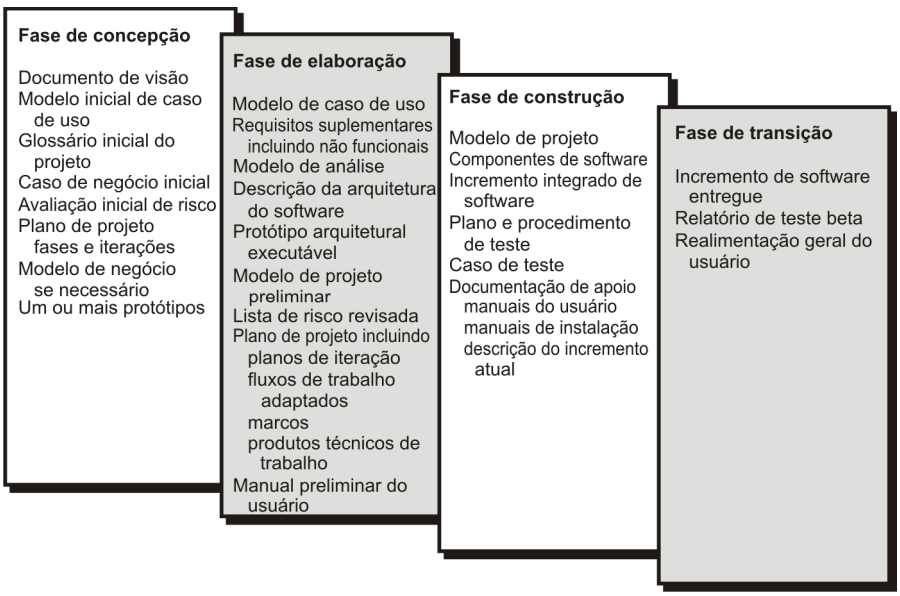


Figura 9 – Produtos de trabalho produzidos em cada fase do PU.
Fonte: Pressman (2006, p.54).

3.2 MODELOS AGEIS

Segundo PRESSMAN (2006) as idéias de um desenvolvimento ágil sempre estiveram junto aos engenheiros de software, mas somente na década de 1990 que estas idéias foram transformadas em um movimento. O desenvolvimento ágil possui um esforço grande para vencer as fraquezas percebidas e reais da engenharia de software convencional, ele possui muitos benefícios mas não é aplicado a todos os tipos de projetos, pessoas e situações. Em 2001 um grupo de notáveis desenvolvedores assinaram um manifesto que expressa toda idéia do desenvolvimento ágil, abaixo temos um trecho do mesmo.

> Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar:
>
> *Indivíduos e interação entre eles* mais que processos e ferramentas;
> *Software em funcionamento* mais que documentação abrangente;
> *Colaboração com o cliente* mais que negociação de contratos;
> *Responder a mudanças* mais que seguir um plano. MANIFESTO AGIL (2001)

3.2.1 Agilidade e Processo Ágil

Para PRESSMAN (2006) agilidade vai muito alem do que uma resposta rápida as mudanças ocorrentes no nosso dia-a-dia como legislação, novos processos e novas tecnologias que exigem modificações em nossos softwares. A agilidade engloba toda uma filosofia que o manifesto ágil tenta demonstrar, como a prioridade para a entrega do produto deixando um pouco de lado a documentação excessiva, adoção do cliente como parte do processo de desenvolvimento, a integração entre a equipe do projeto e um planejamento para o nosso mundo incerto

permitindo um plano de projeto flexível. Abaixo temos 12 princípios do desenvolvimento ágil.

1. Nossa maior prioridade é satisfazer ao cliente desde o inicio por meio de entrega continua de software valioso.
2. Modificações de requisitos são bem vindas, mesmo que tardias no desenvolvimento. Os processos ágeis aproveitam as modificações como vantagens para competitividade do cliente.
3. Entrega de softwares funcionando frequentemente, a cada duas semanas até dois meses, de preferência no menor espaço de tempo.
4. O pessoal de negocio e os desenvolvedores devem trabalhar juntos diariamente durante todo projeto.
5. Construção de projetos em torno de indivíduos motivados. Forneça-lhes o ambiente e apoio que precisam e confie que eles farão o trabalho.
6. O método mais eficiente e efetivo de levar informação para e dentro de uma equipe de desenvolvimento é a conversa face a face.
7. Software funcionamento é a melhor medida de progresso.
8. Processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante, indefinidamente.
9. Atenção continua à excelência técnica e ao bom projeto facilitam a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não efetuado – é essencial.
11. As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva, então sintoniza e ajusta adequadamente seu comportamento. MANIFESTO AGIL (2001)

A agilidade pode ser adaptada a qualquer processo, desde que a equipe esteja preparada a esta nova filosofia e que o projeto seja projetado para permitir um tipo de abordagem de entregas incremental que forneça o software funcionando ao cliente o mais rápido possível.

O processo ágil é baseado na adaptabilidade a mudanças no projeto, deste modo o processo de desenvolvimento deve ser incremental, sempre com entregas frequentes ao cliente e assim que é tido o *feedback* novas adaptações no

são realizadas no projeto e no processo. Os fatores humanos neste tipo de abordagem também são essenciais, para o desenvolvimento ágil a equipe deve possuir algumas características como competência, foco comum, colaboração, capacidade de tomada de decisão, habilidade de resolver problemas vagos, respeito e confiança mutua, auto-organização.

Competência: Talento inato, habilidades especificas relacionadas a software e conhecimento global do processo de trabalho. Habilidades e processos de trabalho devem ser repassadas a todos os membros da equipe.

Foco Comum: Mesmo pessoas trabalhando em partes do projeto diferentes o foco deve ser sempre o mesmo, a entrega dentro do prazo e a satisfação do cliente.

Colaboração: Criar um mecanismo que permite a comunicação entre toda equipe e o cliente, desta forma criasse um vinculo de comprometimento entre todas as partes envolvidas.

Capacidade de tomada de decisão: a equipe deve possuir mecanismos que permitam escolherem o seu próprio destino com autonomia.

Habilidade de resolver problemas vagos: Os gerentes de software devem reconhecer que na equipe ágil existira ambiguidade continuamente devido a modificações continuas. A equipe deve ter consciência que qualquer problema deve ser visto como lições aprendidas para no futuro servirem de forma benéfica.

Respeito e confiança mutua: A equipe ágil deve ser unida ao ponto de gerar respeito e confiança entre todos os envolvidos, desta forma a equipe é vista pela soma de todos e não individualmente.

Auto-organização: A equipe ágil deve ser capaz de gerir seu cronograma, ambiente de trabalho e processo de desenvolvimento.

Na sequencia teremos uma visão geral de alguns modelos de desenvolvimento ágil e suas principais características.

3.2.1.1 Extreme program (XP)

Para PRESSMAN (2006) as idéias e métodos adotados no XP foram criados no final da década de 1980, mas somente em 1999 que foi publicado por Kent Beck. O XP utiliza uma abordagem de desenvolvimento orientada a objetos como paradigma predileto. Este método utiliza-se das seguintes atividades, planejamento, projeto, codificação e testes. Na figura 10 é possível visualizar o processo XP.

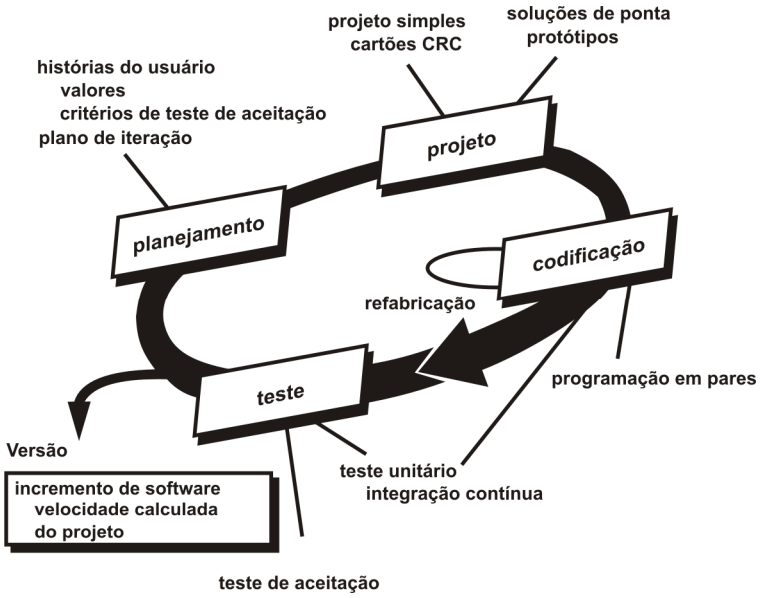


Figura 10 – Processo extreme program.
Fonte: Pressman (2006, p.64).

Na atividade de *planejamento* é solicitado ao cliente para que ele escreve histórias, estas são breves descrições das funções que ele deseja que o software possua, estas mesmas são acrescidas de um número que define a prioridade com que o cliente vê aquela historia, após isto, a equipe de desenvolvimento estima o custo e prazo em semanas para que aquela historia seja desenvolvida, caso ultrapasse 3 semanas a mesma é repassada novamente ao cliente para que ele transforme-a em historias menores. Vale lembrar que novas histórias podem ser descritas a qualquer momento.

Após a definição das historias, a equipe entra em contato com o cliente para definir como elas serão agrupadas e quais incrementos de software serão

entregues primeiro. Os incrementos podem ser entregues de três formas. 1 – Todas as historias serão entregues no menor tempo possível; 2 – As historias com valor mais alto são antecipadas e implementadas primeiro; 3 – As historias que possuem o maior risco serão entregues primeiro. Assim que a primeira versão é entregue a equipe XP calcula o tempo de desenvolvimento médio com base no número de historias que foram desenvolvidas, com isto ela consegue ter uma base para de tempo para as próximas entregas.

Na fase de *projeto* um principio é seguido rigorosamente, o KIS (*Keep it Simple* – Mantenha a Simplicidade). É sempre preferível um projeto simples do que algo mais complexo, alem disso, o projeto da as diretrizes para a implementação das historias, funcionalidades não são encorajadas. O XP utiliza cartões CRC (*Class-Responsibility-Colaborator*), estes identificam e ajudam na organização das classes e objetos. Os cartões CRC são o único artefato gerado na fase de projeto XP.

Caso seja encontrado alguma historia que possua uma complexidade elevada, o XP recomenda a criação imediata de um protótipo que é analisado e avaliado, desta forma é possível minimizar o risco de que ocorra uma falha.

Na atividade de *codificação* os desenvolvedores inicialmente realizam testes unitários nas historias desenvolvidas pelo cliente, desta forma ele fica mais preparado para poder iniciar a codificação do software, assim que liberadas as primeiras verões, novos testes unitários são realizados fornecendo assim um *feedback* para os desenvolvedores. Um dos conceitos mais comentados do XP é a *programação em pares*, recomenda-se que cada historia seja escrita por dois desenvolvedores em uma única estação de trabalho, desta forma uma garantia de qualidade e de resolução de problemas pode ser conseguida em tempo real (duas cabeças frequentemente são melhor que uma). Nesta configuração enquanto um desenvolvedor se preocupa com uma parte especifica da historia, outro garante as normas e qualidade do processo.

A fase de *testes* da destaque para os testes unitários que podem ser realizados no momento da codificação e após isto podem ser realizados os testes de integração entre as historias, desta forma é possível verificar se o software esta se deteriorando. Os testes de aceitação a validação de regras de negocio são especificados ou realizadas pelos clientes, e avaliam de forma global o software com base na integração entre as histórias.

3.2.1.2 Método de desenvolvimento dinâmico de sistemas (DSDM – Dynamic Systems Development Method)

Segundo PRESSMAN (2006) o DSMS fornece um processo voltado a projetos que possuem restrição de prazo apertadas e que permita o uso da prototipagem incremental em um ambiente controlado por projeto. O DSMS sugere uma filosofia que é “emprestada” do principio de Pareto, onde 80% do software será entregue em 20% do prazo estabelecido.

Como o XP o DSDM sugere um processo iterativo de software, mas a cada iteração é levado em conta a regra dos 80%, isto significa, que somente um certo esforço será realizado para permitir o avanço do próximo incremento. Os detalhes restantes serão desenvolvidos assim que mais requisitos de negocio forem conhecidos ou acomodados.

As praticas do DSDM são mantidas pelo DSDM Consortium (http://www.dsdm.org), este é um grupo mundial de empresas que definiu o modelo ágil de processo, chamado *ciclo de vida DSDM* , este possui três ciclos iterativos, precedidos por duas atividades adicionais.

Estudo de viabilidade: estabelece os requisitos e as restrições necessárias para a o projeto em construção e define se esta aderente ao DSDM.

Estudo de negocio: estabelece os requisitos funcionais e não funcionais, que permitiram valorar o negocio. Alem disto define a arquitetura básica e identifica os fatores de manutenibilidade do projeto.

Iteração do modelo funcional: Fornecem um conjunto de protótipos que são avaliados pelo cliente, deste modo forçando a indicação de novos requisitos e obtenção de um feedback dos usuários.

Iteração de projeto e construção: Revisa os protótipos desenvolvidos na iteração do modelo funcional na visão de engenharia e permite a adição de valor de negocio pelo usuário. Esta etapa geralmente ocorre simultaneamente com a iteração do modelo funcional.

Implementação: Transforma o protótipo em um produto, lembrando que este não deve estar 100% completo ou modificações podem ser acomodadas a medica que cada incremento é operacionalizado. Após isto o DSDM volta ao processo de iteração do modelo funcional.

O processo DSDM pode ser combinado com praticas do modelo XP, utilizando técnicas e ferramentas que viabilizem a construção incremental do projeto.

3.2.1.3 Scrum

Segundo PRESSMAN (2006) o Scrum (nome deriva de uma atividade que ocorre no jogo de *rugby*) foi desenvolvido no inicio da década de 1990 por Jeff Sutherland e sua equipe, nos últimos anos foram adicionadas novas praticas ao scrum por Scwaber e Beedle, e este esta totalmente aderente ao manifesto ágil. Abaixo temos algumas características deste modelo.

- Neste modelo pequenas equipes são criadas a fim de maximizar a comunicação e o comprometimento, desta forma minimizando a necessidade de supervisão;
- Deve ser possível a adaptação do processo a qualquer momento garantindo assim um melhor produto entregue;
- O processo produz pequenos incrementos que podem ser documentado, testados, entregues e expandidos;
- Todos as especificações são divididas em partes claras permitindo um desenvolvimento em “pacotes”;
- Testes e documentação são geradas ao passo que os incrementos são desenvolvidos;
- O Scrum fornece a habilidade de liberar o produto a qualquer momento. Porque o cliente solicitou, porque a empresa precisa de dinheiro, porque o concorrente liberou primeiro, etc.

O processo de Scrum incorpora as seguintes atividades no processo de desenvolvimento de software, requisitos, análise, projeto, evolução e entrega. Todas as atividades do Scrum são realizadas em um padrão de processo chamado *Sprint*. O número de sprints dentro de cada atividade pode variar conforme a complexidade do produto a ser produzido, estas são definidas e modificadas pela equipe Scrum normalmente em tempo real. A figura 11 demonstra este processo.

O Scrum enfatiza o uso de um conjunto de padrões de processos que permitem a entrega de produtos com prazo apertado, requisitos que modificam constantemente e regras de negocio criticas.

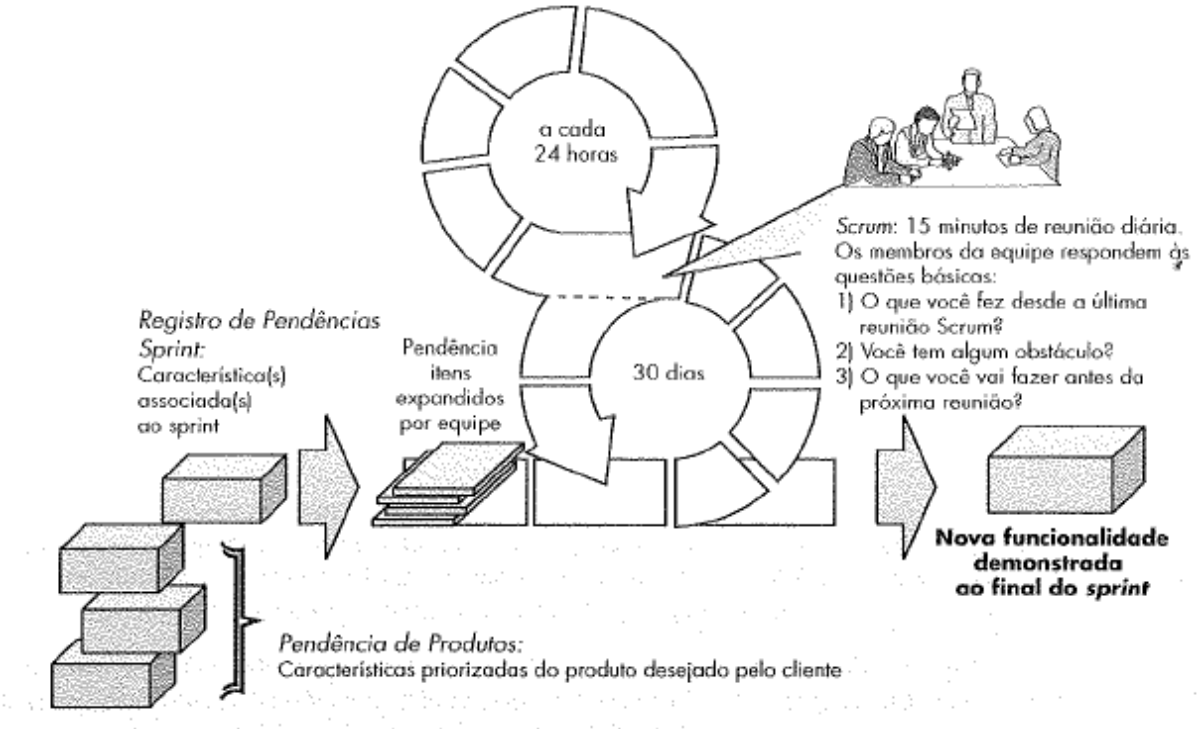


Figura 11 – Fluxo de processo scrum.
Fonte: Pressman (2006, p.70).

Pendência: Uma lista de requisitos e características priorizada conforme o valor de negocio para o cliente. Nesta lista podem ser adicionados novas pendências conforme necessário e o gerente a qualquer momento pode modificar a prioridade das mesmas.

Sprints: Unidades de trabalho que são necessárias para concluir um requisito definido na lista de pendências e que precisa ser entregue em um intervalo de tempo

pré-definido. Durante um sprint os itens de pendências trabalhados são congelados, desta forma, não é possível introduzir modificações durante uma sprint com isto os desenvolvedores conseguem trabalhar em um ambiente de curto prazo e estável.

Reuniões Scrum: Estas são realizadas diariamente, preferencialmente em pé e de curta duração (em torno de 15 minutos). Três pontos são enfatizados com a equipe Scrum. O que você fez desde a ultima reunião? Que obstáculos você encontrou? O que você planeja realizar até a próxima reunião?

As reuniões Scrum geralmente são lideradas pelo Scrum Master. Em linhas gerais o Scrum Master é o líder de uma equipe Scrum. Estas reuniões permitem a equipe descobrir problemas em potencial e saná-los no mais curto espaço de tempo possível, também ela é responsável por socializar o conhecimento entre os membros da equipe, promovendo assim um ambiente de aprendizagem e auto-organização.

Demos: Entregas constantes de versões de software que permite o cliente avaliar, testar e operacionalizar, não necessariamente precisa estar com todas as funcionalidades finalizadas, o importante é que elas sejam entregues dentro do prazo estabelecido.

Para BEEDLE (1999) os padrões de processo Scrum permitem a equipe de desenvolvimento de software trabalhar em um mundo de incertezas de modo bem sucedido, pois a equipe admite a existência do caos. “O Scrum considera antecipadamente a existência do caos...” BEEDLE (1999)

3.2.1.4 Crystal

Crystal é tido como uma *família de métodos ágeis*, esta definição foi criada por Alistair Cockburn e Jim Highsmith a fim de conseguir uma “manobrabilidade” que é segundo COCKBURN(2002) caracterizada por um “jogo cooperativo de invenção e comunicação de recursos ilimitados, com o principal

objetivo de entregar softwares prontos e funcionando e com o objetivo secundário de preparar-se para o jogo seguinte”

A família crystal é na verdade um conjunto de praticas apoiadas por, metodologias cada uma com seu elemento central, papéis, padrões de processo, produtos de trabalhos e praticas especificas a cada uma delas. Este conjunto de praticas pode ser adaptada para diferentes tipos de projetos, o objetivo é que uma equipe ágil escolha a melhor pratica para ser adotada no seu projeto. Existe uma comunidade que possui uma discussão mais abrangente sobre o crystal, este pode ser conferido no endereço http://www.crystalmethodologies.org.

3.2.1.5 Desenvolvimento guiado por características (FDD – Feature Driven Development

O FDD foi concebido inicialmente como uma abordagem para a engenharia de software de projetos orientado a objetos por Peter Coad e sua equipe no final da década de 1990, e aprimorado por Stephen Palmer e John Felsing que descreveram como um processo ágil e adaptativo que pode ser utilizado em projetos de médio a grande porte. No contexto do FDD uma característica é qualquer função valorizada pelo cliente que pode ser entregue em duas semanas ou menos, segundo COAD (1999).

As características são pequenos blocos de funcionalidades passiveis de entrega, que permitem o usuário descrevê-las com facilidade e entender como elas relacionam-se umas com as outras. Elas podem ser agrupadas em nível hierárquico ou a nível de negocio também pelo fato de serem pequenas implantações são mais fáceis de gerir e inspecionar. O planejamento, cronograma e monitoração são guiados pela hierarquia das características em vez de um conjunto de tarefas de engenharia de software. O FDD possui cinco atividades em seu ciclo, que podem ser vistas na figura 12.

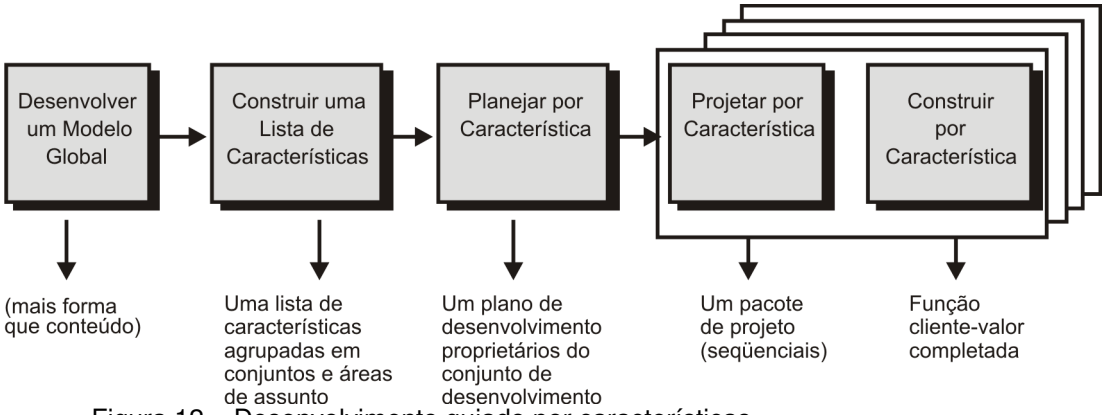


Figura 12 – Desenvolvimento guiado por características.
Fonte: Pressman (2006, p.72).

Este método ágil diferente dos demais, da mais ênfase em diretrizes técnicas de engenharia de software, uma vês que o projeto vai crescendo em tamanho e complexidade a gestão de projetos é acionada. Neste método é essencial que a equipe de desenvolvimento e clientes tenham ciência que avanços serão feitos e problemas serão sanados, caso a prazo de entrega ultrapasse o serviço é analisado se todos os incrementos foram colocados no cronograma. Para conseguir o acompanhamento do projeto o FDD utiliza-se de seis marcos com referencia, são eles, travessia do projeto, projeto, inspeção do projeto, código, inspeção do código, promoção para construção.

4 CAOS

O projeto proposto consiste na aplicação de uma metodologia ágil em um ambiente de caos, segundo o dicionário Aurélio – **1.** *Hist. Filos.* Nas mitologias e cosmologias pré-filosóficas, vazio obscuro e ilimitado que precede e propicia a geração do mundo; “Assim, ó Deus poderoso, ardente de vida, faz surgir do caos o homem, a mulher, os astros” (Graça Aranha, *A Estética da Vida*, PP. 52-52). **2.** Grande confusão ou desordem. – o dia-a-dia de uma indústria de software que vive o caos é exatamente como a definição apresentada.

Neste capitulo proponho identificar alguns problemas encontrados em uma empresa que vive o caos, com isto será possível elencar indicadores e eleger práticas de desenvolvimento que melhorem a organização da empresa.

4.1 A CAUSA DO CAOS

Conforme levantamento de dados realizado na empresa de software, estudo de caso neste projeto, e com alguns membros do núcleo de base tecnológica filiados a ACIC (Associação do Comercio e Indústria de Criciúma) o caos pode ter origem através de alguns fatores como os citados abaixo.

Empresas sem foco: algumas atendem qualquer tipo de seguimento, criando diversos produtos para atender vários tipos de clientes: desde uma indústria de grande porte a um simples comercio. Com isto exige-se equipes com conhecimento em diversos seguimentos, o que acaba dificultando o entendimento de solicitações realizadas por clientes e principalmente o atendimento dos mesmos.

Concorrência: existem desenvolvedores de software que não possuem nenhuma política de comercialização e criam produtos com um custo abaixo do mercado para o cliente, isto acaba “prostituindo” o mercado de software. As empresas pretendem praticar preços de implantação e manutenção que permita um atendimento de

qualidade e um produto confiável, mas devido ao fato de existirem estas práticas muitas acabam reduzindo os custos para poderem concorrer com estes desenvolvedores. A primeira redução ocorre no uso de metodologias, técnicas e ferramentas organizacionais.

Mão de obra: A grande falta de mão de obra qualificada acaba dificultando o dia-a-dia das empresas, principalmente as que sofrem com o problema de caixa, pois as grandes empresas acabam contratando estes funcionários que as pequenas empresas não conseguem manter devido ao custo dos salários e a inexistência de um plano de cargos e salários. Desta forma a contratação de um gerente de projetos ou analista com grande experiência que ajude na organização e documentação dos projetos acaba sendo uma idéia muito vaga, deste modo as empresas acabam contratando muitos estagiários que não possuem conhecimento e experiência necessária para o cargo.

Devidos aos problemas citados as empresas não conseguem encontrar um ponto de equilíbrio e implantar uma metodologia que ajude a organização. Estas por sua vez enxergam as metodologias como custo, mas o objetivo de boas práticas no desenvolvimento de software não é gerar custo e sim possibilitar um produto de qualidade que permita justificar o preço em relação a um concorrente que não possui nenhuma prática e desta forma gerar lucro para a empresa.

4.2 UMA EMPRESA NO CAOS

O dia-a-dia de uma empresa no caos costuma ser muito desgastante para toda a equipe. O problema começa com as solicitações dos clientes e do comercial da empresa que acaba prometendo muitas funcionalidades no software que não existem ou existem parcialmente somente para poder fechar um negocio. A seguir tem-se uma citação interessante sobre o caos no desenvolvimento de software.

> O cenário é contemporâneo, mas bem que poderia se tratar de mais uma batalha épica. De um lado, necessidades incompreendidas, especificações nebulosas, expectativas não-realistas, estimativas infundadas, quebras de comunicação, complexidades do domínio, conflitos de objetivos e mudanças à espreita, aguardando apenas a próxima oportunidade para o ataque. Do outro, equipes de desenvolvimento resistem, apoiadas em ferramentas (mesmo que artesanais), metodologias (mesmo que burocráticas ou genéricas demais) e em habilidades (mesmo que individuais) de seus membros. E o resultado desse confronto, por enquanto, não é nada encorajador... FURTADO (2005)

Grande parte das solicitações de modificações acabam sendo mal interpretadas e desenvolvidas sem nenhum acompanhamento, documentação, analise, cronograma, dificultando para o cliente e para o suporte técnico que é obrigado a entender o funcionamento do requisito sem nenhuma documentação e mesmo assim no momento que o cliente possuir alguma dúvida provavelmente o desenvolvedor será obrigado a atender o cliente, pois na empresa somente ele entende o que foi desenvolvido.

Também não existem normas de codificação, desta forma os requisitos acabam sendo desenvolvidos conforme a metodologia do programador, quando é necessário outro desenvolvedor verificar o código fonte acaba-se perdendo muito tempo até entender a codificação realizada anteriormente.

Algumas empresas também possuem a figura do “herói”, este é a pessoa que tem a capacidade de entender e resolver qualquer problema que possa surgir, geralmente sempre está com uma carga de trabalho excessiva, nunca tira férias e quando, por uma eventualidade, não está na empresa, tudo para.

4.2.1 Estudo de Caso

A Logosystem Sistemas possui ou já passou por todos os problemas descritos anteriormente. Essa empresa esta sendo utilizada como caso de estudo na implantação de boas práticas para resolução das suas principais preocupações, o atendimento ao cliente e o desenvolvimento de novos produtos. Estes dois itens

estão gerando preocupação para os gestores que definiram algumas metas para a empresa crescer e estão com muitas duvidas se isto será possível com a situação atual. Os indicadores que a empresa definiu até o final de 2010 eram: dobrar o faturamento e finalizar a conversão de tecnologia do software ERP que esta na versão 1 para a versão 2.

4.2.1.1 Atendimento ao cliente

A empresa trabalha com atendimento de clientes via telefone, MSN (Programa de mensagem instantânea da Microsoft) e em loco. Atualmente ela possui somente um atendente interno e três atendentes externos. O atendimento interno restringe-se a pequenas duvidas dos usuários e o atendimento externo restringe-se a treinamentos e implantação de sistema em novos clientes. A seguir temos os problemas encontrados.

Desenvolvedores dando suporte: Apesar de existir um atendente interno, este por sua vez acaba tento muita dificuldade na resolução de dúvidas referente a novas funcionalidades desenvolvidas, então por este fato, a equipe de desenvolvimento trabalha na mesma sala e o atendente acaba repassando as ligações. Esta situação esta gerando descontentamento na equipe de desenvolvedores que são quatro pessoas, pois acaba desconcentrando e com isto atrasando os desenvolvimentos.

Atendimento via MSN: Existem momentos onde vários usuários chamam via MSN e o atendente não consegue resolver todos ao mesmo tempo. Mesmo o atendente informando ao usuário que já esta ocupado, os mesmos não entendem e acabam reclamando do atendimento.

Conexão remota: O atendente reclama da necessidade de conexão remota em clientes para resolver problemas de software é muito freqüente, e que o mesmo perde muito tempo devido às conexões serem lentas.

Treinamentos: Referente ao atendimento externo existe o problema do treinamento que é realizado e algumas vezes o usuário acaba reclamando que não o recebeu. Nestes casos a empresa é obrigada a dar o treinamento novamente e sem custo para evitar atritos com o cliente.

4.2.1.2 Desenvolvimento de novos produtos

Atualmente a empresa trabalha com quatro desenvolvedores, o produto principal é um software de ERP (*Enterprise Resouce Planning,* em português*,* Planejamento de Recursos Empresariais) para indústria de confecção, o qual mantêm 82% de seus clientes. Alem disto, possui software de lavanderia industrial, varejo e construtora. Abaixo temos os problemas encontrados.

Equipe reduzida: A empresa possui somente quatro desenvolvedores que precisam manter quatro softwares sendo que o ERP para confecção é o maior de todos, atende indústrias de grande porte e exige constante manutenção.

Legislação: Atualmente a legislação esta sendo um grande empecilho na evolução dos produtos da Logosystem, pois a mesma esta modificando constantemente e principalmente na questão do varejo exige da empresa que o software seja praticamente reescrito.

Cronograma e prioridade: Os desenvolvedores reclamam que não existe um cronograma e uma prioridade para os desenvolvimentos. Tudo é desenvolvido conforme o cliente, se o mesmo é estratégico tudo que é seu desenvolvimento vira prioridade deixando os outros clientes de lado. E existem momentos que o desenvolvedor necessita parar o que esta desenvolvendo para atender uma questão do comercial, pois aquele cliente é muito importante naquele momento.

Muito trabalho, pouco retorno: A empresa trabalha com manutenção mensal em todos os clientes, e também trabalha com número X de horas que aquele cliente

pode utilizar do atendimento e desenvolvimento sem possuir nenhum custo adicional, mas a empresa esta perdendo dinheiro, pois não registra em local algum quanto tempo foi gasto com o cliente no mês e assim esta trabalhando muito alem do contratado e de graça.

5 UMA SOLUÇÃO A ESPREITA

Como descrito anteriormente a Logosystem possui vários problemas que dificultam a concretização de seus objetivos definidos até o final de 2010 e alguns deles são críticos para o crescimento da empresa. Com base nos problemas descritos foram elencadas algumas soluções juntamente com a diretoria da empresa para que seja possível o cumprimento de suas metas.

Na Logosystem existem quatro desenvolvedores, como descrito anteriormente, um deles é o que esta na empresa desde sua criação e tem grande conhecimento sobre os softwares que desenvolve, é tido dentro da empresa como o "herói". Foi definido que sua responsabilidade não é mais apagar incêndios e sim preveni-los com uma analise de tudo que será desenvolvido antes de ser repassado para os outros desenvolvedores e acompanhar o andamento dos projetos. Na etapa de analise foi definido que será desenvolvido o artefato de caso de uso e uma modelagem de processo para que o mesmo seja repassado aos desenvolvedores e suporte, gerando assim uma documentação que ajude no entendimento do que será desenvolvido e também no momento de ser realizado o atendimento. Com estas soluções a empresa estaria minimizando o problema descrito no item **Desenvolvedores dando suporte** e além disto gerando uma documentação simples e que não gera burocracia dentro da empresa.

No item **Atendimento via MSN**, realizamos uma análise e encontramos um software livre chamado Livezilla que permite o atendimento de clientes via chat conforme o MSN, mas possui alguns diferenciais que são: lista de espera, armazenamento da conversa em banco de dados, não é necessário instalar no cliente pois o mesmo realiza o acesso via site, ao final de qualquer conversa é enviado um e-mail para o cliente com todo o assunto tratando e ele pode ser customizado conforme as cores e logo da empresa. No APÊNDICE A mostra o acesso realizado no site da Logosystem.

Referente a tratamento das **Conexões remotas**, vimos que estão ocorrendo com muita freqüência devido a bugs (erros) deixados no software de vendas externo, este é utilizado por representantes das empresas que utilizam o ERP de confecção. Foi definido que se estiverem ocorrendo problemas com

freqüência e que estes problemas exigem conexão remota os mesmos devem ser tidos como prioridades de correção, pois este item acaba consumindo muito tempo do atendente e também acaba gerando descontentamento da parte do cliente.

Para o atendimento externo foi criado um documento padrão, no qual será descrito todo serviço realizado e ainda será assinado pelo cliente, desta forma o problema do **Treinamento** que é realizado e após um tempo o cliente diz que não o recebeu estaria resolvido. No APÊNDICE B é possível visualizar este documento.

A Logosystem possui quatro produtos conforme já descrito, visto que seu principal produto é o ERP de confecção, após uma reunião com os gestores foi definido que: a) o software de construtora não será comercializado visto que atualmente a empresa não possui conhecimento e pessoas disponíveis para sua manutenção, b) o software de varejo é uma necessidade, pois muitas fabricas necessitam da integração com suas lojas, visto que este software exige muitas modificações legais devido a mudanças na legislação foi decidido abandonar o projeto atual e criar uma parceria com uma empresa que trabalha especificamente com varejo e que seja possível uma integração com o ERP da fabrica, c) o software de lavanderia industrial será mantido, pois este é um setor importante dentro da industria de confecção e além disto complementa o ERP para industrias que possui lavanderia interna. Então de uma forma resumida ficou definido que o foco é a industria de confecção ficando somente o sistema de varejo em responsabilidade de uma empresa parceira. Com estas medidas a empresa estaria resolvendo o problema da **Equipe reduzida** com a redução na gama de produtos e da **Legislação** repassando a responsabilidade do software de varejo para um parceiro.

Para a resolução dos problemas referente a **Cronograma e prioridade** foi definido a implantação de uma metodologia ágil para ajudar na definição dos prazos, tonar os produtos flexíveis, permitir a geração de documentação, criar um método padrão de codificação e tornar a equipe independente para tomada de decisões. Com a implantação de uma metodologia de trabalho será possível mensurar quanto é trabalhado para cada cliente, desta forma permitindo um controle sobre as horas trabalhadas e assim a empresa poderá gerar uma cobrança para o cliente caso seja necessário, desta forma podemos resolver os problemas referente a **Muito trabalho, pouco retorno**.

5.1 A ESCOLHA DO MÉTODO

A escolha de um método de trabalho não é uma tarefa fácil, a empresa precisa analisar seus pontos fortes e pontos fracos para assim confrontar com as metodologias existentes e escolher as melhores praticas que se encaixam. Existem empresas que acabam escolhendo uma metodologia, pelo fato dela estar na “moda” e copiam todas as suas praticas, nestes casos com o tempo a metodologia acaba deixando a empresa engessada e com grande burocracia.

Durante certo tempo foi analisado o dia-dia de trabalho na Logosystem e visto alguns pontos interessantes, como citados abaixo.

- O tempo de liberação de modificações no software costuma acontecer de 7 a no máximo 15 dias;
- Quando é solicitado algum desenvolvimento novo que leve mais que 15 dias para ser liberado na completude, a empresa acaba realizando uma analise e assim liberando aos poucos as funções do software para o cliente até atingir a totalidade das modificações, com isto o cliente já consegue ir realizando algumas funções já no inicio dos desenvolvimentos;
- O tempo de retorno para correção de problemas é 24 horas.
- A equipe é pequena, mas muito comunicativa, geralmente toda a equipe sabe o que cada desenvolvedor esta desenvolvendo.

Com base nos pontos citados acima a empresa observou que o Scrum seria o melhor método para utilizar.

A implantação do Scrum iniciou com a inclusão de dois quadros de Kanban (Esta é uma palavra japonesa que significa placa de registros, o Kanban é muito utilizado pela indústria na gestão de produção, controlando os fluxos gerados na industrialização de produtos ERDMANN (1998)): um quadro para desenvolvimento e outro para testes. Nestes foram criados quatro colunas: backlog (Indica o projeto), para fazer, em andamento e feito. Dentro de cada coluna são adicionados *post it* (Papel adesivo para anotações), nestas anotações vão

informações macros sobre o que deve ser desenvolvido ou testado, geralmente a informação no *post it* é o nome de nova função, interface, etc. Após a implantação do Kanban foi colocado um gráfico de Burndown, ele possui dois eixos (X, Y), o X representa os dias de liberação da versão, que foram definidos 5 dias uteis e o Y o trabalho restante. No caso da Logosystem foi visto que a equipe consegue um total de cem em horas de trabalho nos cinco dias uteis, com base nisto o quadro de Kanban é montado toda a semana com base nas cem horas disponiveis. Nas figuras 13 e 14 podemos ver o Kanban e o gráfico de Burndown. Com estas duas ferramentas os desenvolvedores e gestores da empresa possuem uma visão melhor do andamento dos projetos, assim ajudando na tomada de decisão.

Figura 13 – Quadro de Kanban.
Fonte: do Autor.

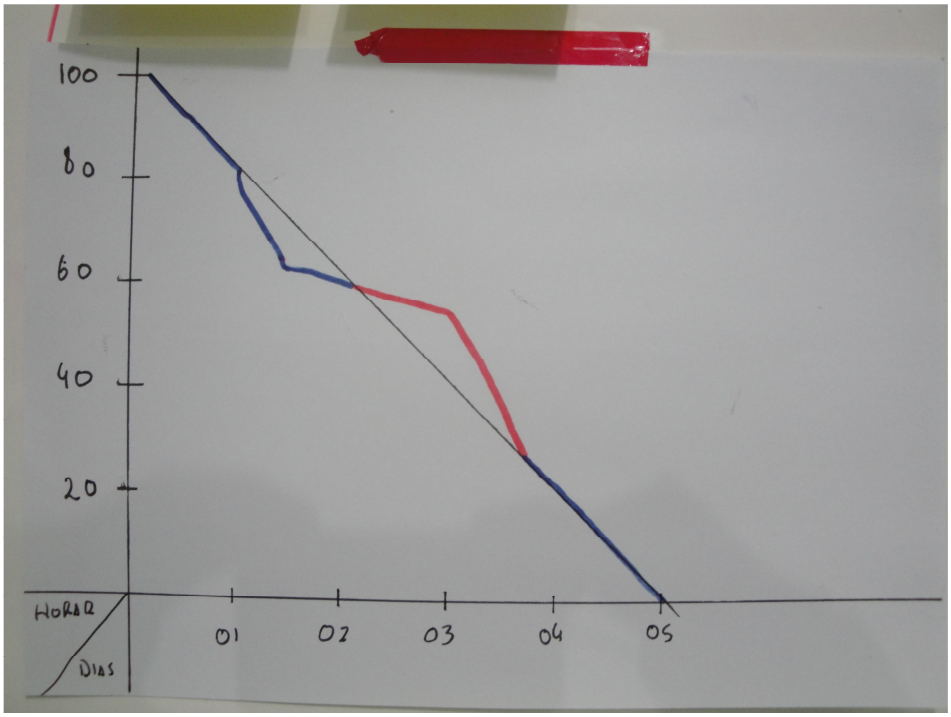


Figura 14 – Gráfico de Burndown.
Fonte: do Autor.

Tanto o quadro como o gráfico foi definido que é responsabilidade de todos da equipe atualizar, gerando na equipe uma maior organização e uma condição de cobrança entre os membros caso eles vejam que algum projeto esta atrasado. Alem da implantação destas duas ferramentas foi definido que todos os dias as 09:00 horas da manhã é a reunião para verificar o andamento dos projetos, esta reunião é realizada de forma informal em pé na sala do café, nela são questionados como foi o andamento do projeto no dia anterior, o que será feito hoje e se existem algum problema que possa atrasar o andamento de algum projeto.

Alem da implantação do SCRUM, foi também definida uma série de documentos com as regras para o desenvolvido, como modelo de codificação, reutilização de código fonte, modelos de telas de cadastro, consulta, relatório e processo. Com a criação destes documentos os desenvolvedores passam a ter um guia para lhes ajudarem no desenvolvimento dos projetos.

6 CONCLUSÃO

Com este trabalho podemos concluir que cada vez mais as empresas estão em busca de soluções que ajudem na sua organização, visando assim um maior controle de seus processos e uma agilidade maior na conclusão dos projetos. Este é um mercado de grande expansão para os profissionais de engenharia de software, uma vez que as empresas de software estão em grande expansão.

Com a implantação de algumas normas e métodos como descrito anteriormente, foi possível observar um grande ganho na organização da Logosystem Sistemas, antes deste trabalho ser iniciado a empresa estava necessitando contratar mais pessoas, pois possuía uma carga excessiva de trabalho, hoje após a implantação deste projeto a empresa conseguiu dobrar seu faturamento e não necessitou aumentar sua equipe, comprovando que o caos estava lhe gerando prejuízo e causando sua estagnação. Atualmente toda equipe esta satisfeita, pois agora conseguem enxergar para onde a empresa esta caminhando.

Também foi possível observar que existem outros documentos ainda a ser implantados para dar uma visão mais completa de todos os projetos, mas como já descrito neste trabalho, não podemos implantar tudo de uma única vez e sim, aos poucos ir criando o ambiente para inclusão de novos métodos.

A Logosystem em setembro de 2010 conseguiu dobrar seu faturamento em mensalidade e a conversão do ERP para versão 2 provavelmente não será terminada neste ano, mas com base no andamento do projeto muito provável em fevereiro de 2011 já estará finalizada, assim a empresa já conseguiu atingir uma de suas metas antes do prazo e já consegue vislumbrar uma data para a finalização da sua segunda meta.

REFERÊNCIAS

AGILE ALIANCE. **Manifesto for Agile Software Development**. Disponível em: <http://agilemanifesto.org>. Acesso em: 09 jul. 2010.

BEEDLE, M., et al., **SCRUM: An Extencion Pattern Language For Hyperproductive Software Development**, incluido em Pattern Language of Program Design 4, Addison Wesley Longman, Reading, MA, 1999. Pode ser baixado em http://jeffsutherland.com/scrum/scrum_plop.pdf.

ERDMANN, Rolf Hermann. **Organização de Sistemas de Produção**. Florianópolis: Insular, 1998.

FURTADO, André. **Pontas do Iceberg do Caos no Desenvolvimento de Software**. Disponível em: <http://www.microsoft.com/brasil/msdn/Tecnologias/Carreira/DesenvolvimentoSoftware.mspx>. Acesso em: 01 ago. 2010.

IEL/SC. **PLATIC – Arranjo Produtivo Catarinense:** Tecnologia da Informação e Comunicação. Florianópolis: IEL/SC, 2007.

PETERS, James F. **ENGENHARIA DE SOFTWARE:** teórica e prática. Rio de Janeiro: Campus, 2001.

PETERS, James F. **SOFTWARE ENGINEERING:** an engineering approach. New York: John Wiley & Songs Inc, 1999.

PFLEEGER, S. L. **ENGENHARIA DE SOFTWARE:** Teórica e Prática. São Paulo: Prentice Hall, 2ª edição, 2004.

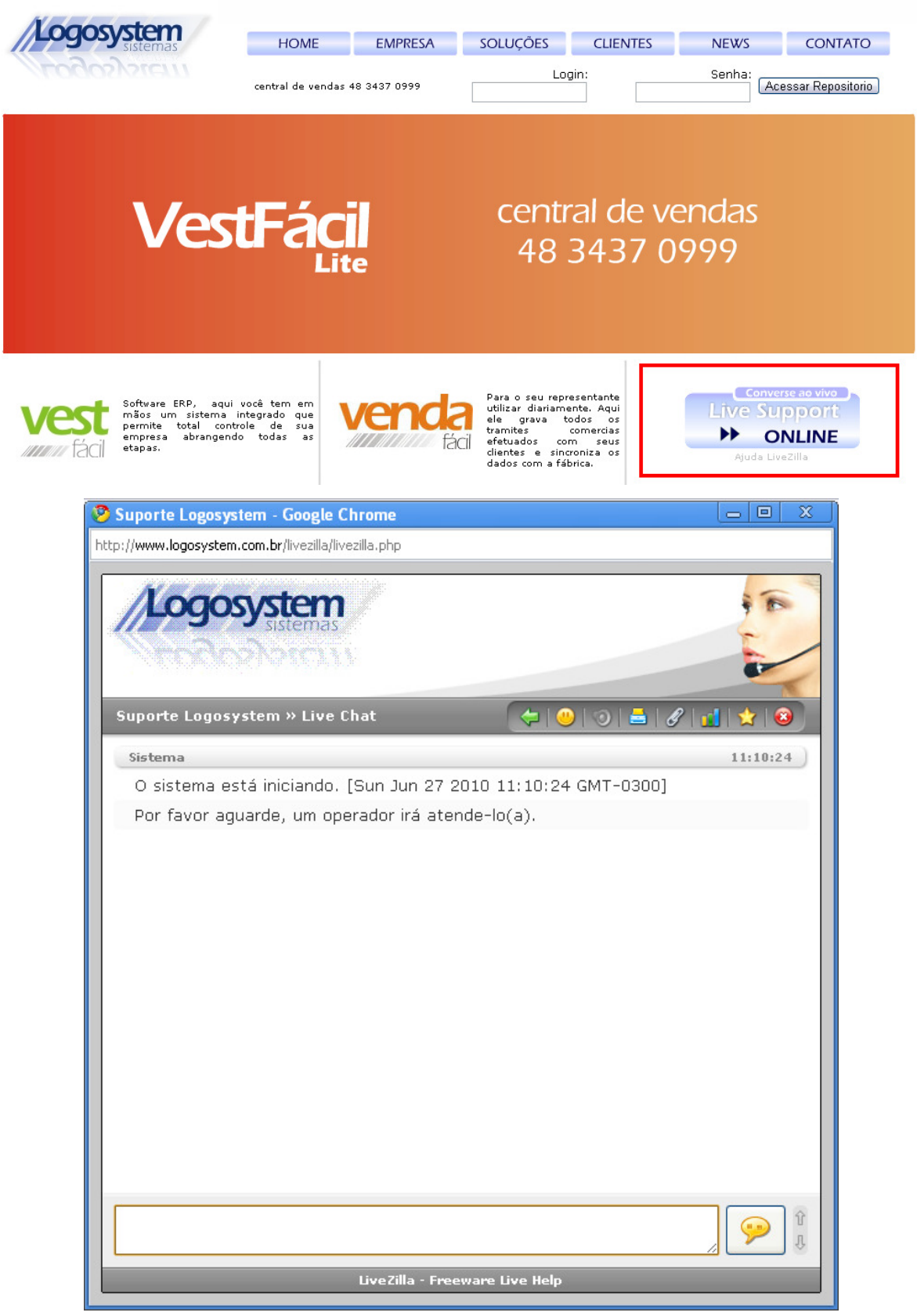
PRESSMAN, Roger S. **ENGENHARIA DE SOFTWARE**. São Paulo: McGraw-Hill, 2006.

PRESSMAN, Roger S. **ENGENHARIA DE SOFTWARE**. São Paulo: Pearson Makron Books, 1995.

SOMMERVILLE, Ian. **ENGENHARIA DE SOFTWARE**. São Paulo: Addison Wesley, 6ª edição, 2003.

APÊNDICE(S)

APÊNDICE A – Ilustração do sistema de comunicação virtual Livezilla



APÊNDICE B - Documento de atendimento externo

(48) 3437-0999 | suporte@logosystem.com.br
Criciúma | SC | logosystem.com.br

02/10/2010 09:30:24

Ordem de Serviço 01192

Cliente Glassi Fábrica
Solicitante ALCIMAR
Motivo Visita para andamento dos estoques da confecção.

Inicio _____
Final _____
KM I _____
KM F _____

Descriminação dos serviços efetuados

Espaço reservado

Quem assinou

Glassi Fábrica