UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEANDRO DAMINELLI

ABORDAGEM DE UMA FERRAMENTA DE APOIO À UTILIZAÇÃO DA TEORIA DA CORRENTE CRÍTICA PARA O GERENCIAMENTO DE PROJETOS NO CONTEXTO DA MELHORIA DO PROCESSO DE SOFTWARE

CRICIÚMA
2012

LEANDRO DAMINELLI

ABORDAGEM DE UMA FERRAMENTA DE APOIO À UTILIZAÇÃO DA TEORIA DA CORRENTE CRÍTICA PARA O GERENCIAMENTO DE PROJETOS NO CONTEXTO DA MELHORIA DO PROCESSO DE SOFTWARE

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: M.Sc Gustavo Bisognin

CRICIÚMA

2012

LEANDRO DAMINELLI

ABORDAGEM DE UMA FERRAMENTA DE APOIO À UTILIZAÇÃO DA TEORIA DA CORRENTE CRÍTICA PARA O GERENCIAMENTO DE PROJETOS NO CONTEXTO DA MELHORIA DO PROCESSO DE SOFTWARE

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Engenharia de Software

Criciúma, 25 de Junho de 2012.

BANCA EXAMINADORA

Prof. MSc. Gustavo Bisognin - UNESC - Orientador

Prof. MSc. Ana Claudia Garcia Barbosa – UNESC

Prof. MSc. Paracelso de Oliveira Caldas - UNESC

RESUMO

Este trabalho aborda o desenvolvimento de uma aplicação direcionada ao Gerenciamento de Projetos, com o objetivo de apresentar e utilizar os conceitos da Teoria da Corrente Critica em um software. Também é apresentado neste trabalho alguns dos principais modelos de Gerenciamento de Projeto, bem como toda a metodologia seguida para realização da pesquisa e desenvolvimento do software.

Palavras-chave: Gerenciamento de Projetos. Teoria da Corrente Crítica. Teoria das Restrições.

ABSTRACT

This paper talks about a Project Management application development. The main objective here is to show and apply theories and concepts around Critical Chain Theory when it comes to software development. This paper also shows some of the most important Project Management templates as well as all the steps followed in order to finish this research and to develop the proposed software successfully.

Keywords: Project Management. Critical Chain. Theory of Constraints.

LISTA DE ILUSTRAÇÕES

Figura 1 - Gráfico de Sprint Burndown.....26

Figura 2 - Quadro Kanban.....33

Figura 3 - Gráfico RUP.....39

Figura 4 - Processos de gerenciamento do PMBOK.....40

Figura 5 - Gráfico de acompanhamento da Corrente Crítica.....44

Figura 6 - Fluxo de atividades do desenvolvimento.....49

Figura 7 - Modelagem dos requisitos.....51

Figura 8 - Modelagem do banco de dados.....52

Figura 9 - Cadastro de atividades parte 1.....53

Figura 10 - Cadastro de atividades parte 2.....54

Figura 11 - Menu de processamento da Corrente Crítica.....55

Figura 12 - Gráfico de consumo do pulmão do projeto.....56

LISTA DE ABREVIATURAS E SIGLAS

PMI	Project Management Institute
PMP	Project Management
RUP	Rational Unified Process
TOC	Teoria das Restrições
UML	Unified Modeling Language
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVO GERAL 11
1.2 OBJETIVOS ESPECÍFICOS 11
1.3 JUSTIFICATIVA 12
1.4 ESTRUTURA DO TRABALHO 13
2 GERENCIAMENTO DE PROJETO 14
2.1 CONCEITO 14
2.2 PROJETO ORIENTADO A OBJETOS 18
2.3 PROJETO DE SOFTWARE DE TEMPO REAL 20
2.4 PROJETO DE INTERFACE COM O USUÁRIO 21
3 MODELOS DE GERÊNCIAMENTO DE PROJETOS 23
3.1 MODELOS ÁGEIS 23
3.1.1 Modelo Scrum 23
3.1.2 Modelo XP 27
3.1.3 Kanban 32
3.2 MODELOS TRADICIONAIS 36
3.2.1 RUP 36
3.2.2 PMBOK 40
4 TEORIA DAS RESTRIÇÕES 42
5 TEORIA DA CORRENTE CRITICA 44
5.1 SINDROME DO ESTUDANTE 45
5.2 LEI DE PARKINSON 46
6 TRABALHOS RELACIONADOS 47
7 IMPLEMENTAÇÃO DO SOFTWARE 48
7.1 METODOLOGIA 48
7.2 DESENVOLVIMENTO DA APLICAÇÃO 49
7.2.1 Levantamento de Requisitos 49
7.2.2 Modelagem dos Requisitos 50
7.2.3 Definição da Estrutura da Aplicação 51
7.2.4 Desenvolvimento da Interface 53

7.2.5 Implementação 56
7.2.6 Resultados Obtidos 57
8 CONCLUSÃO 59
REFERÊNCIAS 60
BILIOGRAFIA COMPLEMENTAR 61
APÊNDICE(S) 62

1 INTRODUÇÃO

Com a constante busca por maior produtividade e menores custos e prazos, a Teoria da Corrente Crítica, escrita por Eliyahu M. Goldratt, foi criada em 1997, trazendo novos conceitos para o gerenciamento de projetos e suas respectivas atividades.

A Teoria da Corrente Crítica traz uma forma diferente de tratar o gerenciamento de projetos. Baseada na Teoria das Restrições, a Corrente Crítica foca nas atividades do projeto e visa aplicar uma metodologia agressiva nas estimativas de tempo, diminuindo drasticamente o tempo estimado para realização de uma atividade do projeto.

A Teoria das Restrições (TOC), também criada por Eliyahu M. Goldratt, visa analisar e tratar as restrições de um projeto, seja esta uma restrição interna, externa ou de mercado, objetivando o aumento do desempenho das atividades do projeto. A TOC pode ser aplicada em diversas áreas de negócio, incluindo a Engenharia de Software.

Dentro dos conceitos abordados pela corrente crítica, estão a Lei de Parkinson e a Síndrome do Estudante, utilizadas como base para justificar os atrasos em projetos, que ocorrem mesmo com as grandes margens de segurança aplicadas sobre as atividades deste projeto.

A Lei de Parkinson, escrita por Cyril Northcote Parkinson no livro The Economist, parte do princípio de que quanto mais tempo livre uma pessoa tem, mais tempo ela vai levar para realizar suas atividades. Contextualizando, se um empregado tem um prazo para entregar o seu trabalho, mesmo que ele tenha terminado este trabalho antes do prazo final, o tempo restante é utilizado para "finalizar completamente" a atividade.

A Síndrome do Estudante é conhecida por todos, porém nem todos sabem que esta síndrome existe. Quando temos um prazo estipulado para desenvolver uma atividade, algumas pessoas acabam deixando para realizar o trabalho na "última hora", estando passíveis de situações inesperadas que resultam em atrasos no projeto.

Nas metodologias comuns de gerenciamento de projetos, são adicionados "pulmões" de segurança na estimativa de tempo de cada atividade do

projeto. Assim, o prazo final de conclusão de um projeto chega a ser até 40% maior que o prazo inicial. Ainda com a margem de segurança aplicada, a maioria dos projetos costuma atrasar graças ao comportamento do ser humano em relação à Lei de Parkinson e a Síndrome do Estudante.

Os constantes atrasos em projetos têm como conseqüência a menor competitividade no mercado, atrito com os clientes, maiores custos, entre outros, e acabam sendo um grande problema para uma empresa que possui muitos projetos e utilizam técnicas antigas de gerenciamento de atividades do projeto.

As formas de gerenciamento e controle tradicionais, abordados pela maioria das empresas desenvolvedoras de soluções de software, não conseguem obter um gerenciamento preciso e efetivo quanto a prazos, configurando um grave problema de custo e desgaste contínuo com o cliente.

Estudos recentes divulgados pelo Standish Group comprovam que mais de 80% dos projetos de software atrasam, causando um prejuízo considerável para as organizações. De acordo com este estudo, a forma de gerenciamento abordada na maioria das vezes é a tradicional, com gerenciamento de atividades e acompanhamento de cronograma, não havendo nenhuma técnica mais precisa para prever os desvios e controlar os riscos envolvidos nesta atividade.

Com base no problema supracitado, este trabalho tem como objetivo desenvolver uma aplicação que implementa a Teoria da Corrente Crítica aplicada no gerenciamento de projetos, abordando o contexto da melhoria contínua do processo de desenvolvimento de software.

1.1 OBJETIVO GERAL

Criar uma aplicação que implemente os métodos da Teoria da Corrente Crítica aplicada no gerenciamento de projetos, abordando o contexto da melhoria contínua do processo de desenvolvimento de software.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste projeto são:

a) estudar e aplicar os métodos da Corrente Crítica em Gerenciamento de Projetos;

b) compreender a importância do Gerenciamento de Projetos para obtenção de resultados expressivos para empresa;

c) compreender as principais causas de atrasos nas atividades de um projeto em relação ao seu cronograma inicial;

d) desenvolver uma ferramenta para gerenciar o cronograma de projetos aplicando os métodos da Corrente Crítica;

e) compreender os métodos utilizados para estimativa de tempo das atividades;

f) compreender o processo de desenvolvimento de software no contexto da melhoria contínua.

1.3 JUSTIFICATIVA

De acordo com o estudo da área de processo de gerenciamento de projetos podemos estabelecer que ela é fundada em três pilares: Tempo, Custo e Prazo, sendo que qualquer alteração em um destes pilares haverá impacto no projeto. A arte de prever e gerenciar estes impactos torna esta área de processo, fundamental para qualquer organização.

As organizações desenvolvedoras de produtos de software trabalham com um agravante ainda maior, as variáveis de ambiente, que pelo seu grau de subjetividade e complexidade, acabam variando muito e com isso causando impactos significativos nos pilares do gerenciamento. Estes impactos geralmente são considerados no planejamento inicial do projeto, porém são extremamente difíceis de controlar durante a execução das atividades planejadas, principalmente em projetos de longo prazo, por isso, é comum a utilização de margens de segurança durante o planejamento dos projetos.

Quando um gerente de projetos aplica uma margem de segurança muito grande nas atividades deste projeto, as pessoas envolvidas tem um certo conforto

em relação aos prazos, porém isto pode ser muito ruim para a empresa caso esta pessoa aproveite o tempo disponível para estender a sua atividade.

Ainda com as grandes margens de segurança aplicadas, vários projetos continuam sofrendo atrasos constantes. Estes atrasos podem ser explicados por muitas variáveis, dentre estas, o comportamento humano em relação a Lei de Parkinson e a Síndrome do Estudante.

Visando a diminuição efetiva do tempo despendido para realização de uma atividade e a constante melhoria do processo de software, a Teoria da Corrente Crítica será aplicada através de um software que gerenciará o cronograma de um projeto.

1.4 ESTRUTURA DO TRABALHO

Inicialmente é feita uma abordagem sobre gerenciamento de projetos, onde este é conceitualizado. Algumas formas de gerenciamento específico de projeto são abordadas.

Em seqüência, os principais modelos de gerenciamento são abordados para melhor entendimento da matéria. Entre os modelos de gerenciamento, estão os modelos ágeis (Scrum, XP, Kanban) e os modelos tradicionais (RUP e PMBOK).

Após explanação sobre os modelos de gerenciamento, a Teoria das Restrições é abordada e contextualizada para o gerenciamento de projetos. A Teoria da Corrente Crítica vem logo em seguida.

O detalhamento da implementação do software é abordado na sequência. O processo de desenvolvimento é explicado, bem como as principais funcionalidades e telas do programa.

2 GERENCIAMENTO DE PROJETO

A Gerência de Projetos é utilizada em larga escala por empresas do mais variado tipo de negócio. Seus benefícios foram sendo comprovados ao longo do tempo, trazendo como principal resultado a redução de custos, que aumenta consideravelmente de acordo com o tamanho dos projetos.

Segundo Molinari (2004), a aplicação da disciplina de Gerenciamento de Projetos atualmente inclui diversas áreas, como: indústrias, construção civil, sistemas de informação, serviços financeiros, educação, treinamentos entre outros. Acredita-se que o domínio e aplicação das boas práticas estabelecidas nesta disciplina, minimizam consideravelmente a perda de produtividade e o custo envolvido no retrabalho das atividades das mais diversas áreas do domínio do conhecimento. Contudo, a aplicação efetiva dos conceitos estabelecidos no gerenciamento de projetos, não é trivial de ser executado, envolvendo um alto investimento e um grande *know how* dos recursos humanos executores. Uma visão mais aprofundada desta disciplina pode ser observada no subitem 2.1.

2.1 CONCEITO

Segundo Sommerville (2007), gerência de projetos é a união, organização e aplicação de técnicas e conhecimento para realização de atividades que visam o alcance de um objetivo, controlando as variáveis de risco, prazo, custos e pessoas envolvidas. Todas estas variáveis são de extrema importância para o sucesso pleno do projeto, sendo assim, os gerentes de projeto envolvidos neste contexto devem possuir vasta experiência na área a fim de multiplicar as chances de sucesso do projeto.

Segundo o PMBOK (2008), o ato de gerir um projeto consiste em definir uma série de tópicos ou áreas de conhecimento acerca do objetivo que se deseja alcançar ao final deste. Estes tópicos são: Escopo, Tempo, Custos, Riscos, Qualidade, Recursos humanos, Comunicações, Aquisições e Integração.

a) escopo: O escopo descreve o projeto em si, de forma a definir seus requisitos, objetivos e meta a ser alcançada, e deve ser controlado

durante todo o ciclo de vida do projeto. Os requisitos são as necessidades que o cliente possui, como por exemplo, a necessidade de desenvolvimento de uma nova rotina para o software ou um novo relatório. O cliente não é necessariamente externo, podendo este ser também um setor da própria organização. Os objetivos são os resultados gerados ao fim do projeto, que irão suprir as necessidades do cliente, sendo que este resultado deve ser entregue dentro da metas estipuladas para o projeto.

b) tempo: O projeto é dividido em pequenas partes que podemos chamar de atividades, e através de técnicas de estimativa de tempo, é realizada a estimativa para cada atividade a fim de definir o cronograma do projeto. As atividades do projeto são definidas pelos gestores do projeto em conjunto com a equipe envolvida, e cabe ao gerente de projeto definir o quão detalhado será o desdobramento das atividades, por exemplo: em um tema comum do nosso cotidiano que é lavar roupas, o gerente de projetos poderia dividir as atividades de forma mais simplificada, como Lavar, Secar, Passar e Guardar, ou então detalhar estas atividades a fim de se alcançar um maior controle em cada passo, como Colocar a roupa na máquina, Lavar, Estender a roupa, Esperar a roupa secar, Passar a roupa, Dobrar a roupa e Guardar a roupa.
c) custos: O custo do projeto deve ser entendido também como valor de investimento. O custo com Recursos Humanos, Aquisições e demais gastos necessários devem ser avaliados para análise do custo-benefício do projeto. Neste tópico, a precisão do gerente de projetos e também da equipe envolvida deve ser alta, pois o orçamento para o projeto é de suma importância para sua aprovação com relação ao seu custo-benefício. Um projeto que não teve o gerenciamento correto de custos pode ser abortado no meio do caminho por falta de recursos, o que é uma falha grave para qualquer organização, seja ela pequena ou grande. Deve ser considerada também a confiança da gerência de alto nível na equipe envolvida no projeto.
d) riscos: Na definição dos riscos devem ser abordados todos os eventos indesejados que possam ocorrer ao decorrer do projeto, desde

mudanças na estrutura da empresa até problemas com colaboradores, visando estar preparado para agir na ocorrência do caso. Este é outro ponto crítico em qualquer projeto, pois riscos que não foram cogitados no momento da elaboração deste podem vir a mudar o cenário e causar até mesmo o fim prematuro do projeto. Qualquer risco por menor que este seja, deve ser considerado e deve possuir um plano de ação definido para contorná-lo.

e) qualidade: Segundo Molinari (2004), a qualidade do projeto visa garantir que o resultado seja realmente alcançado com base no que foi solicitado, com eficiência e alta produtividade. Não basta apenas cumprir o cronograma do projeto, mas para isto deixar de lado outros tópicos também importantes no decorrer das atividades. Entregar o resultado na data prometida, porém entregando um resultado destorcido com relação ao que foi definido no escopo do projeto pode ser muito desgastante no relacionamento entre empresa x cliente, por isto o gerente de projetos deve sempre manter o equilíbrio entre todos os tópicos do projeto, mantendo assim um alto índice de qualidade geral, e não focando apenas em tempo ou custos.
f) recursos humanos: São as pessoas elencadas para desenvolver as atividades a fim de realizar o objetivo do projeto. A equipe definida para desenvolver o projeto é muito importante, pois neste ponto existe impacto direto em outros tópicos como Tempo, Qualidade e Custos. Um colaborador mais experiente pode realizar a mesma atividade com mais qualidade e em menos tempo, porém com um custo maior. Um colaborador que acabou de entrar na organização pode custar menos, porém demorar mais para realizar a atividade e não realizar da forma correta (menor qualidade). Neste ponto o gerente de projetos deve alcançar um ponto de equilíbrio. Uma das formas de alcançar este ponto é envolvendo pessoas experientes e não experientes no projeto, sendo estas últimas sempre supervisionadas, garantindo assim um equilíbrio entre custos, tempo e qualidade, e também contribuindo para o desenvolvimento de colaborados menos experientes.

g) comunicações: Indispensável para o sucesso do projeto, a comunicação entre os participantes deve ser clara e objetiva para que problemas como a má interpretação dos requisitos não ocorra. Reuniões diárias de acompanhamento e comunicação oral, evitando mensagens eletrônicas, pode ajudar muito na interação entre a equipe, o que favorece até mesmo a criar um laço de amizade entre os participantes, aumentando assim a idéia de time entre os colaboradores da empresa. Uma abordagem maior sobre técnicas de comunicação será descrita no capitulo 3.

h) aquisições: Aquisição de produtos e serviços necessários para a execução do projeto. Este tópico também impacta diretamente no Custo do projeto. A necessidade de aquisição de materiais fora do planejado pode trazer dor de cabeça aos gestores, e a definição correta deste tópico é muito importante para avaliação do custo-benefício do projeto. Dependendo do escopo do projeto, este tópico pode definir a viabilidade de execução do projeto.

i) integração: União e interação de todas as áreas de conhecimento realizada no dia-a-dia pelo gerente de projetos, a fim de manter o equilíbrio na definição e acompanhamento de todos os tópicos do projeto.

Os tópicos supracitados são relacionados de tal forma que, caso ocorra alguma alteração em um item no decorrer do projeto, pelo menos algum outro tópico também sofrerá alterações. Como exemplo, em um projeto já definido, caso ocorram alterações no escopo do projeto, o cronograma terá que ser alterado a fim de se adequar ao novo escopo, bem como o custo será alterado por conta do novo cronograma e assim por diante.

Através dos projetos, as empresas buscam a organização, redução de custos, controle de riscos e redução de prazos, de forma a garantir cada vez mais a qualidade do seu negócio, bem como sua rentabilidade.

Os projetos são concebidos geralmente a partir do planejamento estratégico de uma organização, com o objetivo de atender uma solicitação dos clientes, sejam estes internos ou externos à organização, atender a demanda de mercado, troca de tecnologia utilizada, mudança de local físico da organização ou de determinado setor, treinamento da equipe envolvida, entre outros.

O gerenciamento de projetos visa, através de várias técnicas e conhecimento acumulado, organizar a execução de um projeto do início ao fim, prevendo custos, riscos, alocando recursos, garantindo a qualidade e entregando o resultado no prazo definido de acordo com o escopo do projeto.

O inicio do gerenciamento de projetos se deu antes mesmo do nascimento de Cristo. Grandes construções como as pirâmides do Egito exigiram alguma forma de organização para que pudessem ser concluídas.

Apesar de alguns conceitos já serem aplicados há muito tempo atrás, foi na década de 50 que se iniciou a nova era do gerenciamento de projetos. Nesta época, o gráfico de Gantt, criado por Henry Gantt, era amplamente utilizado e sobrevive até hoje nas modernas ferramentas de controle de projetos.

Em 1969, o Project Management Institute (PMI) foi fundado com o objetivo de definir padrões de gerenciamento de projetos e hoje conta com mais de 150 mil membros em todo mundo. Em 1996, lançaram o guia PMBOK, que formaliza diversos conceitos no gerenciamento de projetos e um conjunto de boas práticas, aplicáveis à maioria dos projetos na maior parte do tempo.

O PMI também certifica profissionais da área como Gerente de Projetos Profissional através da certificação Project Management Professional (PMP). No Brasil, as estatísticas revelam que há apenas aproximadamente 400 profissionais certificados nesta área.

Pesquisas realizadas pela consultoria *The Standish Group* em 1994 e 2001 comprovam a rápida evolução do gerenciamento de projetos. A pesquisa realizada em 2001 aponta uma redução de 144% na extrapolação de orçamento, redução de 159% na extrapolação do prazo e aumento de 12% na entrega de resultados dentro do tempo, custo e especificação prevista. Estes dados comprovam a eficácia e os benefícios do gerenciamento de projetos dentro de uma organização.

Nos capítulos subseqüentes serão apresentadas algumas formas com que os projetos de softwares são abordados.

2.2 PROJETO ORIENTADO A OBJETOS

A Orientação a Objetos, conceito muito utilizado para o desenvolvimento de software, utiliza o conceito de classes de objetos como forma de separação de

código, possibilitando a reutilização do que já foi feito sempre que necessário, por exemplo: supondo que determinado sistema de vendas possui uma classe que determina o que está sendo vendido. Considerando como exemplo de objeto um Carro: a classe Carro possui os atributos que qualquer carro possui como Modelo, Marca, Ano de fabricação, Preço, entre outros, e também possui serviços disponíveis que são relativos ao objeto Carro como métodos que informam ao usuário o estado do carro de acordo com seus atributos. Na orientação a objetos, sempre que se fizer necessário envolver um Carro no negócio, não será necessário recodificar tudo sobre o carro, basta utilizar a classe já criada uma única vez. Este mesmo conceito é utilizado no gerenciamento de projetos.

Segundo Sommerville (2007), um objeto é uma entidade que possui um estado e um conjunto de atributos e operações para operá-los. Estas operações fornecem serviços a outros objetos sempre quando necessário.

Os sistemas desenvolvidos sob o conceito de orientação a objetos são mais fáceis de receber manutenção, pois como os objetos são independentes, eles podem também ser alterados de forma independente, onde a adição ou alteração de algum serviço não afeta outros objetos do sistema. Seguindo o exemplo supracitado do carro, caso adicionássemos um novo atributo, este não impactaria no restante de software, pois todo o código relacionado ao carro está contido em um único lugar.

Na gestão de projetos, o conceito de orientação a objetos é utilizado da mesma maneira, porém voltado para os processos do projeto, que segundo Sommerville (2007) possuem cinco estágios: Compreender e definir o contexto do sistema e modos de uso, projetar a arquitetura do sistema, identificar os principais objetos do sistema, desenvolver os modelos dos objetos e especificar as interfaces dos objetos.

A vantagem obtida com o uso desta metodologia é a mesma obtida no desenvolvimento de software: reutilização de objetos. Como os objetos são encapsulados com seus atributos e serviços independentes, estes podem ser reutilizados em vários projetos, reduzindo os custos do projeto, programação e validação, além da considerável diminuição do tempo de planejamento e execução das atividades.

2.3 PROJETO DE SOFTWARE DE TEMPO REAL

Segundo Sommerville (2007), um sistema de tempo real é um software que necessita de resultados produzidos pelo sistema e do respectivo tempo em que este resultado é produzido para ter seu funcionamento correto, ou seja, se o resultado produzido pelo sistema não for o esperado ou então for apresentado na hora errada, o software não funcionará da forma correta, por exemplo: um sistema que deve manter uma determinada temperatura de uma caldeira deve ler em tempo real a temperatura da caldeira, para processar esta informação e gerar uma saída, que seria o ajuste do objeto que aumenta ou diminui a temperatura da caldeira. Se a temperatura atual não fosse informada na hora correta, o software iria processar a informação e gerar uma resposta com uma ação a ser tomada na hora errada, podendo assim acontecer graves problemas por conta do erro no processamento.

O sistema de tempo real é baseado em estímulos e respostas: de acordo com um estímulo de entrada, o sistema realiza o processamento deste estímulo e gera uma resposta. Desta forma, é possível definir uma lista de estímulos que o sistema irá tratar, bem como a respectiva resposta para tal estímulo e o tempo em que a resposta deve ser produzida.

Para gerenciar um projeto de software de tempo real, existem vários estágios a serem considerados (SOMMERVILLE, 2007):

a) listar os estímulos que o sistema trata bem como suas respostas;
b) identificar para cada estímulo e resposta, as restrições de tempo;
c) escolher o hardware e o sistema operacional que irão rodar o sistema;
d) dividir o estímulo e a resposta em processos concorrentes (ao mesmo tempo);
e) projetar os algoritmos que irão realizar os processamentos necessários para cada estímulo e resposta;
f) projetar o cronograma do projeto.

Por existir a necessidade do sistema de tempo real atender as restrições de tempo, dependendo do sistema a ser desenvolvido pode não ser possível a utilização de orientação a objetos. Na orientação a objetos, os atributos de uma determinada classe ficam encapsulados e são acessados apenas por métodos da

própria classe. Em um sistema grande, isto pode significar perda de desempenho e conseqüentemente não atender as restrições de tempo do projeto. No exemplo da caldeira, não é suficiente a leitura da temperatura em tempo real. A resposta do software também deve ser rápida para que o sistema não gere um estímulo para uma condição irreal.

Todas estas necessidades devem ser cuidadosamente avaliadas no escopo do projeto para que este possa ser desenvolvido da maneira correta, atendendo a todas as necessidades do cliente.

2.4 PROJETO DE INTERFACE COM O USUÁRIO

Os softwares possuem a mais variada “gama” de usuários. De jovens a idosos, com diferentes experiências na área, cada um utiliza o software o seu jeito para chegar a um mesmo resultado, claro que sempre dentro do que a aplicação permite.

Projetar a interface de um software é algo muito complexo. Apresentar o software de forma com que este seja intuitivo é uma tarefa difícil, principalmente quando o nível dos usuários que o utilizarão é totalmente variável.

Segundo Sommerville (2007), um software bem projetado com relação a sua estrutura deve transpor isto através de sua interface. As interfaces com o usuário devem considerar tanto a sua experiência como seu ambiente de trabalho, a fim de minimizar os chamados “Erros de usuários”. Um sistema que não possui uma interface amigável limitará o usuário menos experiente ao invés de ajudá-lo a chegar no resultado desejado para qual o sistema foi desenvolvido.

No desenvolvimento de um projeto de software, é importante definir padrões de interfaces a fim de causar o menor impacto possível nas transições de tela ou mesmo de sistemas, por exemplo: determinado sistema da organização possui o menu de opções localizado à esquerda e ordenado em ordem alfabética. Caso a organização venha a desenvolver uma atualização para este sistema e altere o menu para a direita e também altere sua forma de ordenação, o impacto causado ao usuário será grande e o tempo de readaptação também. Este tipo de mudança deve ser evitado pela organização, pois força o usuário a reaprender uma interface

diferente, e este acaba trabalhando com diferentes interfaces no seu dia-a-dia dificultando assim a execução do seu trabalho de forma eficiente.

Portanto, ao definir a interface do software no projeto, a equipe deve estar ciente das necessidades do cliente, dos usuários que utilizarão o software e também dos padrões que já são utilizados para minimizar ao máximo o “efeito surpresa” na entrega do resultado.

No próximo capitulo serão abordados alguns dos principais modelos de gerenciamento de projetos utilizados em todo o mundo.

3 MODELOS DE GERÊNCIAMENTO DE PROJETOS

3.1 MODELOS ÁGEIS

O desenvolvimento ágil de software é, segundo Boehm (2006), uma tendência a partir do ano 2000, onde o ritmo de mudanças na área de tecnologia da informação é acelerado e a necessidade de produzir software em um curto período de tempo é grande.

Os modelos ágeis de desenvolvimento de software visam a entrega rápida do resultado, utilizando técnicas que mantém os padrões de qualidade do projeto, reduzindo custos e o cronograma, aumentando assim a satisfação do cliente e a eficiência da equipe envolvida.

Dentre as várias metodologias ágeis de gerenciamento de projetos, podemos destacar três: Modelo Scrum, Modelo Extreme Programming (XP) e Modelo KamBam, abordadas em seqüência.

3.1.1 Modelo Scrum

Criado em 1996 por Ken Schwaber e Jeff Sutherland, o Modelo Scrum defende que o desenvolvimento de software é imprevisível deve ser abstraído. Este modelo destaca-se dos demais métodos ágeis por dar mais atenção ao processo de gerenciamento de projetos. O desenvolvimento de software foi considerado como imprevisível para Ken Schwaber, pois não é possível definir com precisão a arquitetura do software bem como a estrutura de suas rotinas, por isto o desenvolvimento deve ser abstraído na concepção do projeto.

Segundo Schwaber (2004), o modelo Scrum possui atividades de monitoramento e *feedback*[1], através de reuniões rápidas e diárias com toda a equipe, visando a identificação e correção de qualquer problema no processo de desenvolvimento do software que venha a ocorrer.

Este modelo ágil assume a premissa de que o desenvolvimento de software não pode ser totalmente planejado no seu início por ser complexo e

[1] Palavra em inglês que significa “retorno da informação” ou simplesmente “retorno”.

imprevisível. O método baseia-se ainda em equipes pequenas, de até sete pessoas, requisitos não tão estáveis ou até desconhecidos e liberações curtas, sendo o desenvolvimento divido em intervalos de no máximo 30 dias, chamados de "*Sprints*". Requisitos não estáveis podem ser interpretados como requisitos pouco definidos que podem sofrer alterações no decorrer do projeto. As liberações curtas expressam a idéia de que poucas rotinas são desenvolvidas e liberadas em cada *sprint*.

O ciclo de vida deste modelo possui quatro fases (SCHWABER, 2004):

a) planejamento: nesta fase é estabelecida a visão do projeto e são garantidos os recursos para execução das tarefas, ou seja, é alocada a equipe que participará do projeto. Nesta fase também são criadas as versões iniciais do *Product Backlog*. O *Product Backlog* nada mais é do que os requisitos do projeto, ou seja, as funcionalidades que serão desenvolvidas de acordo com a necessidade do cliente.
b) stagging: nesta fase é avaliado o tamanho do projeto, criando itens adicionais ao *Product Backlog* relacionados com o tipo do sistema, ambiente de desenvolvimento, tipo de aplicação, entre outros. Ainda nesta fase, a equipe alocada para o projeto é dividida em times e são definidos os mecanismos de comunicação e coordenação entre eles.
c) desenvolvimento: nesta fase todas as funcionalidades especificadas no *Product Backlog* são dividas em s*prints* e são desenvolvidas pela equipe responsável.
d) releasing: nesta fase o produto é entregue ao cliente após o desenvolvimento de todas as *sprints*.

Segundo Schwaber (2004), o Modelo Scrum estabelece uma série de regras e práticas gerenciais que devem ser seguidas para o sucesso do projeto. Dentre as práticas gerenciais, podemos citar: *Product Backlog, Daily Scrum, Sprint, Sprint Planning Meeting, Sprint Backlog e Sprint Review Meeting.*

a) product backlog: são os requisitos do projeto, ou seja, as funcionalidades que serão desenvolvidas de acordo com a necessidade do cliente. A definição desta pratica é muito importante para que o produto seja entregue de acordo com as reais necessidades do cliente.
b) daily scrum: são as reuniões diárias realizadas entre a equipe envolvida no projeto. Nesta reunião a equipe discorre sobre todo o

trabalho que foi realizado no dia anterior para que se possa planejar o novo dia de trabalho. O objetivo da reunião não é um *feedback* da equipe ao seu gestor, e sim um *feedback* de cada integrante quando ao seu trabalho perante a equipe.

c) sprint: são os intervalos de desenvolvimento das funcionalidades do sistema. Por exemplo: no Sprint#1, com duração de 15 dias serão desenvolvidas as funcionalidades A, B e C. No Sprint#2, com duração de 15 dias serão desenvolvidas as funcionalidades X, Y e Z.
d) sprint planning meeting: reunião onde são elencadas as funcionalidades com maior prioridade de desenvolvimento, e após esta reunião o Sprint Backlog é gerado.
e) sprint backlog: é a lista de funcionalidades que será desenvolvida pela equipe no *sprint*.
f) sprint review meeting: ao final do *sprint*, a equipe realiza esta reunião com o objetivo de levantar o resultado que foi alcançado através da iteração para avaliação do resultado obtido x planejado.

Dentro do Scrum, existem três papéis fundamentais: o *Product Owner*, que representa o interesse de todos os envolvidos no projeto, o *ScrumMaster*, que gerencia todo o processo do Scrum, e a equipe, responsável por desenvolver as funcionalidades do produto.

De acordo com Schwaber (2004), um projeto que utiliza a metodologia Scrum inicia-se com uma visão do produto que será desenvolvido. Esta visão possui os requisitos do produto estabelecidos pelo fornecedor de requisitos (cliente), além de algumas premissas e restrições. Na seqüência, o *Product Backlog* é criado contendo a lista dos requisitos elicitados. O *Product Backlog* é então dividido em pacotes de liberação, cada um com sua prioridade definida.

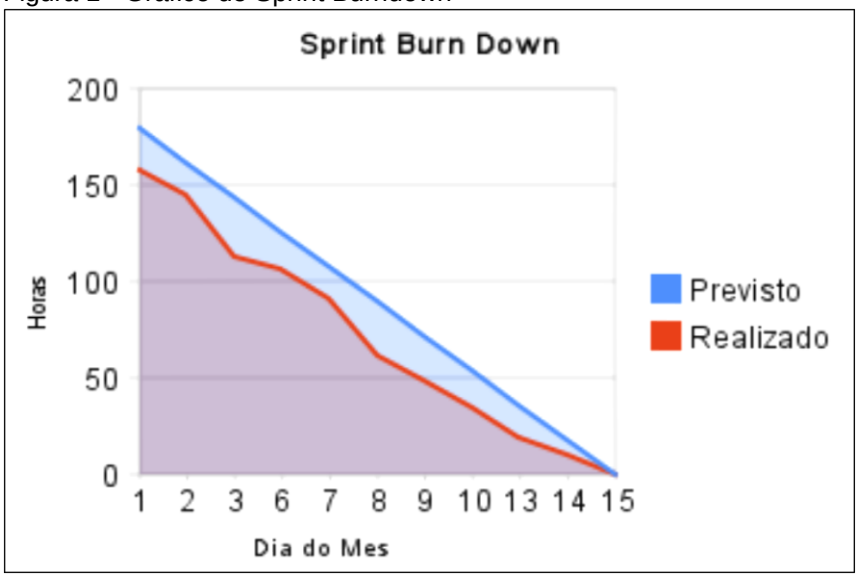
No *Scrum,* as atividades são dividas e realizadas em *Sprints*. De acordo com Schwaber (2004), cada *Sprint* inicia-se com uma reunião de planejamento (*Sprint Planning Meeting*), onde o *Product Owner* e sua equipe decidem juntos o que será implementado. A reunião de planejamento é dividida em duas partes, onde na primeira parte (*Sprint Planning* 1) o *Product Owner* apresenta os requisitos mais importantes e prioriza aqueles que devem ser implementados primeiro. A equipe então define o que poderá entrar no desenvolvimento do próximo *Sprint*, considerando sua capacidade de desenvolvimento. Na segunda parte da reunião de

planejamento (*Sprint Planning* 2), a equipe planeja seu trabalho, definindo o *Sprint Backlog*, que são as atividades necessárias para implementar as funcionalidades selecionadas no *Product Backlog*. A lista de atividades pode ser modificada pela equipe no decorrer do *Sprint,* e podem variar entre quatro e dezesseis horas para a sua conclusão.

Durante a execução dos *Sprints*, a equipe realiza uma reunião diária de 15 minutos para acompanhar o progresso do trabalho e agendar as próximas reuniões. Nesta reunião diária (*Daily Scrum Meeting*), cada membro da equipe responde três perguntas básicas: O que eu fiz no projeto desde a última reunião? O que irei fazer até a próxima reunião? Quais são os impedimentos? Ao final do *Sprint*, é realizada a reunião de revisão (*Sprint Review Meeting*) para que a equipe apresente o resultado alcançado no *Sprint* corrente ao *Product Owner*. Neste momento as funcionalidades são inspecionadas e adaptações do projeto podem ser realizadas. Após a inspeção das funcionalidades, o *ScrumMaster* conduz a reunião de retrospectiva, com o objetivo de melhorar o processo de desenvolvimento e/ou produto para o próximo *Sprint*.

No decorrer do projeto, o seu progresso é monitorado através de dois gráficos: *Product Burndown* e *Sprint Burndown*. Através destes gráficos é possível visualizar ao longo do tempo a quantidade de trabalho que ainda resta a ser feito, tanto para o produto em si como para um *Sprint* específico. Conforme figura 1, segue um exemplo de gráfico *burndown*:

Figura 1 - Gráfico de Sprint Burndown



Fonte: Alecrim (2009).

O gráfico exibido conforme Figura 1, faz um cruzamento de horas x dias, onde o número de horas máximo exibido no início do gráfico é a resultante da estimativa de tempo da equipe com relação ao desenvolvimento das funcionalidades contidas no *sprint* corrente, e o número de dias exibidos é o prazo ideal que a equipe tem para o término do *sprint*.

No exemplo citado, a equipe está realizando mais trabalho do que foi previsto, ou seja, está dentro da meta de quinze dias. Analisando o gráfico, é possível afirmar que próximo ao dia 8, o número de horas de trabalho previsto a ser concluído (linha azul) era aproximadamente 100 horas, enquanto a equipe já havia produzido o suficiente para restar apenas aproximadamente 60 horas de trabalho (linha vermelha) no mesmo dia.

O gráfico *Product Burndown* segue a mesma idéia do gráfico exemplificado, porém o *Product Burndown* exibe a estatística de desenvolvimento do produto inteiro, e não apenas de um *sprint* isolado.

3.1.2 Modelo XP

O modelo Extreme Programming ou XP, apresentado no final da década de 90, apresenta uma maneira diferenciada para se gerenciar um projeto. Muitas de suas regras causam certo impacto no primeiro momento e outras não possuem sentido se aplicadas de forma isolada. O que faz do modelo XP um sucesso é o seu conjunto, que foca no desenvolvimento rápido garantindo a satisfação dos clientes e cumprimento de metas e estimativas.

Segundo Beck (1999), as regras práticas e valores da Extreme Programming proporcionam um agradável ambiente de desenvolvimento de software para as equipes, que são conduzidas por quatro valores: comunicação, simplicidade, *feedback* e coragem.

a) comunicação: ponto chave deste modelo, a comunicação pessoal deve ser ativa entre seus usuários. As pessoas envolvidas devem estar próximas fisicamente uma das outras, de forma a facilitar a comunicação e acelerar o fluxo de informações. Os projetos que utilizam o XP também costumar ter poucos participantes, o que facilita a troca de informações entre a equipe, pois é mais fácil para um

integrante saber o que o seu companheiro está desenvolvendo no momento e não atrapalhá-lo com interrupções.

b) simplicidade: segundo estudos do Standish Group, mais de 50% das funcionalidades dos sistemas não são utilizadas pelos usuários, ou seja, mais da metade do software não teria necessidade de ser implementado. O modelo XP defende que o desenvolvimento do software deve ser o mais simples possível, com o mínimo de generalizações, garantindo o desenvolvimento apenas do que realmente é necessário. Esta não é uma tarefa fácil, pois na maioria das vezes nem mesmo o cliente sabe exatamente o que lhe é necessário, porém a equipe deve manter o foco no problema a ser resolvido para evitar ao máximo o desenvolvimento de funcionalidades dispensáveis.

c) feedback: o modelo XP utiliza ciclos de curtos de desenvolvimento e *feedback*, assegurando que pouco trabalho seja desenvolvido e entregue de cada vez. Desta forma, com rápidos *feedbacks* do usuário, a equipe de desenvolvimento segue o projeto apenas quando o trabalho está correto, e caso ocorram problemas na liberação de funcionalidades, estas são corrigidas antes mesmo do início do desenvolvimento de novas funcionalidades. A quantidade pequena de funcionalidades liberadas por iteração garante também a rapidez de correção de eventuais problemas devido ao pequeno escopo em que a equipe trabalha, agregando valor ao projeto por minimizar a ocorrência de erros e problemas na liberação final do projeto para o cliente.

d) coragem: o desenvolvimento de software é muito complexo. Muitas coisas podem sair erradas e isso exige coragem por parte da equipe. No modelo XP, se parte do princípio de que até os piores problemas ocorrerão. Assim, no início do projeto, uma “rede de proteção” é criada com o objetivo de tratar e reduzir ou até eliminar todos os problemas que podem vir a ocorrer no decorrer do projeto. Esta proteção contra os riscos que o projeto tem é muito importante para garantir tanto a rentabilidade quanto a qualidade do resultado final.

Segundo Beck (1999), a metodologia XP utiliza treze práticas que devem ser seguidas para o bom andamento e sucesso do projeto. São elas:

a) cliente presente: para que o software a ser entregue seja realmente o que o cliente deseja, deve existir uma forte relação entre o cliente e a equipe de desenvolvimento. Falta de comunicação e ausência do cliente é um dos fatores que mais geram falhas no decorrer do projeto. A presença física do cliente é muito importante ao longo do desenvolvimento do software, pois esta relação permite que possíveis erros no projeto sejam detectados e corrigidos rapidamente. O desenvolvimento de funcionalidades incorretas é comum em vários projetos de diferentes organizações, e a presença do cliente não apenas no momento da definição, mas também durante o desenvolvimento das funcionalidades, é muito importante para minimizar estes acontecimentos.
b) jogo do planejamento: no desenvolvimento de software, conforme já citado, muitas funcionalidades são desenvolvidas e nunca são utilizadas. O planejamento no modelo XP é utilizado para assegurar que a equipe trabalhe no que realmente for importante para o momento do projeto. O planejamento é realizado junto ao cliente, que escolhe as funcionalidades que serão desenvolvidas no próximo release, minimizando assim a incidência de funcionalidades não tão importantes para o projeto.
c) stand up meeting: diariamente a equipe envolvida no projeto realiza uma reunião com o objetivo de discutir tudo que foi realizado no dia anterior. Desta forma a equipe fica ciente dos resultados obtidos até então e organiza as atividades a serem realizadas no dia. Esta reunião também é muito importante para que a equipe possa relatar eventuais impedimentos no seu dia de trabalho e buscar soluções em conjunto. A reunião não deve tomar muito tempo, por isto, o líder da reunião deve ficar atento a desvios de foco e sempre que necessário realizar interrupções. Quando a discussão sobre problemas for algo mais complexo, esta deve se resolvida fora da reunião.
d) programação em par: sendo uma das técnicas de rede de proteção do modelo XP, a da programação em par é utilizada para rápida

identificação de problemas que possam levar muito tempo até serem encontrados. Muitos erros podem passar despercebidos pela pessoa que está escrevendo o código, porém, muitos erros podem ser detectados facilmente quando se trabalha em conjunto. Erros simples que envolvem até mesmo a sintaxe da linguagem podem levar muito tempo até serem descobertos. Algumas plataformas de desenvolvimento auxiliam de forma eficiente o desenvolvedor a localizar erros de sintaxes, outras não. É nesse ponto que a programação em par se destaca. A discussão da melhor forma para arquitetar um código também é muito importante pois ajuda no desenvolvimento técnico dos programadores e isso se reflete na qualidade do código produzido.

e) código coletivo: a idéia do código coletivo consiste em permitir que todos os desenvolvedores tenham acesso a todas as classes e métodos do software, de forma que estes possam realizar alterações a qualquer hora, com o objetivo de manter o código sempre limpo e simples. Este método também torna a equipe de desenvolvimento mais forte, pois todos os desenvolvedores acabam trabalhando em todas as partes do software. Este método acaba funcionando também como uma avaliação dos programadores mais experientes sob os programadores iniciantes, visto que fugas do padrão ou codificação incorreta podem ser observadas neste momento e o colaborador pode ser corrigido.

f) código padronizado: uma das maiores dificuldades das equipes de desenvolvimento é compreender o código alheio para dar manutenção. Com a padronização do código, esta tarefa se torna mais simples e rápida de ser feita, e junto com o código coletivo e programação em par, o código padronizado fica mais fácil de ser mantido.

g) design simples: por ser uma metodologia ágil, o modelo XP possui iterações rápidas e os processos de análise, design, desenvolvimento e testes devem ser realizados em um curto espaço de tempo, o que se torna uma tarefa difícil caso o design da aplicação for muito complexo. O modelo XP defende que funcionalidades flexíveis e

genéricas devem ser desenvolvidas apenas quando forem realmente necessárias, e seu design deve ser simples, intuitivo e que possa evoluir com o tempo. É muito importante que o usuário consiga utilizar o sistema de forma natural e para que isso ocorra não é necessário um design carregado e embelezado.

h) desenvolvimento orientado a testes: a correção de problemas após algum tempo da codificação é geralmente muito custosa. Quanto mais complexo for o método, mais difícil será para o desenvolvedor lembrar o “por que” daquele código. Desta forma, os casos de testes são escritos antes mesmo da codificação dos métodos, pois assim que o método for escrito, o desenvolvedor já estará apto a testar o seu código e saber se o que foi desenvolvido está de acordo com o objetivo da funcionalidade.

i) refatoração: o objetivo da refatoração é melhorar a estrutura interna do software sem modificar o seu comportamento para o usuário. A refatoração é importante também para manter o código sempre limpo, claro e de acordo com os padrões definidos pela equipe. Normalmente esta atividade não possui uma alocação de tempo prevista no projeto para ser realizada, e acaba sendo executada diariamente sempre que os desenvolvedores acharem necessário.

j) integração contínua: quando vários programadores trabalham no mesmo software, existe a necessidade de sincronizar o trabalho realizado, pois a equipe deve garantir que suas rotinas se integrem com as demais funcionalidades do software que sofrem mudanças constantes. No modelo XP, os programadores integram seu código com a versão mais recente do software de forma a garantir que sua rotina está funcionando junto das demais funcionalidades do sistema, facilitando assim a manutenção do software, já que os problemas, quando detectados, fazem parte de uma solução pequena, geralmente desenvolvida em algumas horas, facilitando assim a correção de problemas.

k) releases curtos: a combinação de releases curtos com a priorização de funcionalidades de maior valor junto ao cliente garante que nas primeiras liberações a parte principal do sistema esteja pronta, o que

é muito importante para a equipe pois os clientes valorizam o trabalho rápido. Desta forma, com as funcionalidades de maior valor sendo liberadas já no início do projeto, espera-se que o mesmo ocorra com o retorno financeiro, gerando assim um maior valor de retorno já no início do projeto.

l) metáforas: geralmente um software possui mais de um desenvolvedor projetista, e para manter a integridade conceitual do software o modelo XP utiliza o conceito de metáforas ao invés de termos técnicos.

m) ritmo sustentável: o modelo XP defende as pessoas envolvidas no projeto devem trabalhar em seu período normal, com adoção mínima de horas-extras. O ser humano não trabalha como uma máquina. Para uma máquina, se a entrada é duplicada, normalmente o trabalho também será. Porém para o ser humano, vários outros fatores devem ser considerados, como o cansaço e stress, e isto se traduz em resultados às vezes indesejados ao final do trabalho devido a fadiga da pessoal envolvida. Assim, problemas com prazos são tratados através da manipulação do escopo e priorização de funcionalidades.

Quando a equipe envolvida trabalha focada em seguir as práticas supracitadas, o bom resultado do início ao fim do projeto é garantido. Manter todo o este conhecimento alinhado ao longo do projeto não é tarefa fácil, exige experiência dos envolvidos e muita dedicação, porém é certo de que todos os integrantes estarão crescendo a cada dia e agregando valor à equipe.

3.1.3 Kanban

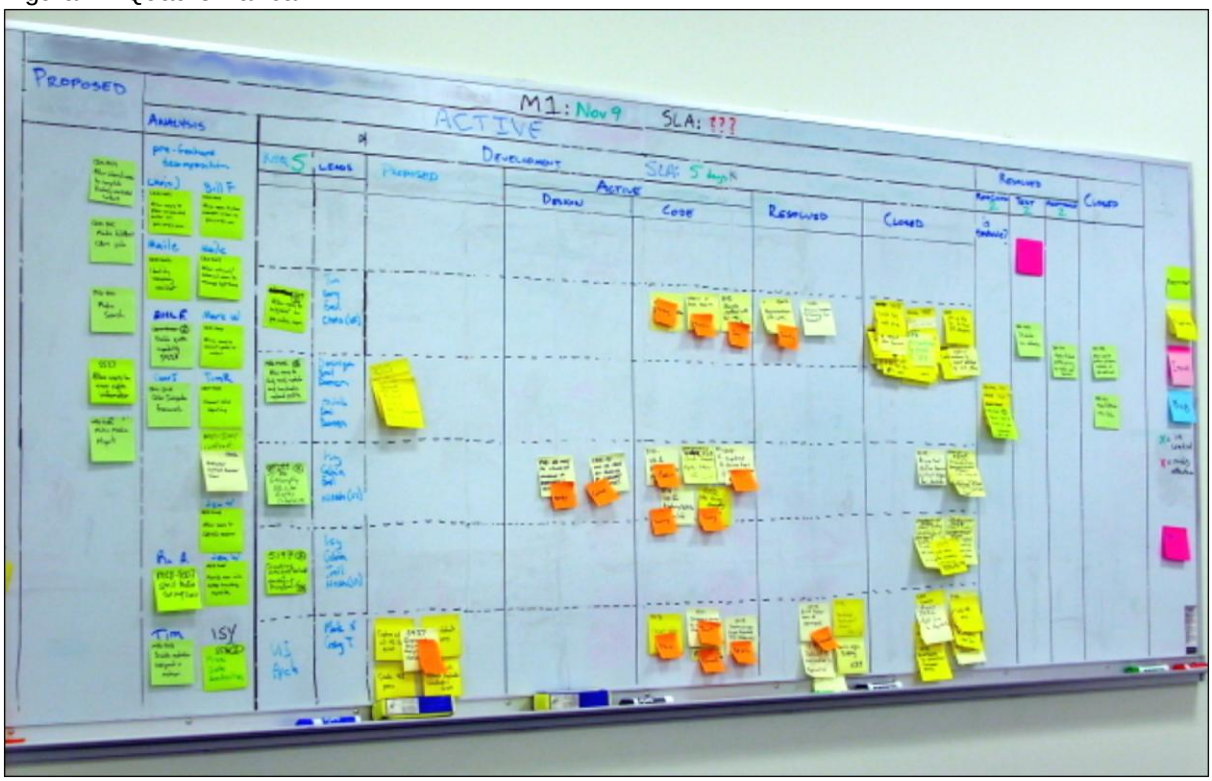
Kanban é uma expressão japonesa que se originou dos cartões utilizados nas empresas japonesas para solicitar componentes a outras equipes da mesma linha de produção, sistema este desenvolvido pela Toyota Motor Company.

O Kanban é utilizado como uma ferramenta para organizar as atividades de um projeto. Estas atividades são dispostas em um quadro através de cartões, alocadas na coluna referente ao seu status perante o projeto (pendente, em análise,

em desenvolvimento, em testes, liberada) e são manipuladas pela equipe de acordo com a evolução do projeto.

Segundo Anderson (2010), o objetivo deste modelo é estabelecer um fluxo único de atividades que começa com uma lista de atividades pendentes e finaliza com a entrega efetiva do resultado ao cliente.

Figura 2 - Quadro Kanban



Fonte: Ladas (2007)

Exemplificando, o desenvolvedor seleciona a sua próxima atividade na lista de funcionalidades pendentes e a posiciona na lista de funcionalidades em desenvolvimento. Depois de terminado, o desenvolvedor coloca o cartão da sua funcionalidade na lista de funcionalidades disponíveis para teste e novamente seleciona uma atividade pendente. O analista de testes seleciona a funcionalidade na lista de funcionalidades pendentes para testes e aloca o cartão na lista de funcionalidades em testes. Depois de finalizado, o analista de teste pode então colocar o cartão representando a funcionalidade na lista de funcionalidades finalizadas, que será disponibilizada para o cliente. Na seqüência, a equipe responsável por disponibilizar as funcionalidades prontas aos clientes verifica através dos cartões postados na lista de funcionalidades finalizadas o que deve ser liberado na próxima *release* do(s) produto(s), encerrando assim o fluxo.

As atividades podem ser organizadas através de diferentes cartões, com cores e tamanhos diferentes que possuem seu significado único, tudo a critério da equipe envolvida no projeto. Por exemplo: a equipe pode definir a complexidade das funcionalidades através da cor dos cartões a fim dos membros da equipe saberem qual funcionalidade este estaria apto a escolher para desenvolver. Cartões verdes representariam funcionalidades com baixa complexidade de desenvolvimento, cartões amarelos representariam complexidade média e cartões vermelhos representariam complexidade alta, que só colaboradores experientes poderiam assumir.

Através do tamanho dos cartões também é possível definir características como tamanho da funcionalidade, custo, entre outras opções que podem ser definidas pela equipe.

Uma forma de organizar o quadro com relação a prioridade das funcionalidades, é organizar os cartões em fila, definindo o inicio e o fim desta para que os integrantes da equipe consigam visualizar de forma rápida e fácil qual é a próxima atividade a ser realizada. Seguindo o exemplo citado anteriormente, a equipe poderia organizar o quadro com filas priorizadas de cada tipo de funcionalidade de acordo com sua complexidade, ou seja, uma fila priorizada dos cartões verdes, outra dos amarelos e por fim dos vermelhos. Assim, cada grupo de desenvolvimento teria de forma organizada a sua lista de atividades pendentes, organizada de acordo com a priorização definida pela equipe.

Para Anderson (2010), no modelo Kanban não existem iterações. Há um fluxo contínuo da entrega de funcionalidades, e a previsão destas entregas se dá com base no histórico de liberações da equipe envolvida. Como não há iterações, a lista de funcionalidades pendentes é alterada o tempo todo, cartões podem ser removidos, inseridos ou reordenados de acordo com a necessidade do cliente e priorização definida pela equipe.

Quando falamos de priorização de funcionalidades pela equipe junto ao cliente, deve-se levar em conta se a funcionalidade em questão é interessante para mais de um cliente, pois neste cenário, a tarefa de priorização deve ser realizada por uma pessoa intermediadora capaz de utilizar diferentes critérios para o estabelecimento da ordem pela qual as funcionalidades devem ser desenvolvidas. A priorização será o resultado de uma análise que leva em conta uma série de aspectos que vão desde questões técnicas até questões de negócio. Também deve

ser levado em conta o chamado “Custo do atraso”, que é o custo decorrente de se fazer uma entrega hoje, amanhã ou na próxima semana.

Segundo Ladas (2009), o modelo Kanban é considerado uma boa alternativa para projetos que:

a) já possuem produtos em ambiente de produção;
b) demandam re-organização constante da lista de atividades pendentes;
c) possuem funcionalidades pendentes interessantes a vários clientes.

Algumas características e vantagens deste modelo, segundo Ladas (2009):

a) controle do trabalho sendo realizado: o modelo Kanban limita o trabalho em execução, impedindo a sobrecarga da equipe, fazendo com que esta trabalhe sempre com sua capacidade real. Um novo cartão só é pego se outro sair primeiro;
b) a demanda de funcionalidades deve se ajustar à capacidade de produção da equipe, e não o contrário. Se a demanda não pode se adaptar a capacidade da equipe, deve haver aumento na equipe para que haja um aumento no seu rendimento.
c) identificação imediata dos problemas: Os problemas são relacionados e resolvidos no momento em que eles surgem. Isso acontece porque o modelo depende do fluxo de funcionalidades, e se um problema interrompe este fluxo, o trabalho de toda a equipe é afetado, o que contribui para que o problema seja resolvido o mais rápido possível;
d) o modelo Kanban estimula o uso de práticas ágeis de engenharia e acompanhamento do projeto de forma que as inspeções não aconteçam muito depois do desenvolvimento das funcionalidades, facilitando o controle de qualidade do software;
e) todo o trabalho da equipe envolvida está fisicamente visível, nenhuma atividade fica oculta;

Nos próximos tópicos serão abordados dois dos principais modelos tradicionais de gerenciamento de projetos.

3.2 MODELOS TRADICIONAIS

3.2.1 RUP

O Rational Unified Process (RUP) é um processo de engenharia de software utilizado nos mais diversos tipos projetos. Desenvolvido pela Rational Software Corporation, suas principais características são:

a) desenvolvimento iterativo e incremental: um ciclo de vida iterativo no contexto de um software se baseia no aumento e refinamento deste através de vários ciclos de desenvolvimento da análise, projeto, implementação e testes. Ao contrário do ciclo de vida em cascata, onde cada atividade é executada uma única vez, o ciclo de vida iterativo implica no aperfeiçoamento sucessivo de um software. Já o objetivo do desenvolvimento incremental é adicionar funcionalidades a um software durante vários ciclos de liberação (iterações).
b) orientado a objetos: O RUP utiliza técnicas de orientação a objetos em sua concepção e é projetado e documentado utilizando a notação Unified Modeling Language (UML) para ilustrar os processos em ação.
c) arquitetura robusta: arquitetura baseada em componentes (orientação a objetos), que possibilita a criação de um software que pode ser facilmente estendido, promovendo a reutilização de componentes e um entendimento intuitivo por parte da equipe envolvida.
d) análise de riscos: Os riscos são identificados e corrigidos o quanto antes no ciclo de vida do projeto, sempre prezando pela garantia da produção de software de alta qualidade, de acordo com as necessidades dos clientes e prazo previstos.
e) utilização de casos de uso: os requisitos funcionais (funcionalidades) e os requisitos não funcionais (regras de negócio, entre outros) do software são mapeados e documentados através de casos de usos e seus diagramas. Este modelo permite também a rastreabilidade entre as funcionalidades através dos seus relacionamentos entre si.

Segundo Kruchten (2010), o modelo RUP foi desenvolvido para ser aplicado a uma grande gama de projetos e pode ser considerado como um framework genérico para processos de desenvolvimento de software. Isso significa que este modelo deve ser configurado para ser utilizado de forma eficiente. A configuração pode ser feita para empresas no que tange a definição do processo padrão de desenvolvimento ou mesmo para projetos específicos.

O RUP é composto por quatro fases (RUP, 2001):

a) iniciação: na etapa inicial deste modelo o escopo do projeto é definido junto ao cliente. As funcionalidades mais importantes são mapeadas através dos casos de usos e os riscos do projeto são estimados. É realizada também a análise de viabilidade econômica do projeto. A partir do mapeamento básico das funcionalidades, o pessoal responsável deve arquitetar o software a fim de se ter maior precisão na estimativa de custos e tempo.
b) elaboração: nesta fase a equipe deve focar em diminuir todos os riscos do projeto do ponto de vista da arquitetura do software, que é o ponto mais crítico do projeto por ser muito custosa caso ocorram falhas de análise. A equipe deve também assegurar que itens como arquitetura e os requisitos do software estejam estáveis o suficiente para prosseguir para a próxima etapa.
c) construção: nesta fase a equipe envolvida no projeto deve desenvolver o software de acordo com todas as informações já levantadas nas fases anteriores. Os requisitos menos importantes que não foram concluídos são esclarecidos e finalizados. Todo o trabalho é realizado de forma que o custo do projeto seja minimizado e a eficiência da equipe otimizada, mantendo o nível de qualidade do software.
d) transição: neste ponto atividades como teste do software e garantia da qualidade são executadas para garantir que o software a ser entregue ao cliente está de acordo com o escopo do projeto e livre de erros desde sua instalação até o uso em produção. Outro ponto importante nesta fase é o treinamento dos usuários do sistema desenvolvido e também da equipe técnica que irá atendê-los sempre que necessário.

Ao final de cada fase a equipe deve se certificar de que todos os objetivos até o momento foram alcançados com sucesso.

O modelo RUP possui ao todo nove disciplinas que definem quais atividades a equipe deve desempenhar em cada etapa do projeto. Basicamente, estas disciplinas definem o fluxo de trabalho a ser seguido dentro deste modelo (RUP, 2001).

Na Modelagem de Negócio, a equipe de negócio deve se ater ao negócio do cliente para identificar qual o fluxo de trabalho utilizado, quais os principais processos e quais as necessidades básicas para o funcionamento da organização e quais suas maiores dificuldades. O objetivo neste ponto não é obter informações detalhadas, mas sim entender os processos e a estrutura do cliente.

Posteriormente, na disciplina de Requisitos, a equipe de negócio deve realizar uma análise dos problemas identificados anteriormente e buscar compreender a necessidade do cliente para arquitetar uma solução. Após a compreensão das necessidades a equipe deve especificar a solução de acordo com a análise realizada. O nível de especificação das funcionalidades fica a critério da equipe envolvida.

Na disciplina de Análise e Design a equipe de desenvolvimento deve arquitetar uma solução de software para iniciar o desenvolvimento das funcionalidades especificadas pela equipe de negócio, a fim de deixar a base do software pronta para suprir a necessidade de funcionalidades do cliente.

A próxima disciplina, chamada de Implementação, representa a fase onde a equipe de desenvolvimento codifica as funcionalidades de acordo com o trabalho realizado pela equipe de negócio. É importante que neste momento os desenvolvedores realizem os testes unitários a fim de minimizar ao máximo o retorno de funcionalidades pela equipe de testes.

No momento dos Testes, quinta disciplina, a equipe de testes deve garantir a estabilidade do que foi liberado pela equipe de desenvolvimento testando todas as funcionalidades que foram implementadas e verificando se estão de acordo com que a equipe de negócios definiu inicialmente, garantindo assim a qualidade do software na entrega ao cliente.

A sexta disciplina, Implantação, trata do momento em que o software está pronto para entrar em produção e é disponibilizado ao cliente. Além de todo o processo que vai da instalação até a disponibilização do sistema na produção, é

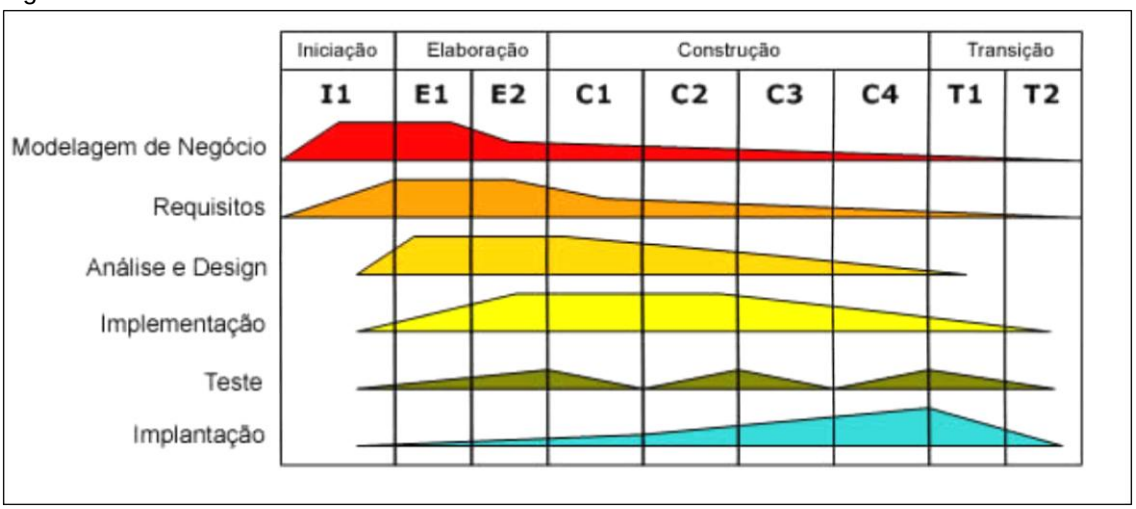
necessário também preparar treinamentos e material de suporte para que o cliente possa operar o software da forma correta.

As outras três disciplinas (Gerenciamento de configuração e mudança, Gerenciamento de projeto, Ambiente) envolvem questões administrativas que são executadas ao longo do projeto.

Durante a execução das atividades descritas nas disciplinas supracitadas, os artefatos do projeto são gerados de acordo com o fluxo de trabalho. Estes artefatos têm como objetivo registrar através de documentos os diversos momentos do projeto, como documentos gerados na fase de levantamento de requisitos, desenvolvimento, testes, entre outros.

Os artefatos são criados por integrantes da equipe que por sua vez possuem um ou mais papéis definidos dentro do projeto, como Analista de Negócios, Analista de Testes, Desenvolvedor, entre outros.

Figura 3 – Gráfico RUP



Fonte: Do autor

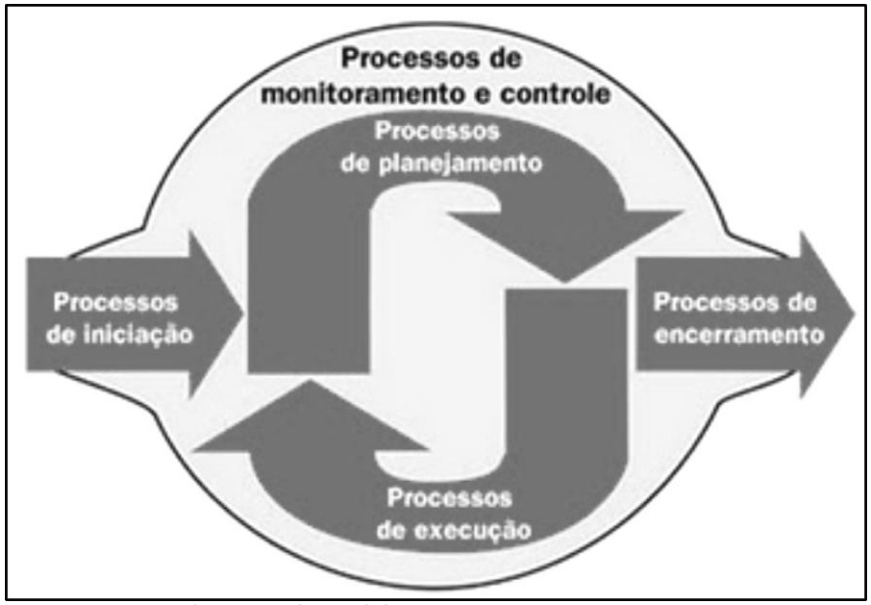
O gráfico exibido conforme Figura 3, demonstra um cruzamento entre as quatro fases do modelo RUP e suas seis principais disciplinas conforme explanado anteriormente. Através do gráfico é possível visualizar a incidência de cada disciplina ao longo das fases do modelo.

Conforme o fluxo de trabalho do modelo RUP, inicialmente há maior participação do projeto pela equipe de negócios, em seqüência pela equipe de desenvolvimento, testes e por fim a equipe de implantação.

3.2.2 PMBOK

O PMBOK é um guia que contém uma vasta base de conhecimento acumulado no que diz respeito a gerenciamento de projetos. Criado e mantido pelo Project Management Institute (PMI), este modelo é conhecido como o guia de boas práticas no gerenciamento de projetos, e em razão disso, utilizado como base pelo PMI. Este guia é baseado em processos e subprocessos que descrevem de forma organizada quais as atividades que devem ser realizadas em cada etapa do projeto. Segundo o PMBOK (2008), todos os projetos, independentes de serem pequenos ou grandes, podem assumir um ciclo de vida genérico representado por quatro fases, que são Início do projeto, Organização e preparação, Execução das atividades e Encerramento do projeto. Dentro do ciclo de vida do projeto existem cinco processos de gerenciamento conhecidos como “grupos de processos de gerenciamento de projetos”.

Figura 4 – Processos de gerenciamento do PMBOK



Fonte: PMBOK (2008, Adaptado)

A figura 4 ilustra de forma clara o fluxo dos processos gerenciais detalhados abaixo.

a) processo de iniciação: processo inicial do projeto, neste ponto as partes interessadas no projeto são identificadas e o escopo inicial é definido. Os recursos financeiros necessários inicialmente são

garantidos. O Termo de Abertura do Projeto envolvendo cliente x empresa deve ser criado e aprovado para continuação do projeto.

b) processo de planejamento: neste momento do projeto, a relação entre a equipe e o cliente deve ser forte para garantir o desenvolvimento completo e correto do escopo e o refinamento dos objetivos do projeto. Tópicos como a lista de atividades do projeto, cronograma, estimativa de custos, recursos a serem utilizados, entre outros, são definidos, e com base nas informações levantadas, o Plano de Gerenciamento do Projeto é criado contendo toda a documentação necessária para servir como fonte principal de informações sobre como o planejamento, execução, monitoramento, controle e encerramento do projeto se dará.
c) processo de execução: nesta etapa do projeto, as atividades são executadas conforme as definições documentadas no Plano de Gerenciamento do Projeto. As auditorias de qualidade do projeto devem ser realizadas e a equipe deve se certificar de que o que está sendo produzido está alinhado com as necessidades e expectativas do cliente para garantia do sucesso do projeto.
d) processo de monitoramento e controle: este processo consiste no monitoramento contínuo do projeto com o objetivo de garantir a “saúde” do projeto em todas as suas fases. Os tópicos já citados anteriormente como recursos, cronograma, custos, entre outros, são reavaliados e controlados durante todo o projeto.
e) processo de encerramento: nesta etapa do projeto a equipe deve se certificar de que todos os outros processos já foram terminados para então formalizar o término do projeto.

Tópicos do projeto como cronograma, riscos, custos, entre outros, são tratados dentro do PMBOK como áreas de conhecimento, sendo que estas já foram abordadas no item 2.1 deste documento.

Dentro do próximo capitulo será abordada a Teoria das Restrições, teoria esta utilizada como base para criação da Teoria da Corrente Crítica, também abordada neste documento.

4 TEORIA DAS RESTRIÇÕES

A Teoria das Restrições (TOC) teve seu início na década de 70, quando o físico Israelense Eliyahu Goldratt se envolveu com problemas de logística de produção de uma organização.

Goldratt elaborou então um método de administração da produção totalmente novo. O método elaborado foi muito bem visto pelo mercado e outras empresas se interessaram em aprender sua técnica. A partir daí, Goldratt se dedicou a continuar o seu trabalho e a disseminá-lo.

Nos dias de hoje, a TOC é composta por dois campos: os Processos de Raciocínio e os Aplicativos Específicos (como logística de produção). Os processos de raciocínio da TOC foram além da área de Administração e hoje são utilizados em muitas outras áreas de conhecimento, formando a base de toda a teoria. Esta teoria é hoje amplamente utilizada nas áreas de manufatura, logística e distribuição, cadeia de suprimentos, gerenciamento de projetos, marketing e vendas, contabilidade, prestação de serviços, tecnologia da informação, engenharia de software, saúde, educação, entre outras.

A Teoria das Restrições contribuiu muito para o mundo da administração através do seu processo de melhoria contínua. Este processo é definido por cinco etapas (GOLDRATT, 1990):

a) identificar as restrições do sistema: da mesma maneira que em uma corrente sempre existe um elo mais fraco que pode se partir, em uma organização sempre existirá um recurso que limitará a sua capacidade. Para buscar a otimização da produção, a organização deve identificar o “elo mais fraco” para então tomar uma ação na próxima etapa.

b) decidir como a restrição será tratada: com as restrições já identificadas, a organização deve otimizar ao máximo esta restrição a fim de maximizar o fluxo neste ponto.

c) subordinar os recursos ao elo fraco: todos os outros recursos da organização devem seguir no mesmo ritmo da restrição identificada, nem mais rápido e nem mais devagar. Trabalho a menos resultaria em diminuição no ritmo da produção, ao passo que trabalho a mais

resultaria em esforço extra à toa, já que existe um recurso limitando a produção.

d) elevar a restrição do sistema: após extrair o máximo da restrição no segundo passo, neste momento a organização deve considerar várias alternativas para investir mais nas restrições a fim de quebrá-las. Neste momento, outras restrições são identificadas e então o processo volta a etapa um.

e) não permitir que a inércia cause uma restrição: a partir do momento em que uma restrição é quebrada, a organização deve continuar em busca da melhoria contínua. Toda a organização possui pelo menos uma restrição, e ela deve estar sempre em foco para que a organização nunca pare de se desenvolver.

As restrições de uma organização não são necessariamente ruins, algumas apenas existem. Se uma organização não possuísse nenhuma restrição, seu desempenho seria teoricamente infinito. De acordo com os pressupostos da Teoria das Restrições, restrição é qualquer coisa que limita uma organização em conseguir maior desempenho no seu fluxo de trabalho. Pode-se afirmar que toda organização possui pelo menos uma restrição ou que toda corrente possui sempre um elo mais fraco.

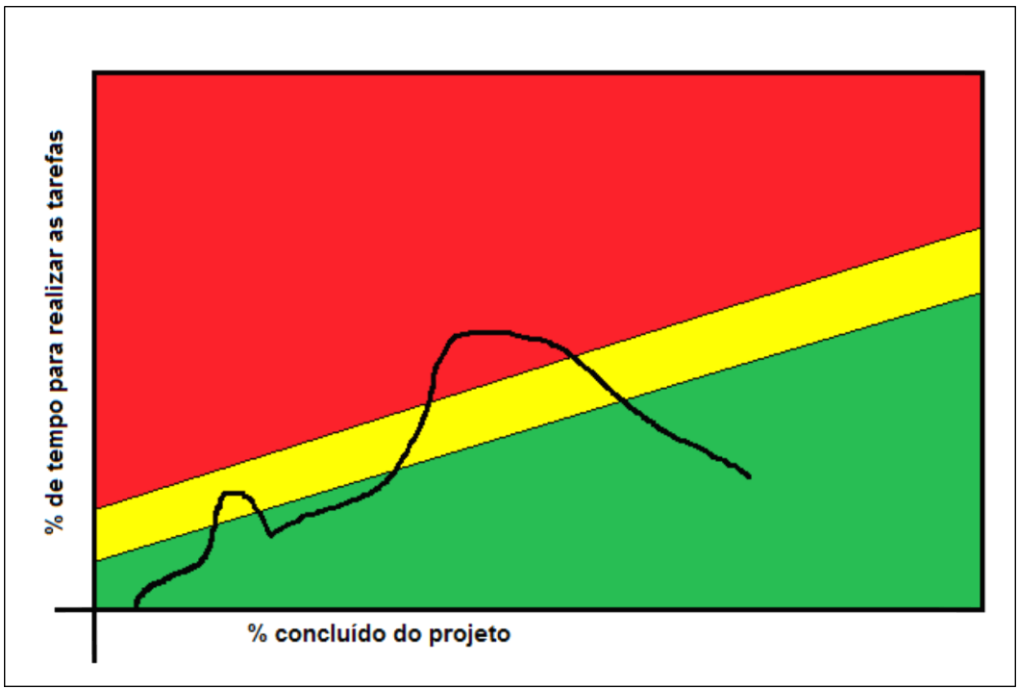
Na TOC, os processos são descritos através da utilização de três perguntas fundamentais: “O que mudar?”, “Mudar para o que?” e “Como causar a mudança?”. Estas perguntas fornecem a estrutura para os processos de raciocínio da TOC. Os processos de raciocínio são um conjunto de ferramentas e processos que permite que um indivíduo ou grupo resolva um problema ou desenvolva uma estratégia integrada com o rigor e a lógica de causa e efeito, buscando inicialmente os sintomas e terminando com um plano de ação detalhado que coordene as atividades de todos os envolvidos na implementação da solução para a restrição.

5 TEORIA DA CORRENTE CRITICA

Criada por Eliyahu M. Goldratt, a Corrente Crítica se inicia a partir da observação básica de que os problemas comuns a todos os projetos são a alta probabilidade de extrapolação de orçamento, extrapolação de tempo, e comprometimento do conteúdo. A Corrente Crítica foi desenvolvida para ser implantada como um projeto estratégico de gestão a fim de evitar atrasos no causados pela Lei de Parkinson e Síndrome do Estudante, abordados em seqüência. O objetivo é tentar minimizar o impacto da Lei de Parkinson e Síndrome do Estudante através da construção do cronograma com duração das estimativas baseadas em um nível de confiança de 50%.

A idéia principal desta metodologia, segundo Goldratt (1998), é realizar uma estimativa de tempo das atividades do projeto da forma com que a equipe achar melhor. A partir daí, estas estimativas devem ser cortadas pela metade e este é o tempo que a equipe envolvida terá para realizar as tarefas. A diferença entre os tempos (estimativa inicial e estimativa 50%) é colocada ao final do projeto como um pulmão de segurança.

Figura 5 - Gráfico de acompanhamento da Corrente Crítica



Fonte: Do autor

Através do gráfico de acompanhamento, conforme figura 5, é possível visualizar se o ritmo da produção está de acordo com o tempo que foi disponibilizado para realização das tarefas. As áreas “Amarela” e “Verde” mostram a porcentagem relativa aos 50% estimados para realização da tarefa, enquanto a área “Vermelha” significa atraso nas atividades. Quando o indicador invade a faixa amarela, significa que o gestor deve ficar atento ao que está ocorrendo e possuir um plano de ação, porém sem intervir. Quando o indicador invade a área vermelha, o plano de ação deve ser posto em prática para resolução dos problemas relacionados.

O gestor do projeto deve ficar atento também quando o indicador fica muito tempo na faixa vermelha ou muito tempo na faixa verde, pois isto pode indicar falha na estimativa inicial das tarefas. Normalmente é de se esperar que o indicador oscile entre as três faixas.

Inicialmente o uso desta técnica pode indicar muitos atrasos devido ao fato de a equipe não conhecer a sua real capacidade de produção. As estimativas iniciais podem não ser ideais, e ao aplicar a taxa de 50% sobre estas estimativas, pode dificultar ainda mais o cumprimento do cronograma. Porém, esta forma agressiva de estimar o tempo consegue minimizar ao máximo a incidência da Lei de Parkinson e da Síndrome do Estudante, diminuindo o tempo efetivo de duração do projeto.

5.1 SINDROME DO ESTUDANTE

A Síndrome do Estudante pode ser melhor entendida quando associada ao comportamento dos alunos em seu período letivo. A maioria dos alunos realiza seu trabalho sempre na última hora possível, e este comportamento também é observado dentro de organizações.

Quando várias tarefas são delegadas a uma pessoa, esta acaba realizando-as sempre no último momento. Quando ocorrem acontecimentos imprevistos, a entrega da tarefa acaba atrasando trazendo prejuízo para a organização.

5.2 LEI DE PARKINSON

A Lei de Parkinson também é muito comum dentro de organizações do mais variado tipo de negócio. Esta lei afirma que quando uma tarefa tem o seu prazo maior do que o necessário para sua execução, a pessoa responsável ao invés de manter o ritmo de trabalho e terminar a tarefa antes do prazo, acaba diminuindo o seu ritmo para finalizar a tarefa no prazo em que lhe foi concedido.

Este comportamento é muito comum e também muito prejudicial a qualquer organização, pois apesar da pessoa cumprir o prazo de sua atividade, este foi superestimado. Como tudo ocorre dentro do prazo, é muito provável que as próximas estimativas sigam a mesma metodologia, fazendo com que a equipe trabalhe sempre abaixo da sua capacidade máxima de produção.

6 TRABALHOS RELACIONADOS

O documento com o tema “Gerencia de Projetos de Software”, escrito por Ana Cristina Rouiller, conceitua o gerenciamento de projetos e aborda de forma clara as disciplinas do PMBOK na gestão de projetos.

A monografia escrita por Ricardo Cortez Toledo, com o tema “Integração das metodologias RUP e PMI no desenvolvimento de projetos de software”, aborda um dos modelos tradicionais mais utilizados no mundo do gerenciamento de projetos e também aborda toda a metodologia de gerenciamento de projetos desenvolvida pela equipe do PMI.

Escrito por Maria Isabel Palmerio Marcantonio, o trabalho com tema “A Corrente Crítica Aplicada na Ferramenta de Gestão de Projetos MS Project” faz uma ligeira abordagem sobre o gerenciamento de projetos, Teoria das Restrições e Corrente Crítica, e mostra como aplicar este conhecimento em um projeto com auxílio do software Microsoft Project.

O trabalho escrito por Carolina Garcia e Gustavo Severo de Borba com o título “Gerenciamento de Projetos Aliado a Ótica da TOC (Corrente Crítica)” conceitua o gerenciamento de projetos e a Corrente Crítica, dissertando também sobre a aplicação destes conceitos em um projeto real de uma indústria do ramo farmacêutico.

7 IMPLEMENTAÇÃO DO SOFTWARE

Na área de projetos, conforme visto ao longo deste documento, existe muito conhecimento acumulado acerca das maneiras de se gerenciar um projeto. Dentro dos diversos modelos de gerenciamento, existem diferentes formas de executá-lo com relação ao gerenciamento de custos, prazos, recursos, porém nenhum modelo aborda uma maneira diferenciada de gestão de atividades quanto a sua estimativa de tempo.

A estimativa de tempo de uma atividade é muito importante para um projeto, pois estimar o tempo das atividades ajuda a ditar o ritmo com que o projeto andará. Quando uma equipe trabalha com folga, ou seja, o tempo estimado foi além do necessário, é normal que a equipe se acomode por possuir prazo para isso, deixando de trabalhar com sua capacidade máxima de produção.

Atualmente, os gerentes de projeto utilizam técnicas tradicionais para estimativa de tempo das atividades, e este trabalho propõe a criação de um software que gerencie o cronograma de um projeto, aplicando as técnicas e diretrizes da Teoria da Corrente Crítica para realizar estimativas de tempo agressivas, com o intuito de estimular a equipe envolvida e aumentar a produtividade, diminuindo assim custos e o prazo do projeto.

7.1 METODOLOGIA

Primeiramente foi realizado o levantamento de material bibliográfico para estudo através de pesquisas na biblioteca da UNESC, na Biblioteca Municipal de Criciúma e em artigos / livros encontrados na internet.

Após levantar o material necessário, foram estudadas metodologias de gerenciamento de projetos, assim como a própria disciplina de gerenciamento de projetos para obter um melhor entendimento do assunto antes de iniciar o estudo do tema deste trabalho que é a Teoria da Corrente Crítica. Em seqüência foi realizado o estudo da Teoria das Restrições e Teoria da Corrente Crítica para dar início a escrita do documento.

Finalizado o estudo, foi iniciado o desenvolvimento da ferramenta, contemplando o conhecimento obtido através dos estudos realizados.

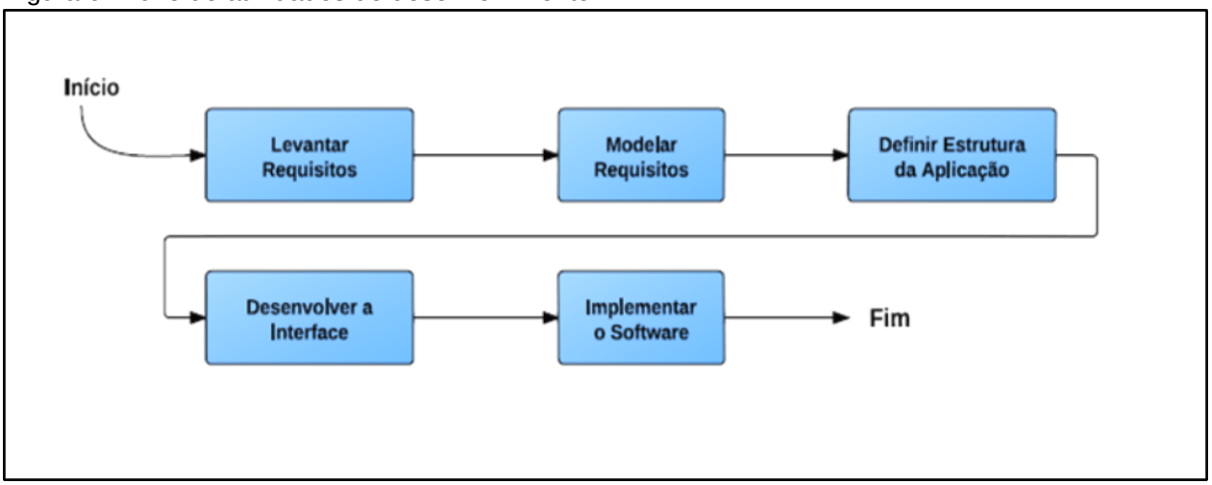
7.2 DESENVOLVIMENTO DA APLICAÇÃO

Considerando os pré requisitos para desenvolvimento do software, foi realizada uma análise para definir qual tecnologia seria mais adequada à proposta deste trabalho. Os principais critérios utilizados para escolha da ferramenta foram a agilidade de desenvolvimento, simplicidade na manutenção do código-fonte e facilidade de uso por parte do usuário final.

O software proposto neste trabalho foi desenvolvido sob a plataforma .Net C# da Microsoft, utilizando a ferramenta Microsoft Visual Studio 2010 para o desenvolvimento do sistema. Os dados são persistidos em um banco de dados Microsoft SQL Server 2008.

O fluxo exibido conforme figura 6 ilustra a seqüência de atividades seguida até o término do desenvolvimento da aplicação, as quais serão detalhadas nos tópicos a seguir.

Figura 6. Fluxo de atividades do desenvolvimento



Fonte: Do autor.

7.2.1 Levantamento de Requisitos

Nesta etapa de desenvolvimento o foco se deu na análise da necessidade. Para evitar a possibilidade de desenvolver algo fora do escopo, foi necessário primeiramente listar o que era realmente necessário desenvolver, e depois definir as regras de negócio que o software iria seguir.

Os requisitos do software desenvolvido foram definidos baseando-se em todo o conhecimento obtido durante o desenvolvimento deste projeto, sendo eles:

a) o sistema deve permitir ao usuário cadastrar as atividades do projeto. Neste cadastro o usuário informa dados básicos da atividade como o recurso responsável, a data de início e término planejada e também as datas de inicio e término reais.
b) o sistema deve permitir ao usuário cadastrar os recursos do projeto. Neste cadastro o usuário informa o nome do recurso que será utilizado posteriormente no planejamento das atividades do projeto.
c) o sistema deve realizar um cálculo sobre as atividades cadastradas, aplicando as regras e diretrizes da Teoria da Corrente Crítica para ajustar as datas de início e término de cada atividade.
d) o sistema deve gerenciar o pulmão do projeto sempre que o cálculo da Corrente Crítica for aplicado.
e) o sistema deve possuir um gráfico que demonstra o consumo do pulmão para cada atividade do projeto ao longo do seu desenvolvimento.

A partir do levantamento dos requisitos básicos do sistema, foi possível iniciar a modelagem destes requisitos a fim de possuir um material de análise documentado, facilitando assim o desenvolvimento da aplicação e evitando possíveis desvios ao longo da sua implementação.

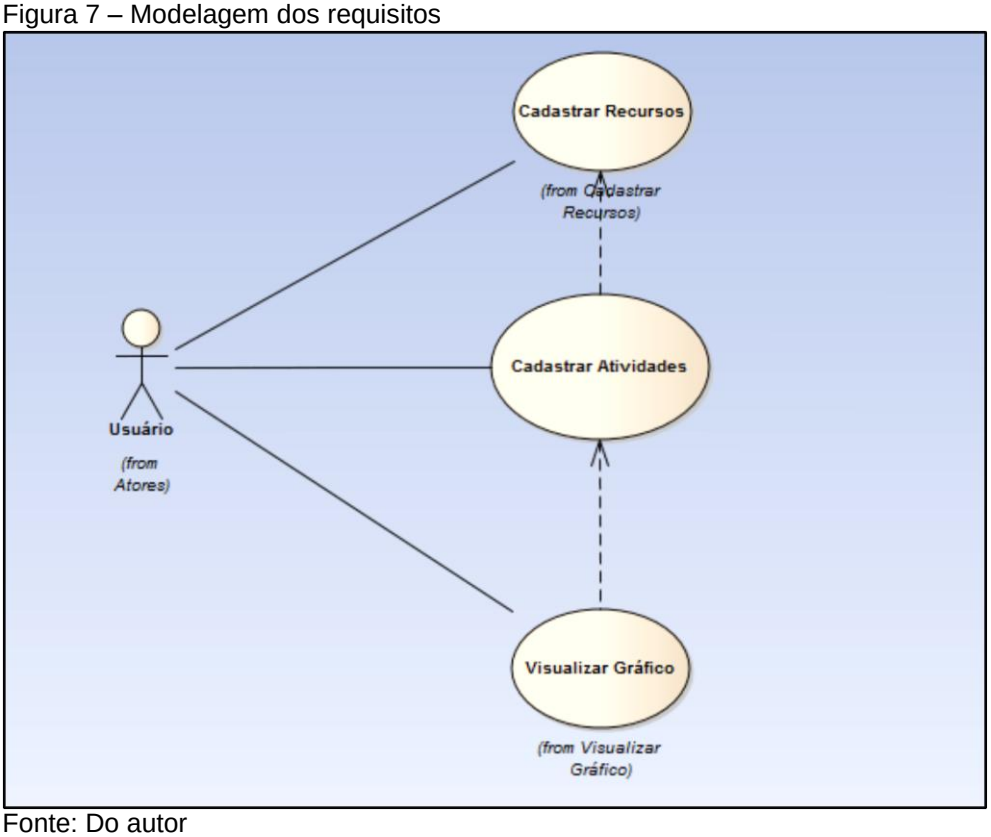
7.2.2 Modelagem dos Requisitos

Na modelagem dos requisitos, os casos de uso do sistema foram mapeados e detalhados a fim de se ter a definição das funcionalidades compreendidas pela aplicação desenvolvida.

Através da análise realizada sob o levantamento de requisitos, foi possível visualizar que as necessidades seriam atendidas dividindo-as em três telas (Casos de Uso).

Abaixo é exibido o diagrama de caso de uso do sistema, que abrange todas as funcionalidades desenvolvidas. O diagrama foi modelado em uma versão de avaliação gratuita do software Enterprise Architect, da Sparx Systems.

Figura 7 – Modelagem dos requisitos



Fonte: Do autor

A imagem acima demonstra o relacionamento do usuário com as funcionalidades do sistema, e também o relacionamento das funcionalidades entre si. É possível ver que o cadastro de Atividades, principal cadastro do sistema, depende do cadastro de Recursos para ter o correto funcionamento, enquanto o gráfico da Teoria da Corrente Crítica depende das atividades cadastradas para seu funcionamento.

Com todos os requisitos do programa modelados, foi possível iniciar então a modelagem da estrutura da aplicação a ser desenvolvida.

7.2.3 Definição da Estrutura da Aplicação

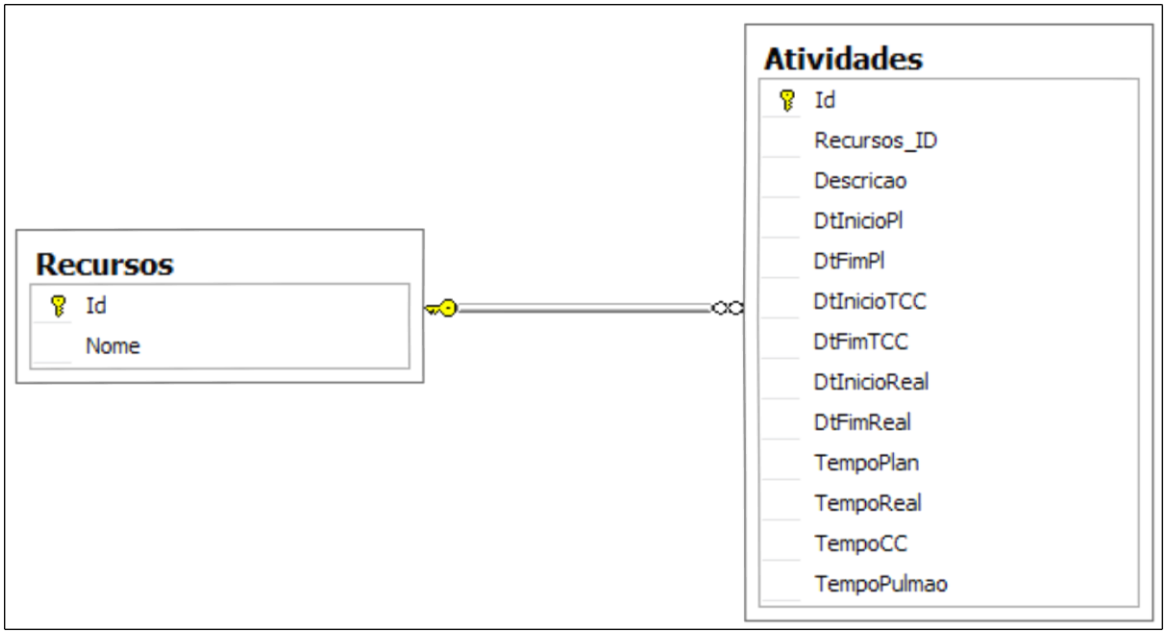
Para atender as necessidades do software a ser desenvolvido, foi necessário criar duas tabelas no banco de dados.

A tabela “*Recursos*” foi criada com o intuito de centralizar as informações referentes aos recursos responsáveis pelas atividades do projeto. Organizando os dados desta forma, é possível extrair informações direcionadas a determinado recurso com maior facilidade, dentre outras vantagens.

Na tabela “*Atividades*” estão concentrados os principais dados do software desenvolvido. Nesta tabela são registradas todas as atividades do projeto, bem como suas datas de inicio e fim, e o recurso responsável.

Através da figura 8 é possível visualizar o relacionamento entre as tabelas do sistema.

Figura 8 – Modelagem do banco de dados



Fonte: Do autor

Conforme é possível visualizar através da figura 8, a tabela de Atividades possui relacionamento com a tabela de Recursos, de forma que um recurso possa ser atrelado a várias atividades.

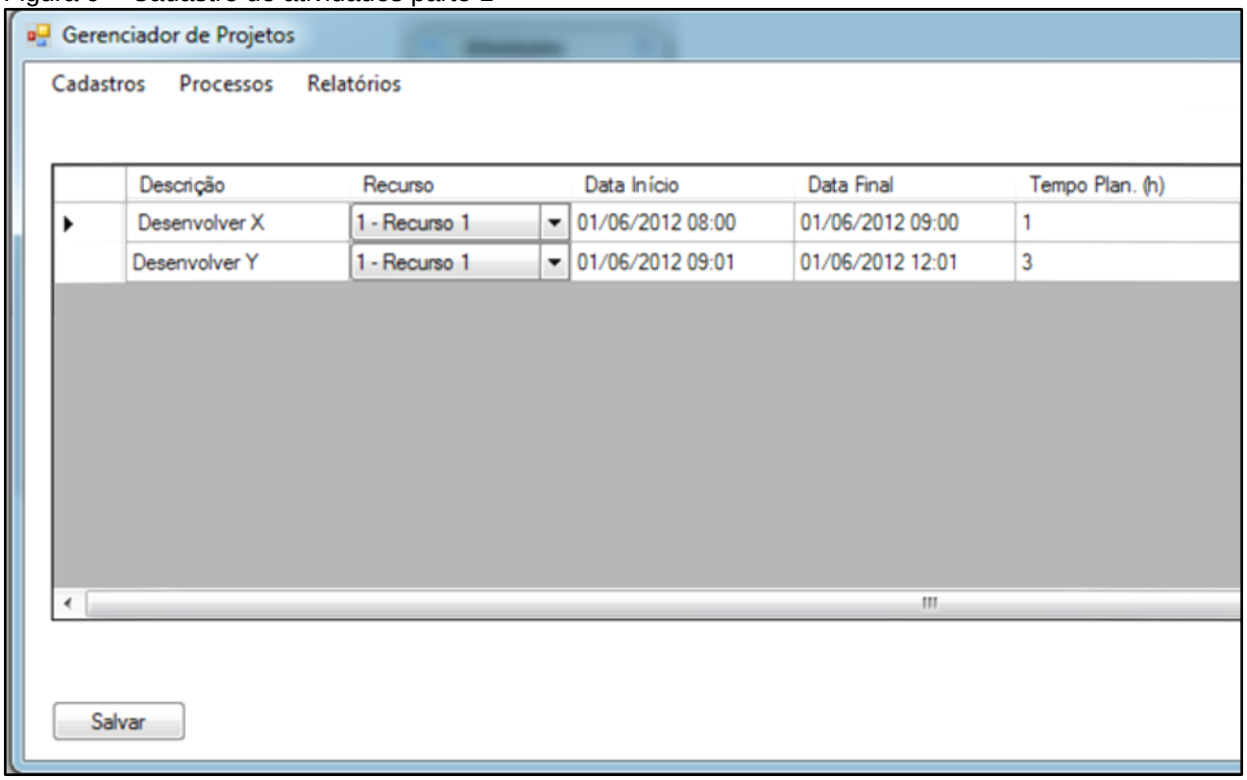
Com este modelo de banco de dados foi possível desenvolver as telas do software conforme é exibido na próxima sessão.

7.2.4 Desenvolvimento da Interface

Para atender os requisitos do software foi necessário implementar uma interface que possuísse uma tabela de dados. Nesta tabela, o usuário é capaz de listar todas as atividades do seu projeto, definir suas datas e recurso responsável.

As figuras 9 e 10 mostram a tela principal do software, onde o usuário detalha as atividades do projeto.

Figura 9 – Cadastro de atividades parte 1



Fonte: Do autor

Coluna “Descrição”: Nesta coluna o usuário deve informar uma descrição para que se possa identificar a atividade que está sendo cadastrada.

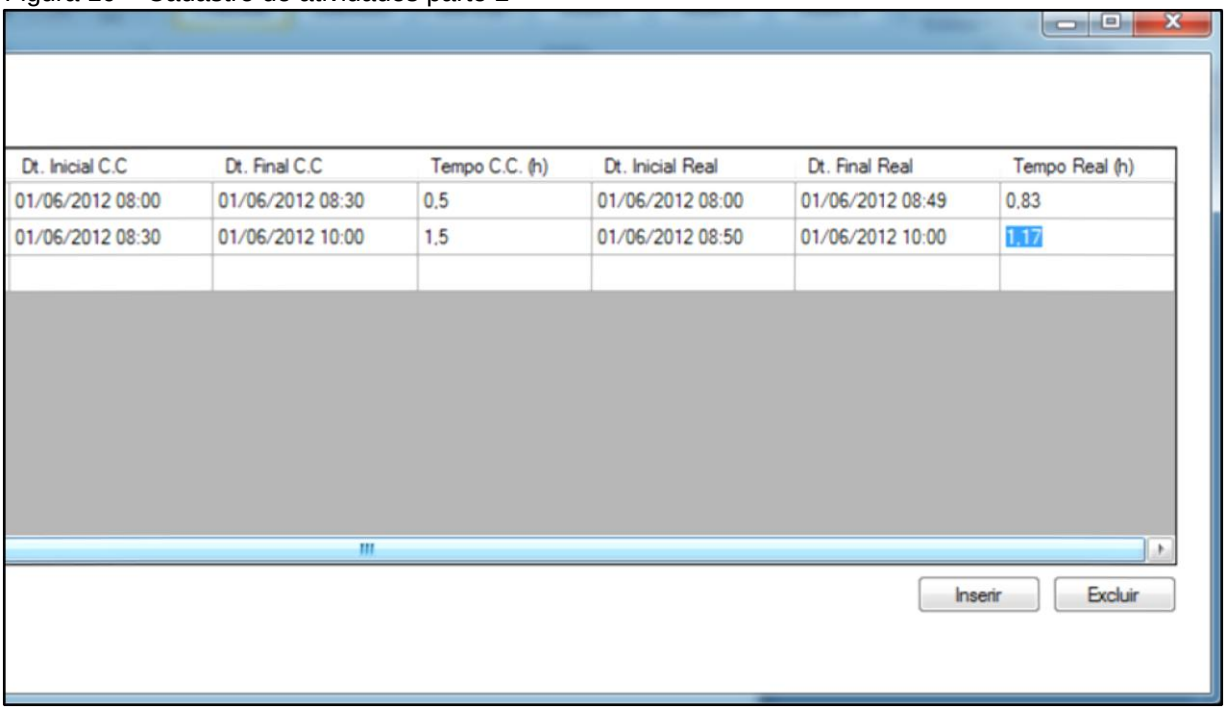
Coluna “Recurso”: Nesta coluna são exibidos todos os recursos cadastrados no sistema para que o usuário possa atribuí-lo à atividade cadastrada.

Coluna “Data Início”: Nesta coluna o usuário deve informar qual a data planejada pelo gerente de projetos para que a atividade seja iniciada pelo recurso alocado.

Coluna “Data Final”: Nesta coluna o usuário deve informar qual a data limite planejada pelo gerente de projetos para que a atividade tenha seu término.

Coluna “Tempo Plan. (h)”: Nesta coluna o usuário deve informar o tempo total em horas de duração da atividade. Caso o usuário preencha os campos “Data Início” e “Data Final”, este campo será preenchido automaticamente. O usuário também tem a opção de preencher o campo “Data Início” e depois preencher o campo “Tempo Plan. (h)”. Neste caso, o campo “Data Final” será preenchido automaticamente de acordo com os dados informados.

Figura 10 – Cadastro de atividades parte 2



Fonte: Do autor

Coluna “Dt. Inicial C.C”: Esta coluna não pode ser editada pelo usuário. Os dados são gerados quando a rotina de cálculo da Teoria da Corrente Crítica é executada. O cálculo que gera a informação exibida neste campo será detalhado na próxima sessão.

Coluna “Dt. Final C.C”: Esta coluna não pode ser editada pelo usuário. Os dados são gerados quando a rotina de cálculo da Teoria da Corrente Crítica é executada. O cálculo que gera a informação exibida neste campo será detalhado na próxima sessão.

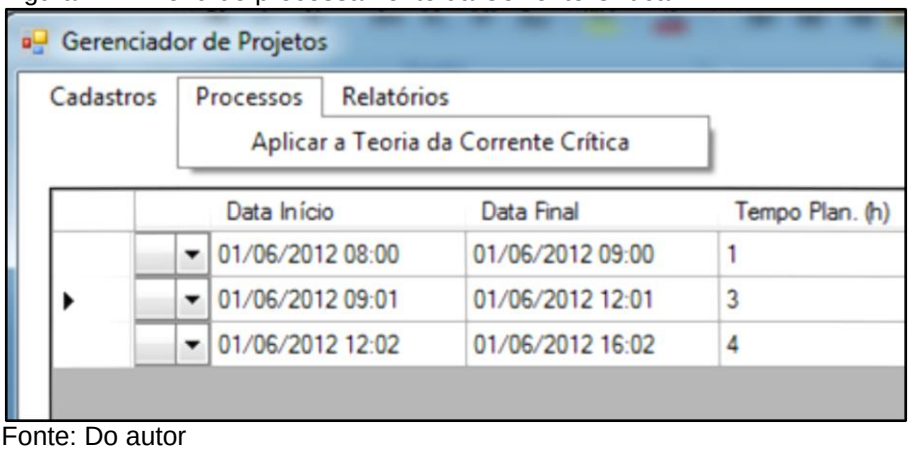
Coluna “Tempo C.C (h)”: Esta coluna não pode ser editada pelo usuário. Os dados são gerados quando a rotina de cálculo da Teoria da Corrente Crítica é executada. O cálculo que gera a informação exibida neste campo será detalhado na próxima sessão.

Coluna “Dt. Inicial Real”: Nesta coluna o usuário deve informar a data real em que a atividade foi iniciada pelo recurso.

Coluna “Dt. Final Real”: Nesta coluna o usuário deve informar a data real em que a atividade foi finalizada pelo recurso.

Coluna ”Tempo Real (h)”: Nesta coluna o usuário deve informar o tempo total real em horas de duração da atividade. Caso o usuário preencha os campos “Dt. Inicial Real” e “Dt. Final Real”, este campo será preenchido automaticamente. O usuário também tem a opção de preencher o campo “Dt. Inicial Real” e depois preencher o campo ”Tempo Real (h)”. Neste caso, o campo “Dt. Final Real” será preenchido automaticamente de acordo com os dados informados.

Figura 11 – Menu de processamento da Corrente Crítica

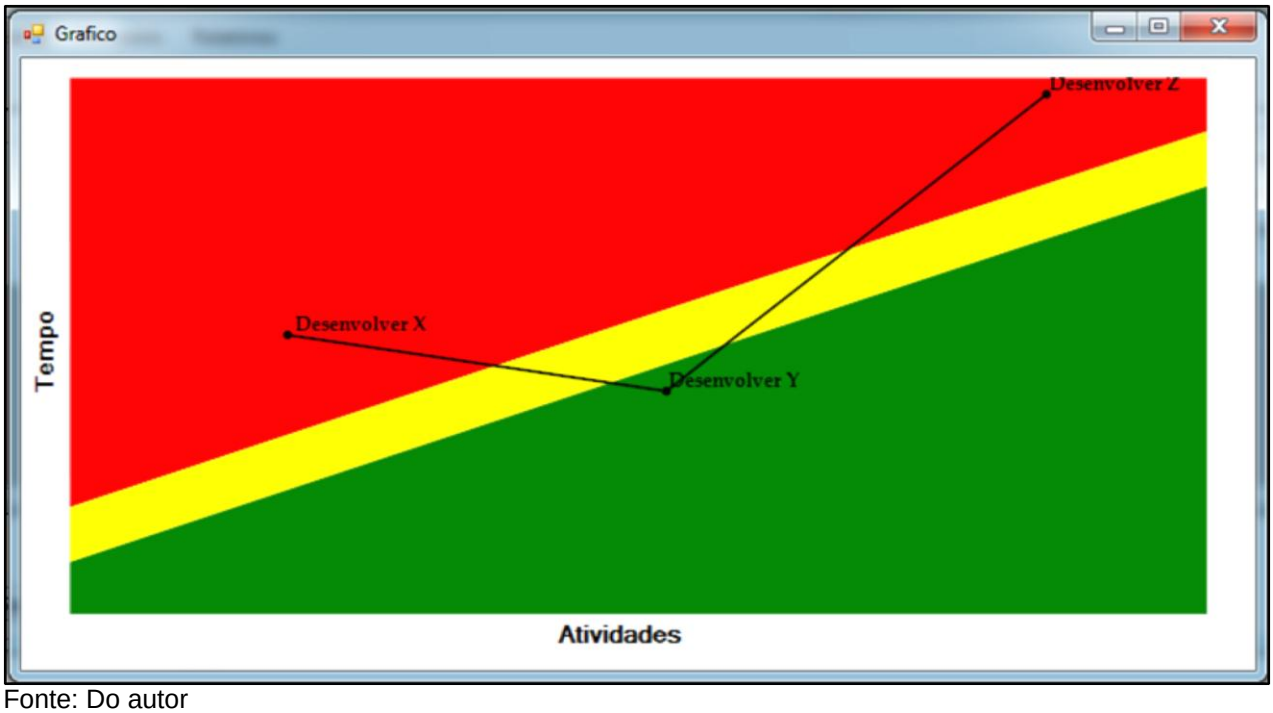


Fonte: Do autor

Conforme é ilustrado na figura 11, para executar a rotina de cálculo da Teoria da Corrente Crítica o usuário deve acessar o menu Processos > Aplicar a Teoria da Corrente Crítica. Ao pressionar este botão do menu, o sistema irá executar os cálculos necessários de acordo com as informações dispostas na tabela.

Após o preenchimento correto da tabela, execução da rotina de cálculo da Teoria da Corrente Crítica e informar as datas reais de execução da atividade, é possível visualizar o gráfico de consumo do pulmão do projeto, conforme exemplo na figura 12.

Figura 12 – Gráfico de consumo do pulmão do projeto



Fonte: Do autor

O eixo Y do gráfico representa o tempo que a atividade levou para ser finalizada, enquanto no eixo X as atividades são dispostas com relação a sua data de término.

Quando a atividade se encontra na área verde do gráfico, significa que ela está sendo realizada dentro do tempo estimado pela Teoria da Corrente Crítica. Na área amarela, significa que o tempo de execução passou do planejado, porém a diferença é tolerável. Quando a atividade se encontra na área vermelha, significa que a atividade está consumindo o tempo pulmão do projeto.

Na próxima sessão será abordada a forma como o software foi implementado.

7.2.5 Implementação

Para melhor organização do projeto, o conceito Model View Control (MVC) – Modelo Visão Controle – foi utilizado parcialmente. Neste modelo, o projeto é dividido em três camadas de desenvolvimento independentes. No caso do projeto desenvolvido, este conceito não foi seguido completamente, porém boa parte das

melhores práticas de projeto de software foram utilizadas a fim de facilitar a manutenção do código.

Na raiz do projeto, o código foi dividido em três pastas: Model, Utils e View. A pasta “Model” contém todos os códigos-fontes relacionados a banco de dados, ou seja, códigos para consulta, adição e atualização das tabelas do sistema. Na pasta “Utils” está implementada toda a lógica de negócio do software. Já na pasta “View” estão contidos os fontes relacionados a interface do software.

A interface foi desenvolvida através da utilização da biblioteca “*System.Windows.Forms*”. Esta biblioteca contém todos os objetos necessários para o desenvolvimento da interface, como formulários com eventos gerenciados, tabelas, botões, entre outros.

Os principais controles realizados nas telas do software são baseados em eventos, ou seja, quando determinado evento ocorre, como um simples foco em um campo, um código específico é executado. As validações e ações nas telas foram todas desenvolvidas desta maneira, através de eventos.

Para persistência dos dados foi utilizado o “*Entity Framework”*, conjunto de bibliotecas da plataforma .Net que possui toda uma interface de comunicação com o banco de dados Microsoft SQL Server, bem como implementação de métodos para consulta, persistência e alteração de dados.

O gráfico de consumo do pulmão foi desenvolvido utilizando a biblioteca gráfica “*System.Drawing”*, que já possui métodos prontos para desenhar a base do gráfico, traçar linhas e inserir pontos.

Para desenhar o gráfico, foi utilizada uma fórmula que calcula a proporção do tempo da atividade em cada ponto do eixo X. A área verde do gráfico contempla o tempo total da atividade, porém conforme o eixo X cresce, a área verde tem sua altura aumentada. Da forma com que a fórmula desenvolvida, este aumento na altura é considerado para que o gráfico mantenha sua proporção.

7.2.6 Resultados Obtidos

Através do estudo realizado, foi possível desenvolver um software que gerenciasse as atividades de um projeto. Com as atividades cadastradas no

sistema, é possível aplicar o conceito da Teoria da Corrente Crítica e realizar a análise dos resultados através de um gráfico.

O software desenvolvido permite uma simulação do que seria aplicar os conceitos da TOC em um projeto real, visualizando o consumo do pulmão do projeto através de um gráfico e podendo assim tomar decisões acerca do andamento de cada atividade.

É importante salientar que o gráfico não possui uma escala no eixo “Y” devido a forma com que o gráfico é apresentado. No eixo “Y”, os valores podem ser diferentes para o mesmo ponto em “Y”, pois ele varia com relação ao eixo “X”.

Outro ponto a ser considerado, é que a eficácia da teoria aplicada depende muito da maturidade da equipe envolvida, pois ao longo do projeto, ajustes devem ser feitos na forma com que a estimativa de tempo é realizada. Se praticamente todas as atividades do projeto estão na área verde do gráfico, é possível que a estimativa de tempo destas atividades esteja acima do que seria correto. Por outro lado, se praticamente todas as atividades estão na área vermelha do gráfico, pode ser um indício de que as estimativas de tempo estão baixas demais, então cabe ao gestor do projeto interpretar o gráfico da forma correta para conseguir extrair o máximo possível de sua equipe com a utilização das técnicas da Corrente Crítica.

8 CONCLUSÃO

A Teoria da Corrente Crítica traz novos paradigmas para os gerentes de projeto. Seus métodos diferenciados permitem uma nova forma de abordagem para as estimativas de tempo das atividades de um projeto, bem como otimizar o tempo e a produtividade dos recursos envolvidos.

O simples fato de se reduzir o tempo disponível para conclusão de uma atividade faz com que o recurso responsável trabalhe em um ritmo acelerado para comprimento dos prazos. Ainda que a meta não seja alcançada, existe um ganho com relação à estimativa inicial, e ao longo de um projeto com várias atividades, o ganho se torna considerável.

O software desenvolvido possibilitou uma visualização de forma mais clara da aplicação dos conceitos da Corrente Crítica sobre as atividades de um projeto, porém de forma limitada. Como sugestão para trabalhos futuros, o software desenvolvido pode ser aprimorado para que seu uso seja relevante de forma profissional.

REFERÊNCIAS

ANDERSON, David J.. **Kanban: Successful Evolutionary Change for Your Technology Business**. Blue Hole Press, 2010.

BECK, Kent. **Programação Extrema Explicada**. Bookman, 1999.

BOEHM, Barry, **A View of 20th and 21st Century Software Engineering**, ICSE, 2006.

GOLDRATT, Eliyahu M.. **What is This Thing Called Theory of Constraints and How Should It Be Implemented?**. North River Press, 1990.

GOLDRATT, Eliyahu M. **Corrente Crítica**. São Paulo: Nobel. 1998.

LADAS, Corey. **Scrumban - Essays on Kanban Systems for Lean Software Development**. Modus Cooperandi Press, 2009.

MOLINARI, Leonardo. **Gestão de Projetos: Técnicas e Práticas com Ênfase em Web**. São Paulo: Érica, 2004.

KRUCHTEN, Philippe. **The Rational Unified Process: An Introduction**. Addison-Wesley Professional, 2a edição, 2000.

PRESSMAN, Roger S.. **Engenharia de Software**. São Paulo: McGraw-Hill, 2006.

Project Management Institute. **PMBOK Guide**. PMI, 4ª edição, 2008.

Rational Unified Process, **Documentação do processo RUP**. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 2011/2.

SCHWABER, Ken, **Agile Project Management With Scrum**, Microsoft, 2004.

SOMMERVILLE, Ian, **Engenharia de Software**, Pearson Addison-Wesley, 2007.

BILIOGRAFIA COMPLEMENTAR

BALTAR, Marcelo S.; PEREIRA, Rodrigo S. C.. **Corrente Crítica em Gerenciamento de Projetos**. Disponível em: <http://www.techoje.com.br/site/techoje/categoria/detalhe_artigo/746>. Acesso em: 2011/2.

MENEZES, Klinger. **Corrente Crítica em Gerenciamento de Projetos**. Disponível em: <http://klingermenezes.wordpress.com/2007/07/06/corrente-critica-em-gerenciamento-de-projetos/>. Acesso em: 2011/2.

MARCANTONIO, Maria I. P. M.. **A Corrente Crítica Aplicada na Ferramenta de Gestão de Projetos MS Project**. Disponível em: <http://www.labceo.com.br/bibliografia/archive/files/a-5_51d04174b9.pdf>. Acesso em: 2011/2.

QUELHAS, Oswaldo; BARCAUI, André B. **A Teoria das Restrições aplicada a Gerência de Projetos: Uma Introdução à Corrente Crítica**. Disponível em: <http://www.bbbrothers.com.br/scripts/Artigos/Introdu%E7%E3o%20a%20Corrente%20Critica.pdf>. Acesso em: 2011/2.

ROUILLER, Ana Cristina. **Gerência de Projetos de Software**. Lavras: UFLA/FAEPE, 2008.

ZANATTA, Alexandre L.; VILAIN, Patrícia. **Uma análise do método ágil Scrum conforme abordagem nas áreas de processo Gerenciamento e Desenvolvimento de Requisitos do CMMI**. Disponível em: <http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER05/alexandre_zanatta.pdf>. Acesso em: 2011/2.

TOLEDO, Ricardo C. **Integração das metodologias RUP e PMI no desenvolvimento de projetos de software**. Disponível em: <www.apaebrasil.org.br/arquivo.phtml?a=10229>. Acesso em: 2011/2.

APÊNDICE(S)

APÊNDICE A – Artigo Científico

Abordagem de uma Ferramenta de Apoio à Utilização da Teoria da Corrente Crítica para o Gerenciamento de Projetos no Contexto da Melhoria do Processo de Software

Leandro Daminelli[1], Gustavo Bisognin[1]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – 88806-000– Criciúma – SC – Brasil

leandrodaminelli @hotmail.com, gbisog@gmail.com

Resumo. *O presente artigo tem como objetivo apresentar de forma resumida o trabalho de conclusão de curso desenvolvido por Leandro Daminelli, com orientação de M.Sc Gustavo Bisognin. O objetivo do trabalho de conclusão de curso foi estudar e aplicar os conceitos da Teoria da Corrente Crítica no Gerenciamento de Projetos, através do desenvolvimento de um software.*

Palavras – chave: *Gerenciamento de Projetos, Teoria da Corrente Crítica, Teoria das Restrições.*

1. Introdução

Com a constante busca por maior produtividade e menores custos e prazos, a Teoria da Corrente Crítica, escrita por Eliyahu M. Goldratt, foi criada em 1997, trazendo novos conceitos para o gerenciamento de projetos e suas respectivas atividades.

A Teoria da Corrente Crítica traz uma forma diferente de tratar o gerenciamento de projetos. Baseada na Teoria das Restrições, a Corrente Crítica foca nas atividades do projeto e visa aplicar uma metodologia agressiva nas estimativas de tempo, diminuindo drasticamente o tempo estimado para realização de uma atividade do projeto.

A Teoria das Restrições (TOC), também criada por Eliyahu M. Goldratt, visa analisar e tratar as restrições de um projeto, seja esta uma restrição interna, externa ou de mercado, objetivando o aumento do desempenho das atividades do projeto. A TOC pode ser aplicada em diversas áreas de negócio, incluindo a Engenharia de Software.

Dentro dos conceitos abordados pela corrente crítica, estão a Lei de Parkinson e a Síndrome do Estudante, utilizadas como base para justificar os atrasos em projetos, que ocorrem mesmo com as grandes margens de segurança aplicadas sobre as atividades deste projeto.

A Lei de Parkinson, escrita por Cyril Northcote Parkinson no livro The Economist, parte do princípio de que quanto mais tempo livre uma pessoa tem, mais tempo ela vai levar para realizar suas atividades. Contextualizando, se um empregado tem um prazo para entregar o seu trabalho, mesmo que ele tenha terminado este trabalho antes do prazo final, o tempo restante é utilizado para "finalizar completamente" a atividade.

A Síndrome do Estudante é conhecida por todos, porém nem todos sabem que esta síndrome existe. Quando temos um prazo estipulado para desenvolver uma atividade, algumas pessoas acabam deixando para realizar o trabalho na "última hora", estando passíveis de situações inesperadas que resultam em atrasos no projeto.

Nas metodologias comuns de gerenciamento de projetos, são adicionados "pulmões" de segurança na estimativa de tempo de cada atividade do projeto. Assim, o prazo final de conclusão de um projeto chega a ser até 40% maior que o prazo inicial. Ainda com a margem de segurança aplicada, a maioria dos projetos costuma atrasar graças ao comportamento do ser humano em relação à Lei de Parkinson e a Síndrome do Estudante.

Os constantes atrasos em projetos têm como conseqüência a menor competitividade no mercado, atrito com os clientes, maiores custos, entre outros, e acabam sendo um grande problema para uma empresa que possui muitos projetos e utilizam técnicas antigas de gerenciamento de atividades do projeto.

As formas de gerenciamento e controle tradicionais, abordados pela maioria das empresas desenvolvedoras de soluções de software, não conseguem obter um gerenciamento preciso e efetivo quanto a prazos, configurando um grave problema de custo e desgaste contínuo com o cliente.

Estudos recentes divulgados pelo Standish Group comprovam que mais de 80% dos projetos de software atrasam, causando um prejuízo considerável para as organizações. De acordo com este estudo, a forma de gerenciamento abordada na maioria das vezes é a tradicional, com gerenciamento de atividades e acompanhamento de cronograma, não havendo nenhuma técnica mais precisa para prever os desvios e controlar os riscos envolvidos nesta atividade.

Com base no problema supracitado, este trabalho tem como objetivo desenvolver uma aplicação que implementa a Teoria da Corrente Crítica aplicada no gerenciamento de projetos, abordando o contexto da melhoria contínua do processo de desenvolvimento de software.

2. Gerenciamento de Projetos

A Gerência de Projetos é utilizada em larga escala por empresas do mais variado tipo de negócio. Seus benefícios foram sendo comprovados ao longo do tempo, trazendo como principal resultado a redução de custos, que aumenta consideravelmente de acordo com o tamanho dos projetos.

Segundo Molinari (2004), a aplicação da disciplina de Gerenciamento de Projetos atualmente inclui diversas áreas, como: indústrias, construção civil, sistemas de informação, serviços financeiros, educação, treinamentos entre outros. Acredita-se que o domínio e aplicação das boas práticas estabelecidas nesta disciplina, minimizam consideravelmente a perda de produtividade e o custo envolvido no retrabalho das atividades das mais diversas áreas do domínio do conhecimento. Contudo, a aplicação efetiva dos conceitos estabelecidos no gerenciamento de projetos, não é trivial de ser executado, envolvendo um alto investimento e um grande *know how* dos recursos humanos executores.

Segundo Sommerville (2007), gerência de projetos é a união, organização e aplicação de técnicas e conhecimento para realização de atividades que visam o alcance de um objetivo, controlando as variáveis de risco, prazo, custos e pessoas envolvidas. Todas estas variáveis são de extrema importância para o sucesso pleno do projeto, sendo assim, os gerentes de projeto envolvidos neste contexto devem possuir vasta experiência na área a fim de multiplicar as chances de sucesso do projeto.

Segundo o PMBOK (2008), o ato de gerir um projeto consiste em definir uma série de tópicos ou áreas de conhecimento acerca do objetivo que se deseja alcançar ao final deste.

Estes tópicos são: Escopo, Tempo, Custos, Riscos, Qualidade, Recursos humanos, Comunicações, Aquisições e Integração.

j) escopo: O escopo descreve o projeto em si, de forma a definir seus requisitos, objetivos e meta a ser alcançada, e deve ser controlado durante todo o ciclo de vida do projeto. Os requisitos são as necessidades que o cliente possui, como por exemplo, a necessidade de desenvolvimento de uma nova rotina para o software ou um novo relatório. O cliente não é necessariamente externo, podendo este ser também um setor da própria organização. Os objetivos são os resultados gerados ao fim do projeto, que irão suprir as necessidades do cliente, sendo que este resultado deve ser entregue dentro da metas estipuladas para o projeto.
k) tempo: O projeto é dividido em pequenas partes que podemos chamar de atividades, e através de técnicas de estimativa de tempo, é realizada a estimativa para cada atividade a fim de definir o cronograma do projeto. As atividades do projeto são definidas pelos gestores do projeto em conjunto com a equipe envolvida, e cabe ao gerente de projeto definir o quão detalhado será o desdobramento das atividades, por exemplo: em um tema comum do nosso cotidiano que é lavar roupas, o gerente de projetos poderia dividir as atividades de forma mais simplificada, como Lavar, Secar, Passar e Guardar, ou então detalhar estas atividades a fim de se alcançar um maior controle em cada passo, como Colocar a roupa na máquina, Lavar, Estender a roupa, Esperar a roupa secar, Passar a roupa, Dobrar a roupa e Guardar a roupa.
l) custos: O custo do projeto deve ser entendido também como valor de investimento. O custo com Recursos Humanos, Aquisições e demais gastos necessários devem ser avaliados para análise do custo-benefício do projeto. Neste tópico, a precisão do gerente de projetos e também da equipe envolvida deve ser alta, pois o orçamento para o projeto é de suma importância para sua aprovação com relação ao seu custo-benefício. Um projeto que não teve o gerenciamento correto de custos pode ser abortado no meio do caminho por falta de recursos, o que é uma falha grave para qualquer organização, seja ela pequena ou grande. Deve ser considerada também a confiança da gerência de alto nível na equipe envolvida no projeto.
m) riscos: Na definição dos riscos devem ser abordados todos os eventos indesejados que possam ocorrer ao decorrer do projeto, desde mudanças na estrutura da empresa até problemas com colaboradores, visando estar preparado para agir na ocorrência do caso. Este é outro ponto crítico em qualquer projeto, pois riscos que não foram cogitados no momento da elaboração deste podem vir a mudar o cenário e causar até mesmo o fim prematuro do projeto. Qualquer risco por menor que este seja, deve ser considerado e deve possuir um plano de ação definido para contorná-lo.
n) qualidade: Segundo Molinari (2004), a qualidade do projeto visa garantir que o resultado seja realmente alcançado com base no que foi solicitado, com eficiência e alta produtividade. Não basta apenas cumprir o cronograma do projeto, mas para isto deixar de lado outros tópicos também importantes no decorrer das atividades. Entregar o resultado na data prometida, porém entregando um resultado destorcido com relação ao que foi definido no escopo do projeto pode ser muito desgastante no relacionamento entre empresa x cliente, por isto o gerente de projetos deve sempre manter o equilíbrio entre todos os tópicos do projeto, mantendo assim um alto índice de qualidade geral, e não focando apenas em tempo ou custos.

o) recursos humanos: São as pessoas elencadas para desenvolver as atividades a fim de realizar o objetivo do projeto. A equipe definida para desenvolver o projeto é muito importante, pois neste ponto existe impacto direto em outros tópicos como Tempo, Qualidade e Custos. Um colaborador mais experiente pode realizar a mesma atividade com mais qualidade e em menos tempo, porém com um custo maior. Um colaborador que acabou de entrar na organização pode custar menos, porém demorar mais para realizar a atividade e não realizar da forma correta (menor qualidade). Neste ponto o gerente de projetos deve alcançar um ponto de equilíbrio. Uma das formas de alcançar este ponto é envolvendo pessoas experientes e não experientes no projeto, sendo estas últimas sempre supervisionadas, garantindo assim um equilíbrio entre custos, tempo e qualidade, e também contribuindo para o desenvolvimento de colaborados menos experientes.
p) comunicações: Indispensável para o sucesso do projeto, a comunicação entre os participantes deve ser clara e objetiva para que problemas como a má interpretação dos requisitos não ocorra. Reuniões diárias de acompanhamento e comunicação oral, evitando mensagens eletrônicas, pode ajudar muito na interação entre a equipe, o que favorece até mesmo a criar um laço de amizade entre os participantes, aumentando assim a idéia de time entre os colaboradores da empresa. Uma abordagem maior sobre técnicas de comunicação será descrita no capitulo 3.
q) aquisições: Aquisição de produtos e serviços necessários para a execução do projeto. Este tópico também impacta diretamente no Custo do projeto. A necessidade de aquisição de materiais fora do planejado pode trazer dor de cabeça aos gestores, e a definição correta deste tópico é muito importante para avaliação do custo-benefício do projeto. Dependendo do escopo do projeto, este tópico pode definir a viabilidade de execução do projeto.
r) integração: União e interação de todas as áreas de conhecimento realizada no dia-a-dia pelo gerente de projetos, a fim de manter o equilíbrio na definição e acompanhamento de todos os tópicos do projeto.

Os tópicos supracitados são relacionados de tal forma que, caso ocorra alguma alteração em um item no decorrer do projeto, pelo menos algum outro tópico também sofrerá alterações. Como exemplo, em um projeto já definido, caso ocorram alterações no escopo do projeto, o cronograma terá que ser alterado a fim de se adequar ao novo escopo, bem como o custo será alterado por conta do novo cronograma e assim por diante.

Através dos projetos, as empresas buscam a organização, redução de custos, controle de riscos e redução de prazos, de forma a garantir cada vez mais a qualidade do seu negócio, bem como sua rentabilidade.

Os projetos são concebidos geralmente a partir do planejamento estratégico de uma organização, com o objetivo de atender uma solicitação dos clientes, sejam estes internos ou externos à organização, atender a demanda de mercado, troca de tecnologia utilizada, mudança de local físico da organização ou de determinado setor, treinamento da equipe envolvida, entre outros.

O gerenciamento de projetos visa, através de várias técnicas e conhecimento acumulado, organizar a execução de um projeto do início ao fim, prevendo custos, riscos, alocando recursos, garantindo a qualidade e entregando o resultado no prazo definido de acordo com o escopo do projeto.

O inicio do gerenciamento de projetos se deu antes mesmo do nascimento de Cristo. Grandes construções como as pirâmides do Egito exigiram alguma forma de organização para que pudessem ser concluídas.

Apesar de alguns conceitos já serem aplicados há muito tempo atrás, foi na década de 50 que se iniciou a nova era do gerenciamento de projetos. Nesta época, o gráfico de Gantt, criado por Henry Gantt, era amplamente utilizado e sobrevive até hoje nas modernas ferramentas de controle de projetos.

Em 1969, o Project Management Institute (PMI) foi fundado com o objetivo de definir padrões de gerenciamento de projetos e hoje conta com mais de 150 mil membros em todo mundo. Em 1996, lançaram o guia PMBOK, que formaliza diversos conceitos no gerenciamento de projetos e um conjunto de boas práticas, aplicáveis à maioria dos projetos na maior parte do tempo.

O PMI também certifica profissionais da área como Gerente de Projetos Profissional através da certificação Project Management Professional (PMP). No Brasil, as estatísticas revelam que há apenas aproximadamente 400 profissionais certificados nesta área.

Pesquisas realizadas pela consultoria *The Standish Group* em 1994 e 2001 comprovam a rápida evolução do gerenciamento de projetos. A pesquisa realizada em 2001 aponta uma redução de 144% na extrapolação de orçamento, redução de 159% na extrapolação do prazo e aumento de 12% na entrega de resultados dentro do tempo, custo e especificação prevista. Estes dados comprovam a eficácia e os benefícios do gerenciamento de projetos dentro de uma organização.

3. Teoria das Restrições

A Teoria das Restrições (TOC) teve seu início na década de 70, quando o físico Israelense Eliyahu Goldratt se envolveu com problemas de logística de produção de uma organização.

Goldratt elaborou então um método de administração da produção totalmente novo. O método elaborado foi muito bem visto pelo mercado e outras empresas se interessaram em aprender sua técnica. A partir daí, Goldratt se dedicou a continuar o seu trabalho e a disseminá-lo.

Nos dias de hoje, a TOC é composta por dois campos: os Processos de Raciocínio e os Aplicativos Específicos (como logística de produção). Os processos de raciocínio da TOC foram além da área de Administração e hoje são utilizados em muitas outras áreas de conhecimento, formando a base de toda a teoria. Esta teoria é hoje amplamente utilizada nas áreas de manufatura, logística e distribuição, cadeia de suprimentos, gerenciamento de projetos, marketing e vendas, contabilidade, prestação de serviços, tecnologia da informação, engenharia de software, saúde, educação, entre outras.

A Teoria das Restrições contribuiu muito para o mundo da administração através do seu processo de melhoria contínua. Este processo é definido por cinco etapas (GOLDRATT, 1990):

f) identificar as restrições do sistema: da mesma maneira que em uma corrente sempre existe um elo mais fraco que pode se partir, em uma organização sempre existirá um recurso que limitará a sua capacidade. Para buscar a otimização da produção, a organização deve identificar o “elo mais fraco” para então tomar uma ação na próxima etapa.

g) decidir como a restrição será tratada: com as restrições já identificadas, a organização deve otimizar ao máximo esta restrição a fim de maximizar o fluxo neste ponto.

h) subordinar os recursos ao elo fraco: todos os outros recursos da organização devem seguir no mesmo ritmo da restrição identificada, nem mais rápido e nem mais devagar. Trabalho a menos resultaria em diminuição no ritmo da produção, ao passo que trabalho a mais resultaria em esforço extra à toa, já que existe um recurso limitando a produção.

i) elevar a restrição do sistema: após extrair o máximo da restrição no segundo passo, neste momento a organização deve considerar várias alternativas para investir mais nas restrições a fim de quebrá-las. Neste momento, outras restrições são identificadas e então o processo volta a etapa um.

j) não permitir que a inércia cause uma restrição: a partir do momento em que uma restrição é quebrada, a organização deve continuar em busca da melhoria contínua. Toda a organização possui pelo menos uma restrição, e ela deve estar sempre em foco para que a organização nunca pare de se desenvolver.

As restrições de uma organização não são necessariamente ruins, algumas apenas existem. Se uma organização não possuísse nenhuma restrição, seu desempenho seria teoricamente infinito. De acordo com os pressupostos da Teoria das Restrições, restrição é qualquer coisa que limita uma organização em conseguir maior desempenho no seu fluxo de trabalho. Pode-se afirmar que toda organização possui pelo menos uma restrição ou que toda corrente possui sempre um elo mais fraco.

Na TOC, os processos são descritos através da utilização de três perguntas fundamentais: “O que mudar?”, “Mudar para o que?” e “Como causar a mudança?”. Estas perguntas fornecem a estrutura para os processos de raciocínio da TOC. Os processos de raciocínio são um conjunto de ferramentas e processos que permite que um indivíduo ou grupo resolva um problema ou desenvolva uma estratégia integrada com o rigor e a lógica de causa e efeito, buscando inicialmente os sintomas e terminando com um plano de ação detalhado que coordene as atividades de todos os envolvidos na implementação da solução para a restrição.

4. Teoria da Corrente Crítica

Criada por Eliyahu M. Goldratt, a Corrente Crítica se inicia a partir da observação básica de que os problemas comuns a todos os projetos são a alta probabilidade de extrapolação de orçamento, extrapolação de tempo, e comprometimento do conteúdo. A Corrente Crítica foi desenvolvida para ser implantada como um projeto estratégico de gestão a fim de evitar atrasos no causados pela Lei de Parkinson e Síndrome do Estudante, abordados em seqüência. O objetivo é tentar minimizar o impacto da Lei de Parkinson e Síndrome do Estudante através da construção do cronograma com duração das estimativas baseadas em um nível de confiança de 50%.

A idéia principal desta metodologia, segundo Goldratt (1998), é realizar uma estimativa de tempo das atividades do projeto da forma com que a equipe achar melhor. A partir daí, estas estimativas devem ser cortadas pela metade e este é o tempo que a equipe envolvida terá para realizar as tarefas. A diferença entre os tempos (estimativa inicial e estimativa 50%) é colocada ao final do projeto como um pulmão de segurança.

Inicialmente o uso desta técnica pode indicar muitos atrasos devido ao fato de a equipe não conhecer a sua real capacidade de produção. As estimativas iniciais podem não ser ideais, e ao aplicar a taxa de 50% sobre estas estimativas, pode dificultar ainda mais o cumprimento do cronograma. Porém, esta forma agressiva de estimar o tempo consegue minimizar ao máximo a incidência da Lei de Parkinson e da Síndrome do Estudante, diminuindo o tempo efetivo de duração do projeto.

5. Síndrome do Estudante

A Síndrome do Estudante pode ser melhor entendida quando associada ao comportamento dos alunos em seu período letivo. A maioria dos alunos realiza seu trabalho sempre na última hora possível, e este comportamento também é observado dentro de organizações.

Quando várias tarefas são delegadas a uma pessoa, esta acaba realizando-as sempre no último momento. Quando ocorrem acontecimentos imprevistos, a entrega da tarefa acaba atrasando trazendo prejuízo para a organização.

6. Lei de Parkinson

A Lei de Parkinson também é muito comum dentro de organizações do mais variado tipo de negócio. Esta lei afirma que quando uma tarefa tem o seu prazo maior do que o necessário para sua execução, a pessoa responsável ao invés de manter o ritmo de trabalho e terminar a tarefa antes do prazo, acaba diminuindo o seu ritmo para finalizar a tarefa no prazo em que lhe foi concedido.

Este comportamento é muito comum e também muito prejudicial a qualquer organização, pois apesar da pessoa cumprir o prazo de sua atividade, este foi superestimado. Como tudo ocorre dentro do prazo, é muito provável que as próximas estimativas sigam a mesma metodologia, fazendo com que a equipe trabalhe sempre abaixo da sua capacidade máxima de produção.

7. Aplicação Desenvolvida

Primeiramente foi realizado o levantamento de material bibliográfico para estudo através de pesquisas na biblioteca da UNESC, na Biblioteca Municipal de Criciúma e em artigos / livros encontrados na internet.

Após levantar o material necessário, foram estudadas metodologias de gerenciamento de projetos, assim como a própria disciplina de gerenciamento de projetos para obter um melhor entendimento do assunto antes de iniciar o estudo do tema deste trabalho que é a Teoria da Corrente Crítica. Em seqüência foi realizado o estudo da Teoria das Restrições e Teoria da Corrente Crítica para dar início a escrita do documento.

Finalizado o estudo, foi iniciado o desenvolvimento da ferramenta, contemplando o conhecimento obtido através dos estudos realizados.

Para melhor organização do desenvolvimento da aplicação, foi realizado o levantamento dos requisitos de software, modelagem destes requisitos, modelagem do banco de dados, modelagem da interface, e por fim a codificação na linguagem C#.

O conceito Model View Control (MVC) – Modelo Visão Controle – foi utilizado parcialmente na codificação do software. Neste modelo, o projeto é dividido em três camadas de desenvolvimento independentes. No caso do projeto desenvolvido, este conceito não foi seguido completamente, porém boa parte das melhores práticas de projeto de software foram utilizadas a fim de facilitar a manutenção do código.

8. Resultados Obtidos

Através do estudo realizado, foi possível desenvolver um software que gerenciasse as atividades de um projeto. Com as atividades cadastradas no sistema, é possível aplicar o conceito da Teoria da Corrente Crítica e realizar a análise dos resultados através de um gráfico.

O software desenvolvido permite uma simulação do que seria aplicar os conceitos da TOC em um projeto real, visualizando o consumo do pulmão do projeto através de um gráfico e podendo assim tomar decisões acerca do andamento de cada atividade.

É importante salientar que o gráfico não possui uma escala no eixo “Y” devido a forma com que o gráfico é apresentado. No eixo “Y”, os valores podem ser diferentes para o mesmo ponto em “Y”, pois ele varia com relação ao eixo “X”.

Outro ponto a ser considerado, é que a eficácia da teoria aplicada depende muito da maturidade da equipe envolvida, pois ao longo do projeto, ajustes devem ser feitos na forma com que a estimativa de tempo é realizada. Se praticamente todas as atividades do projeto estão na área verde do gráfico, é possível que a estimativa de tempo destas atividades esteja acima do que seria correto. Por outro lado, se praticamente todas as atividades estão na área vermelha do gráfico, pode ser um indício de que as estimativas de tempo estão baixas demais, então cabe ao gestor do projeto interpretar o gráfico da forma correta para conseguir extrair o máximo possível de sua equipe com a utilização das técnicas da Corrente Crítica.

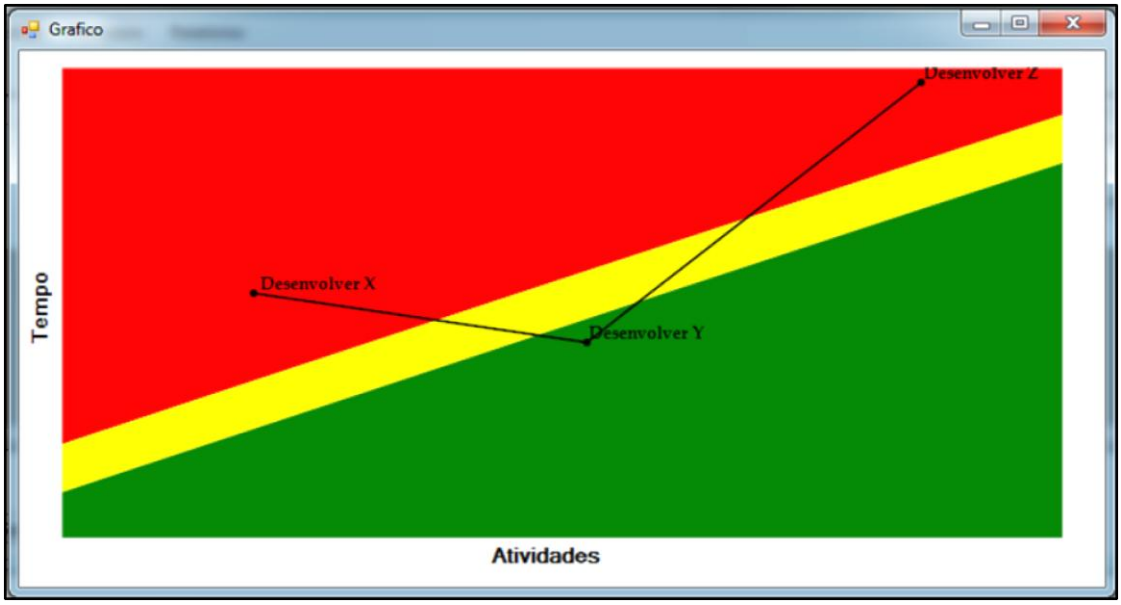


Figura 1 – Gráfico de Consumo do Pulmão do Projeto

9. Conclusão

A Teoria da Corrente Crítica traz novos paradigmas para os gerentes de projeto. Seus métodos diferenciados permitem uma nova forma de abordagem para as estimativas de tempo das atividades de um projeto, bem como otimizar o tempo e a produtividade dos recursos envolvidos.

O simples fato de se reduzir o tempo disponível para conclusão de uma atividade faz com que o recurso responsável trabalhe em um ritmo acelerado para comprimento dos prazos. Ainda que a meta não seja alcançada, existe um ganho com relação à estimativa inicial, e ao longo de um projeto com várias atividades, o ganho se torna considerável.

O software desenvolvido possibilitou uma visualização de forma mais clara da aplicação dos conceitos da Corrente Crítica sobre as atividades de um projeto, porém de forma limitada. Como sugestão para trabalhos futuros, o software desenvolvido pode ser aprimorado para que seu uso seja relevante de forma profissional.

Referências

ANDERSON, David J.. **Kanban: Successful Evolutionary Change for Your Technology Business**. Blue Hole Press, 2010.

BECK, Kent. **Programação Extrema Explicada**. Bookman, 1999.

BOEHM, Barry, **A View of 20th and 21st Century Software Engineering**, ICSE, 2006.
GOLDRATT, Eliyahu M.. **What is This Thing Called Theory of Constraints and How Should It Be Implemented?**. North River Press, 1990.

GOLDRATT, Eliyahu M. **Corrente Crítica**. São Paulo: Nobel. 1998.
LADAS, Corey. **Scrumban - Essays on Kanban Systems for Lean Software Development**. Modus Cooperandi Press, 2009.

MOLINARI, Leonardo. **Gestão de Projetos: Técnicas e Práticas com Ênfase em Web**. São Paulo: Érica, 2004.

KRUCHTEN, Philippe. **The Rational Unified Process: An Introduction**. Addison-Wesley Professional, 2a edição, 2000.

PRESSMAN, Roger S.. **Engenharia de Software**. São Paulo: McGraw-Hill, 2006.
Project Management Institute. **PMBOK Guide**. PMI, 4ª edição, 2008.

Rational Unified Process, **Documentação do processo RUP**. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 2011/2.

SCHWABER, Ken, **Agile Project Management With Scrum**, Microsoft, 2004.
SOMMERVILLE, Ian, **Engenharia de Software**, Pearson Addison-Wesley, 2007.