UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDSON DE OLIVEIRA BRANCO

TEORIA DA CORRENTE CRITICA PARA MINIMIZAR O PROBLEMA DE GERENCIAMENTO DE PROJETOS DE SOFTWARES

CRICIÚMA

2017

EDSON DE OLIVEIRA BRANCO

TEORIA DA CORRENTE CRITICA PARA MINIMIZAR O PROBLEMA DE GERENCIAMENTO DE PROJETOS DE SOFTWARES

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

Orientador: Prof. MSc. Gustavo Bisognin

CRICIÚMA

2017

EDSON DE OLIVEIRA BRANCO

Teoria da Corrente Crítica para Minimizar o Problema de Gerenciamento de Projetos 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, 22 de junho de 2017.

BANCA EXAMINADORA

Prof. Gustavo Bisognin– MSc. - (UNESC) - Orientador

Prof. Ana Claudia Garcia Barbosa –MSc. - (UNESC)

Prof. Paracelso de Oliveira Caldas – MSc. - (UNESC)

“Dedico primeiramente a Deus pelas bênçãos ao longo dessa caminhada, aos meus pais por estarem sempre do meu lado em todos os momentos, me motivando e amparando nos bons e maus momentos”.

“No fim tudo dá certo, se não deu certo é porque ainda não chegou ao fim”.

Fernando Sabino

RESUMO

A teoria da corrente crítica originou-se por Eliyahu Goldratt em 1997, com a evolução dos negócios e das empresas vindo de um mundo globalizado Goldratt observou três problemas para gestão de projetos, a lei de Parkinson, a Síndrome de Estudante e a Multitarefa. Assim a certa a adaptação de conhecimento, habilidades e técnicas facilita a melhor visão dos resultados a serem entregues dentro dos critérios de tempo, orçamento, escopo e qualidade. A teoria da corrente critica aborda uma forma diferente de tratar o gerenciamento de projeto no contexto de software. A teoria das restrições busca a otimização dos desempenhos das atividades do projeto; os modelos de processos de software tende a tornar os projetos mais gerenciáveis, CMMI e MPS.BR modelos de qualidade; atualmente destaca-se os modelos de gerenciamento de projetos, o modelo Scrum, o modelo *Xtreme programming* (XP) e modelos tradicionais como o RUP e PMBOK. De modo geral a corrente crítica é um modelo para gerenciamento de projetos que seu foco é basicamente na administração de prazos e atividades, considerando alocação de recursos baseado na TOC. Nesse contexto a presente pesquisa busca apresentar uma melhor técnica de gerenciamento do tempo em projetos considerada como um dos mais inovadores avanços na área de gerenciamento de projetos; por se tratar de uma pesquisa aplicada em um projeto real, onde foram analisados somente os dados pertinentes para a conclusão da pesquisa, assim a proposta contempla a fase de início, planejamento, execução, controle e encerramento.

Palavras chave: Gerenciamento de projetos. Corrente critica. Teoria das Restrições.

ABASTRACT

The theory of the critical current, originated by Eliyahu Goldratt in 1997, with the evolution of the business and of the companies coming from a globalized world Goldratt noted three problems for management of projects, Parkinson's law, the Syndrome of the Student and the Multitasking. Thus, some adaptation of the knowledge, skills, and techniques facilitates a better vision of the results to be delivered within the criteria of time, budget, scope and quality. The theory of the current critique addresses a different way to treat the project management in the context of software. The theory of constraints seeks the optimization of the performance of the activities of the project; models of software processes tends to make the projects more manageable, CMMI and MPS.BR quality models; currently we highlight the models of project management, the Scrum model, the model Xtreme programming (XP) and traditional models like the RUP and the PMBOK. Generally, the current critique is a model for project management that your focus is primarily on the administration of deadlines and activities, considering resource allocation based on the TOC. In this context, the present research seeks to present a better technique of time management in projects is considered as one of the most innovative advances in the area of project management; for being an applied research in a real project, where they were analyzed only the data relevant to the completion of the search, so the proposal includes the stage of initiation, planning, execution, control and closing.

Keywords: Project management. Critical current. Theory of restrictions.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMMI	*Capability Maturity Model Integration*
SEI	*Software Engineering Institute*
PMMM	*Project Management Maturity Model*
MPS.BR	Melhoria de Processo do Software Brasileiro
PMBOK	*Project Management Body of Knowledge*
RUP	*Rational Unified Process*
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
TOC	Teoria das Restrições
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Com base em estudos históricos, identificou-se que a teoria da corrente crítica nasceu nos anos de 1997, tal técnica, apoia a constante busca por produtividade associada a uma redução nos custos de gerenciamento de projetos de engenharia de software. Esta técnica, foi escrita por Eliyahu M. Goldratt, e é responsável pela criação de novos conceitos aplicados ao gerenciamento de projetos e suas respectivas atividades e tarefas.

O gerenciamento de projeto existe a alguns anos, e vem sendo exercido em vários setores, no qual, práticas têm sido divulgadas e documentadas a um período de tempo, contribuindo assim para evolução dos projetos uma vez que abordam conceitos claros e ágeis em seu gerenciamento (CLELAND; IRELAND, 2007). Os projetos possuem distintas dimensões a serem gerenciadas, a organização é uma das peças importante para desenvolvimento de produtos, contudo, o gerenciamento de projetos busca a redução de custo, prazo e entregas para o cliente final (SILVA; RODRIGUES; LACERDA, 2012).

A Teoria das Restrições visa o tratamento das restrições de um projeto, sejam elas internas ou externas ou de mercado. Tal teoria busca a otimização do desempenho das atividades do projeto, aumentando a probabilidade de entrega no prazo definido. Neste contexto, identifica-se que a teoria das restrições pode ser aplicada em diversas áreas de conhecimento, como projetos de engenharia civil, mecânica, elétrica e principalmente na engenharia de software.

No entanto a teoria da corrente crítica aborda uma forma diferenciada de tratar o gerenciamento de projetos no contexto de software. Fortemente baseada na teoria das restrições, a corrente crítica enfatiza o gerenciamento das atividades do projeto e objetiva a aplicação de metodologia agressiva no que se refere as estimativas de tempo, diminuindo consideravelmente o tempo estimado para a realização de uma determinada atividade do projeto.

A teoria da corrente crítica aborda basicamente três conceitos, sendo um deles denominado de síndrome do estudante e o outro de teoria de Parkinson os quais servem como base para a justificativa dos atrasos nos projetos de desenvolvimentos de software, os quais ocorrem mesmo com a utilização de grandes margens de tempo no planejamento das atividades.

A Lei de Parkinson, escrita por Cyril Northcote Parkinson no livro The Economist, consiste no principio que as atividades tendem a se expandir ocupando o tempo disponível, as pessoas estimam as durações protegendo-se de cortes.

A Síndrome de Estudante destaca o comportamento das pessoas em esperar até o último momento para iniciar as atividades, estimativas de duração são baseadas em experiências pessimistas.

De acordo com as metodologias usadas em gerenciamento de projetos, está relacionado com o conceito de reserva, oferecendo assim proteção contra atrasos ou não cumprimento das metas do projeto, sejam elas de tempos ou recursos.

Para Tukel, Rom e Eksioglu (2006), Goldratt adicionou este conceito de pulmão de projeto para proteger a corrente crítica e tornar executável o cumprimento do prazo do projeto como planejado.

O buffer (pulmão) também chamado de reserva funciona como um mecanismo de contingencia que consome ou diminui o efeito da variabilidade provocada pela incerteza inerente à programação de tempo, atividades e recursos em um projeto (ORDOÑEZ, 2013).

Deste modo o gerenciamento de projetos é executado nas seguintes fazes do ciclo de vida de projetos: iniciação, panejamento, execução, monitoramento e controle, e encerramento, os gerentes de projetos são responsáveis pela realização dos objetivos do projeto, o que inclui a aplicação de ferramentas, técnicas e habilidades para que o mesmo seja concluído com muito sucesso (AZANHA et al, 2014).

O gerenciamento de projeto enfrena alguns processos fundamentais, por este motivo outros métodos que buscam solucionar dificuldades de gerenciamento foram criados como métodos alternativos a Corrente Critica que

se fundamenta na teoria das restrições, consiste em reconhecer a duração de um projeto. A corrente crítica é uma das mais importantes áreas no desenvolvimento do gerenciamento de projeto (SILVA; RODRIGUES; LACERDA, 2012).

A satisfação do cliente está ligada diretamente aos benefícios pertencentes de um projeto, a corrente crítica é aplicada como método de abordagem gerencial e de diagrama de rede levando uma melhora significativa aos projetos procurando resolver os principais conflitos, buscando respostas lógicas e sistemáticas de qualquer processo de melhoria contínua (QUELHAS; BARCAUI, 2010).

Goldratt (1998), propõe que os aspectos do comportamento humano podem influenciar no andamento de um projeto. A corrente crítica da mesma forma que a teoria das restrições busca alcançar melhores resultados e para isto propõe mudanças em algumas premissas utilizadas tradicionalmente no gerenciamento de projetos (MARCANTONIO, 2010).

1.1 OBJETIVO GERAL

Propor um modelo utilizando o método da teoria da Corrente Critica para minimizar a dificuldade de gerenciamento de projetos de softwares.

1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo do trabalho foram necessárias as seguintes etapas:

a) compreender os métodos utilizados para estimativa de tempo das atividades;
b) identificar as principais causas de atrasos nas atividades de um projeto em relação ao seu cronograma inicial;
c) compreender a importância de Gerenciamento de projetos para obtenção de resultados na empresa Contare Tecnologia;
d) aplicar o método da Corrente Crítica em Gerenciamento de projetos;
e) analisar a melhoria com a teoria da corrente crítica;

f) analisar as áreas de gerenciamento de projetos, softwares para gerência de projetos;

g) criar um simulador para acompanhamento das atividades usando a corrente crítica.

1.3 JUSTIFICATIVA

De acordo com estudos realizados na área de engenharia de software, identificamos que a área de processo de gerenciamento de projetos, é fundamentada em três pilares. São eles: tempo, custo e prazo, sendo que qualquer alteração em um deles irá refletir diretamente nas outras, causando impactos no projeto como um todo. A arte de prever e gerenciar estes impactos torna o domínio desta área de processo, fundamental para qualquer organização.

A teoria da corrente critica é uma vertente administrativa que visa o bom gerenciamento de um projeto, fundamentada na teoria de restrições, ela tem grande impacto no reconhecimento dos recursos associados ao projeto, tornando assim o método importante, principalmente no que se refere a ambientes multi-projetos. O principal problema no gerenciamento de projetos são as incertezas, que originam dificuldades e obstáculos. Sendo assim é importante ressaltar que o não gerenciamento dos fenômenos da lei de Parkinson bem como a síndrome do estudante e multitarefas, podem acarretar no atraso do projeto (SILVA; RODRIGUES; LACERDA, 2012).

A corrente critica propõe que haja uma estimativa brusca na redução das atividades fazendo com que o tempo corresponda as espectativas das atividades e que reduza em 50% em relação ao habitual. As empresas estão concentradas cada vez mais em otimizar processo, aumentar produtividade, e cortar custo, para manterem-se em competitividade, o desempenho das empresas depende dos esforços de um conjunto de todos os elementos, tornando-o forte quanto o seu elo mais fraco. (MARCOANTONIO, 2010). A teoria da correte critica será aplicada através de ummodelo para o método da teoria da corrente crítica para minimizar a

dificuldade de gerenciamento de projetos de softwares, que gerenciará o cronograma de um projeto. Sendo assim, os gerentes de projetos utilizam formas tradicionais para estimativa de tempo das atividades, e esta proposta foca em um modelo para o método da teoria da corrente crítica para minimizar o problema de gerenciamento de projetos de softwares, utilizando assim as diretrizes da teoria da corrente crítica para realizar estimativas de tempo hostil, com intenção incentivar a equipe envolvida aumentando assim a produtividade, diminuindo assim custos e prazo do projeto.

1.4 ESTRUTURA DO TRABALHO

No primeiro capítulo há uma breve introdução do tema, formulação do problema, objetivos e justificativas.

Em seguida, no segundo capitulo, destaca-se a fundamentação teórica realizada pela autoria para realização do estudo, onde será abordada os modelos de processos de software e seu conceito, o modelo CMMI e modelo mps.br para melhor entendimento.

No terceiro capitulo abordou-se sobre a contextualização do gerenciamento de projetos, e o ciclo de vida de um projeto.

Em seguida, no capitulo quatro (4) enfatizou-se sobre os modelos de gerenciamento de projetos dentre os modelos destacou-se os modelos Ágeis e os modelos (SCRUM E XP) e outros modelos tradicionais como (RUP, PMBOK) abordadas no capitulo cinco (5).

Em sequência vem a teoria das restrições onde é abordada e contextualizada para o gerenciamento de projetos. Logo após, a teoria da corrente crítica, lei de Parkinson, Multitarefa e Síndrome de Estudante.

2 MODELOS DE PROCESSO DE SOFTWARE

De modo que as ações de melhorias possam ser desligadas, as empresas que precisam melhorar seus processos precisam adotar-se de normas e modelos capazes de arriscar possíveis padronizações afim de conseguir melhores resultados que leva a uma melhoria contínua da qualidade do processo de desenvolvimento de software. Com isso alguns modelos e normas e qualidades de software vêm ganhando importância, seja por sua qualidade ou aplicação. Os modelos de processo de software tem como objetivos melhorar a qualidade do sistema para tornar os projetos mais gerenciáveis, as datas de entregas e os custos mais previsíveis para que a equipe passa guiar e realizar o trabalho necessário para construir o sistema (PRESSMAN, 2006).

2.1 MODELOS DE MATURIDADE

Segundo Quintella e Rocha (2006), a definição de níveis de maturidade é baseada em princípios de qualidade de produtos, de acordo com os autores Crosby adaptou a estrutura de maturidade da qual esses princípios foram mostrados como “Aferidor de Maturidade da Gerência de Qualidade”, o qual citou os cinco estágios na adoção das práticas de qualidade: Incerteza, Despertar, Esclarecimento, Sabedoria e Certeza. Esses princípios foram adaptados pelo SEI, Software *Engineering Institute*, da Carnegie Mellon University, para o processo de desenvolvimento de software em 1986, pelo CMM (*Capability Maturity Model*). Os outros modelos de maturidade foram desenvolvidos ao longo do tempo: o PMMM (*Project Management Maturity Model*), desenvolvido pelo *Project Management Institute.* Com base no sucesso do CMM, outros modelos foram criados, procurando assim cobrir outras áreas de interesse como: o Software *Acquisition* CMM (SA-CMM), utilizado para avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de softwares desenvolvidos por terceiros; o People CMM (P-CMM), avalia a maturidade da organização em seus processos de administração de recursos humanos, no qual se refere a software; o *Integrated*

Product Development Capability Maturity Model (IPD-CMM), é incluído os processos necessários à produção e suporte ao produto, tais como suporte ao usuário, processos de fabricação entre outros.

O projeto do *CMMI* (Integração dos Modelos de Maturidade de Capabilidade) criado para preservar os investimentos governamentais e da indústria em melhoria de processos e para substituir e melhorar os múltiplos modelos de maturidade que surgiram ao longo dos anos, também facilita o uso da tecnologia CMM em diversas áreas (QUINELLA; ROCHA,2006).

2.1.1 MODELO CMMI

O modelo *Capability Maturity Model Integration* (CMMI) é um modelo de qualidade criado pelo *Software Engineering Institute* (SEI) que desenvolveu um abrangente metamodelo de processo que se dedica a um conjunto de forças de engenharia de software que devem estar presentes à medida que as empresas atingem níveis de capacidade e maturidade de processo. Para alcançar essas capacidades, o SEI afirma que uma empresa deve construir um modelo de processo que siga as normas do CMMI (PRESSMAN, 2006).

Desde a data de 1991, o SEI tem se destacado por ser responsável pelos modelos para engenharia de sistemas, engenharia de software, aquisição de software e desenvolvimento de produto e processo integrado. A ideia do CMMI é de abastecer-se no desenvolvimento e manutenção de produtos e serviços e também de uma estrutura extensível de modo que outros conhecimentos de corpo possam ser salvos (SAMARANI, 2005). O CMMI representa um metamodelo de processo de dois tipos de modos diferentes que são: o primeiro é um modelo contínuo e o segundo é um modelo em estágio. O CMMI define cada área de processo em termos de metas e práticas específicas, necessário para atingir as metas. As metas específicas estabelecem as características que devem existir se as atividades determinadas por uma área de processo tiverem de ser efetivas. As práticas específicas redefinem metas num conjunto de atividades relacionadas ao processo (PRESSMAN, 2006).

2.1.2 MODELO MPS.BR

O MPS.BR é um programa mobilizador, de longo prazo, criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), o MPS.BR tem como objetivo melhorar o Processo de Software e Serviços, com duas metas a alcançar a médio e longo prazos, metas essas que são:

a) meta técnica, que visa à criação e aprimoramento do Modelo MPS;
b) meta de negócio, que visa à disseminação e adoção do Modelo MPS em todo país, em um intervalo de tempo justo, a um custo razoável, tanto em micro, pequenas e médias empresas.

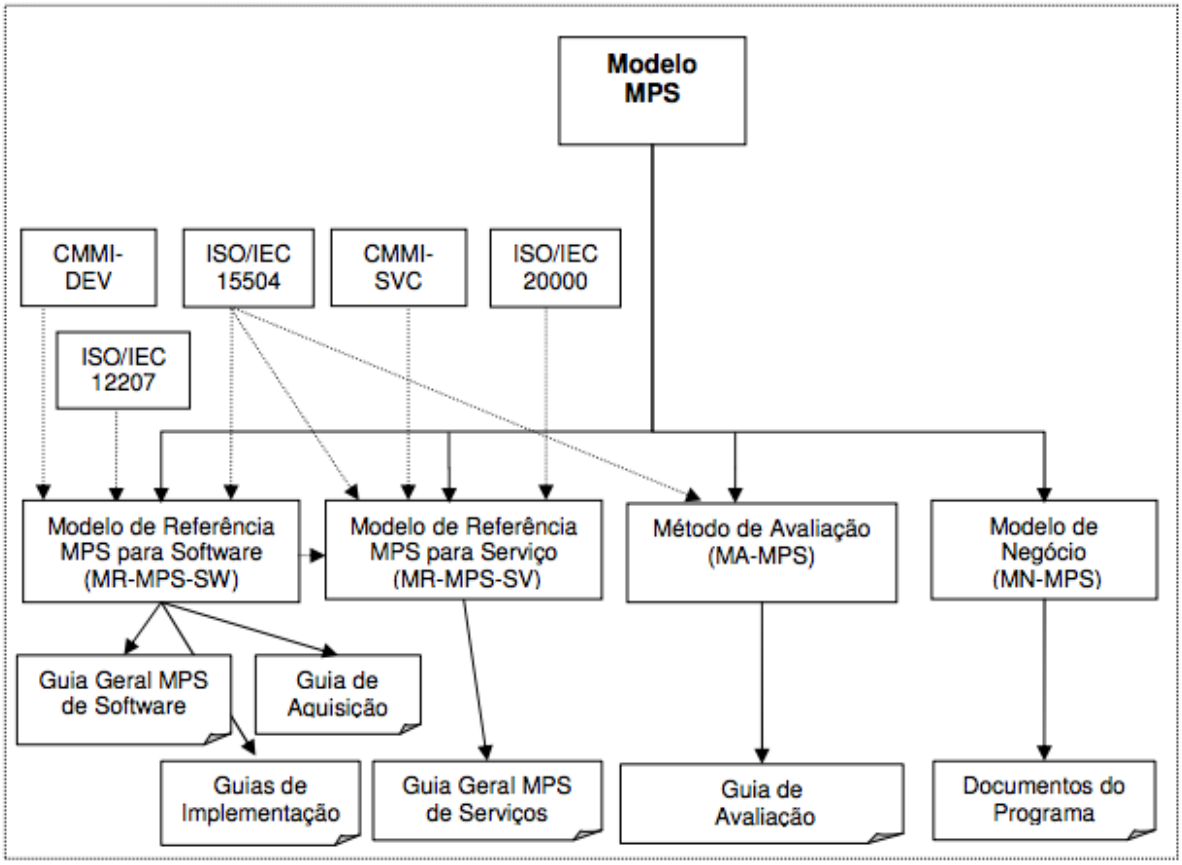
Mudanças tem ocorrendo nos ambientes de negócios têm motivado as empresas e também têm modificado estruturas organizacionais e processos produtivos, avisão tradicional tem ficado pra trás baseada em áreas funcionais em direção a redes de processos centrados no cliente. A competitividade depende, tem sido cada vez mais, do estabelecimento de conexões nestas redes, criando assim elos essenciais nas cadeias produtivas, alcançar competitividade pela qualidade, para as empresas de software, implica tanto na melhoria da qualidade dos produtos de software e serviços correlatos, como dos processos de produção e distribuição de software (SOFTEX, 2012).

De acordo com Softex (2012) o modelo MPS baseia-se em conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de software e serviços correlatos e também para a melhoria da qualidade e produtividade dos serviços prestados. O modelo MPS é formado por quatro componentes: Modelo de Referência MPS para Software (MR-MPS-SW); Modelo de Referência MPS para Serviços (MR-MPS-SV); Método de Avaliação (MA-MPS) e Modelo de Negócio para Melhoria de Processo de Software e Serviços.

O modelo MPS está descrito por meio de documentos em formato de guias:

a) Guia Geral MPS de Software: contém a descrição geral do modelo MPS e detalha o Modelo de Referência MPS para Software (MR-MPS-SW), seus componentes e as definições comuns necessárias para seu entendimento e aplicação;
b) Guia Geral MPS de Serviços: contém a descrição geral do modelo MPS e detalha o Modelo de Referência MPS para Serviços (MR-MPS-SV), seus componentes e as definições comuns necessárias para seu entendimento e aplicação;
c) Guia de Aquisição: descreve um processo de aquisição de software e serviços correlatos. É descrito como forma de apoiar as instituições que queiram adquirir produtos de software e serviços correlatos apoiando-se no MR-MPS-SW;
d) Guia de Avaliação: descreve o processo e o método de avaliação MA-MPS, os requisitos para avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA);
e) Guia de Implementação: série de documentos que fornecem orientações para implementar nas organizações os níveis de maturidade descritos no Modelo de Referência MR-MPS-SW.

Figura 1 - Componentes do modelo MPS



Fonte: Softex (2012).

Conforme a figura 1, é apresentado todos os componentes do modelo que foram descritos por meio de documentos específicos, com os tais respetivos guias.

3 GERENCIAMENTO DE PROJETO

A gerencia de projetos é um ramo bastante usada em várias empresas de variados tipos de negócio. Com o tempo foram comprovando os benefícios que ela proporciona, trazendo como principal resultado a redução de custos, que vem se destacando de acordo com o tamanho dos projetos.

De acordo com Molinari (2004), a disciplina de gerenciamento de projetos é aplicada em diversas áreas como: industrias, construção civil, sistemas de informação, educação entre outros.

Gerenciamento de projetos são práticas que remontam a antiguidade, à medida que teoria e prática de gerenciamento de projeto se ampliam surgem novas áreas de conhecimento proporcionando assim alicerces de um futuro promissor (CLELAND; IRELAND, 2007).

Em geral projeto significa empreendimento, que é um trabalho que mira na criação de um produto ou a realização de um determinando serviço especifico, temporário, não repetitivo e que envolve um certo grau de incerteza (MARTINS, 2010).

Os gerentes de projetos trabalham com quatro competências que são: conhecimento, habilidade, capacidade e competência, elas estão interligadas entre si proporcionando um bom projeto. A equipe de um projeto tem que ser organizacional exclusiva, pois dedica-se a entrega dos resultados de projetos no prazo tendo em conta o orçamento e especificações predeterminadas (CLELAND; IRELAND, 2007).

Segundo o Guia PMBOK (2008), o gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto afim de atender aos seus requisitos. O mesmo é realizado através da aplicação e da interação dos seguintes processos de gerenciamento de projetos: iniciação, planejamento, execução, monitoramento, controle e encerramento.

O gerenciamento de projeto traça e reconhece as principais áreas da prática e do conhecimento, áreas essas que são: Gerenciamento de Integração, Gerenciamento de escopo, tempo, custos, qualidade, recursos

humanos, comunicações, riscos, aquisições e integração. Estas áreas mostram as informações e boas práticas de relação ao gerenciamento de projetos em função dos processos que as compõem. A seguir, são apresentadas as áreas de cada uma delas (XAVIER, 2008).

O gerenciamento de Integração descreve os processos necessários para assegurar que os diversos elementos dos projetos sejam adequadamente coordenados; Escopo descreve os processos necessários para assegurar que o projeto contemple todo o trabalho requerido, e nada mais que o trabalho requerido, para completar o projeto com sucesso; Tempo descreve os processos necessários para assegurar que o projeto termine dentro do prazo previsto; o gerenciamento de custos descreve os processos envolvidos no planejamento, estimativa, cálculo do orçamento e controle de custos, de forma que o projeto termine dentro do orçamento planejado; gerenciamento de comunicação enfatiza os processos necessários para assegurar a geração, captura, distribuição, armazenamento e pronta apresentação das informações do projeto para que sejam feitas de forma adequada e no tempo certo; já o gerenciamento dos riscos refere os processos que dizem respeito à identificação, análise e resposta aos riscos do projeto. Nesse processo é importante decidir como abordar, planejar e executar as atividades de gerenciamento de riscos de um projeto; a gerência de qualidade relata a garantia de que o projeto será satisfatório, cumprindo assim o objetivo nos quais foi realizado; e por final o gerenciamento de recursos humanos explica os processos que organizam e gerenciam a equipe do projeto, porém consiste em identificar e documentar funções, responsabilidades e relações hierárquicas do projeto, e também produz plano de gerenciamento de pessoal.

O gerenciamento das aquisições descreve os processos necessários para aquisição de mercadorias e serviços fora da organização que desenvolve o projeto (XAVIER, 2008). Formando assim a produção dos resultados expressivos da organização, sendo esta redução de custo e prazo no desenvolvimento de novos produtos, aumento no tempo de vida dos produtos, aumento de vendas e receitas, aumento de número de clientes satisfeitos e aumento da chance do sucesso do projeto (PMBOK, 2008).

3.1 CICLO DE VIDA DE UM PROJETO

Segundo o Guia PMBOK (2008) o ciclo de vida de um projeto consiste nas fases do mesmo que normalmente é uma continuação que as vezes se seguem, cujo nome e número são decidido pela falta de gerenciamento e controle das organizações envolvidas. Um ciclo de vida pode ser definido de acordo com aspectos restritos da organização, indústria ou tecnologia empregada. Todos os projetos têm um início e um fim a ser definido, as entregas e atividades específicas conduzidas provisoriamente poderão mudar de acordo com o projeto (PMBOK, 2008).

3.2 MODELOS DE GERENCIAMENTO DE PROJETOS

3.2.1 MODELOS ÁGEIS

Segundo Boehm (2006), enfatiza que o desenvolvimento de software tem despertado grandes interesses entre as organizações do mundo inteiro. Pois, gerenciamento ágil foi criado através dos conceitos de desenvolvimento rápido como uma das alternativas afim de poder tratar com ambiente competitivo do mercado atual que exige resultados imediatos, sob condições de grandes incertezas e constantes mudanças.

De acordo com Nerur, Mahapatra e Mangalara (2005), este método é fundamental por fazer interações curtas e direcionada a produtos, com ações colaborativas e continua na integração de novas aplicações e rápida incorporação de alterações.

Atualmente, destaca-se entre os métodos ágeis de desenvolvimento de software o Modelo *Scrum*, Modelo *Extreme Programming* (XP), que serão abordadas em sequência.

3.2.1.1 MODELO SCRUM

Criado por Ken Schwaber e Jeff Sutherland (1996), o modelo *Scrum* proposto por eles, defende que o desenvolvimento de software é imprevisível e que deve ser separado por se destacar dos demais métodos ágeis por este estar mais focado ao processo de gerenciamento de projetos. Inicialmente o *Scrum* foi gerado com um estilo de gerenciamento de projetos em empresas de fabricação de automóveis e produtos de consumo

Segundo Schwaber (2004), o modelo *Scrum* reúne atividades de monitoramento e *feedback*, fazendo reuniões rápidas e diárias com toda a equipe, visando na identificação e correção de quaisquer deficiências ou impedimentos no processo de desenvolvimento.

A utilização deste método pode trazer benefícios tais como a satisfação do cliente, por meio da diminuição das reclamações; Melhoria na comunicação e aumento da colaboração entre envolvidos nos projetos; Aumento do retorno do investimento em projetos de novos produtos; Aumento da motivação da equipe de desenvolvimento de produtos; Melhoria da qualidade do produto produzido; Diminuição dos custos de produção; Aumento de produtividade da equipe de desenvolvimento; Diminuição no tempo gasto para terminar projetos de desenvolvimento de novos produtos; Diminuição do risco em projetos de desenvolvimento de novos produtos (CARVALHO; MELLO, 2012).

Este método não fornece qualquer técnica específica para a fase de desenvolvimento, apenas estabelecem conjuntos de regras e práticas gerenciais que devem ser adotadas para o sucesso de um projeto, ele é um modelo ágil que assume a premissa que o desenvolvimento de software não pode ser totalmente planejado no seu início por sua complexidade e por ser imprevisível. O modelo *Scrum* de ciclos mais curtos são chamados de *sprints* , o ponto inicial do *Scrum* é o *Backlog* do Produto é responsável pelo armazenamento dos requisitos coletados (CARVALHO; MELLO, 2012).

Este modelo tem quatro fases que são (SCHWABER, 2004):

a) **planejamento**: é a fase em que se estabelece a visão do projeto e são garantidos recursos para a execução das tarefas, se

determina a equipe que fará parte do projeto. Também são criados as versões inicias do *Backlog* do Produto;

b) ***stagging***: é a fase onde se avalia o tamanho do projeto, são criados itens adicionais ao *Backlog* do Produto relacionado com tipo do sistema, ambiente de desenvolvimento, tipo de aplicação, entre outros. Nessa fase a equipe selecionada para o projeto é dividida em times e também são definidos os mecanismos de comunicação e coordenação entre eles;

c) **desenvolvimento:** é a fase em que todas as funcionalidades especificadas no *Backlog* do Produto são divididas em *sprints* e desenvolvidas pela equipe responsável;

d) ***releasing*:** nesta fase o produto é entregue ao cliente após o desenvolvimento de todas as *sprints*.

O modelo *Scrum* tem uma serie de práticas de gerenciamento e artefatos que devem ser seguidas para obtenção de sucesso do projeto, dentre elas temos:

a) ***backlog* do Produto:** são os requisitos do projeto, nesta prática por meio de reuniões com todos os envolvidos, clientes, investidores e parceiros dos projetos, são apontados todas necessidades de negócios e funcionalidades a serem desenvolvidas;

b) ***daily Scrum*:** são realizadas reuniões diárias, é um rápido encontro entre os membros do time para definir quais serão as tarefas do dia e saber os resultados das tarefas anteriores, Três perguntas são respondidas por cada membro sobre suas responsabilidades: O que foi feito ontem? O que será feito hoje? Há algum obstáculo à realização das atividades? Os membros do time não respondem a essas perguntas como forma de prestar contas à gerência, mas sim como formalização do comprometimento com o resto da equipe;

c) *sprint*: é considerado a principal prática do *Scrum*. É o período de tempo no qual são implementados os itens de trabalho definidos no *Backlog* do Produto pela equipe *Scrum*.

d) **Sprint *Backlog*:** o *Backlog* do *Sprint* é um subconjunto do *Backlog* do Produto. Ele é uma lista de atividades a serem desenvolvidas durante o *Sprint*. Sua definição acontece durante a Reunião de Planejamento do *Sprint*;

e) ***Sprint Review Meeting*:** é a reunião que acontece após cada *Sprint*. Nela a equipe discute sobre seus erros, acertos e lições aprendidas;

f) ***sprint Planning Meeting*:** são reuniões onde as funcionalidades com maior propriedade de desenvolvimento, após esta reunião o *Sprint Backlog* é gerado.

No início do projeto, cliente e desenvolvedores definem o *Backlog* do Produto. E também são estimados os custos do projeto e definidas as datas para entrega de resultados a partir da priorização mais favorável ao cliente, análise inicial de riscos é preparada, as ferramentas de trabalho e os integrantes das equipes são escolhidos. Existe alguns papéis fundamentais como *Scrum* Master que cujo papel se assemelha a um gerente de projetos, trabalhando para que o processo *Scrum* aconteça e para que não existam impedimentos para os membros da equipe realizarem seu trabalho. Remover os obstáculos apontados na reunião de *Scrum* diária é seu dever, de modo que os desenvolvedores se concentrem apenas nas questões técnicas. Esses obstáculos são colocados em uma lista chamada de *Backlog* de Impedimentos, que fica à vista de todos. E o outro é outro papel importante *Product Owner:* Ele define quais são os requisitos e qual é o grau de importância e prioridade de cada um deles. Este membro necessita conhecer muito bem as regras de negócios do cliente, de forma que ele possa tirar qualquer dúvida que o time possa ter em relação às funcionalidades do produto (CARVALHO; MELLO, 2012).

No início de cada Sprint, quando as equipes fazem a lista das atividades que precisam ser realizadas naquele *Sprint* (*Backlog* do *Sprint*), as

responsabilidades são distribuídas. Os desenvolvedores discutem os padrões que serão adotados e as atividades de análise, codificação e testes se iniciam. Ao final de cada Sprint, uma versão do produto é apresentada ao cliente para obter a retroalimentação. Os defeitos encontrados são adicionados ao *Backlog* do produto. Ao longo de todo o projeto, são aplicados mecanismos de gerência *Scrum*, como o acompanhamento de alguns controles. A quantidade de funcionalidades não entregues, a necessidade de mudanças para corrigir defeitos ou para atualização tecnológica, os problemas técnicos encontrados e os riscos e as estratégias para evitá-los são exemplos de controles observados durante o desenvolvimento.

O último artefato do *Scrum* é o gráfico *burn down*: Trata-se de uma representação gráfica do trabalho restante em comparação com o trabalho já realizado. Geralmente, coloca-se a quantidade de trabalho no eixo vertical e o tempo no eixo horizontal. Ele é muito útil para predizer quando todo o trabalho será completado e para alarmar o time em caso de atraso (que ficará bastante aparente). Geralmente é traçada uma linha com a representação da execução do trabalho. Esta linha representa o esforço já realizado na execução das tarefas. Espera-se que a execução das atividades (tarefas) leve a linha de início em Y ao encontro de X. Este encontro representa o término das execuções das tarefas (CARVALHO; MELLO, 2012).

3.2.1.2 PLANNING POKER

A *Planning Poker* é uma técnica de estimativa focada para metodologias ágeis, inclusive para o *Scrum*, diferente dos métodos anteriores onde o seu foco consiste nas metodologias tradicionais.

É um jogo de cartas para a obtenção de estimativa, onde a principal meta por trás é que toda a equipe de desenvolvimento participe usando sua visão de complexidade, sendo que assim todos juntos possam chegar em um consenso de ideias em comum para a equipe. Nas metodologias ágeis, as estimativas de esforço e tempo são aplicadas a estórias e é essa atividade é de responsabilidade da equipe técnica por alguns motivos (SILVEIRA, 2012):

a) Na parte de planeamento, geralmente não se sabe quem vai implementar quais partes de quais estórias;
b) Quando se tem diversas pessoas com diferentes conhecimentos se envolvem as estórias (design de interface de usuário, teste, codificação, etc.);
c) Cada estória precisa da compreensão de algum tipo do membro da equipe, para se obter uma estimativa. No contexto para um melhor entendimento os membros se ajudarão na interação. Isso faz com que as estórias de questões importantes surjam cedo;
d) Quando todos os membros estimam uma estória, provavelmente surgira dois membros com estimativas diferentes para a estória. Porém será preciso descobrir e discutir essa pendência antes do início da interação.

No Scrum, o *Planning Poker* é realizado durante o Sprint Planning Metting. Com o objetivo de atribuir uma complexidade em comum a todos os membros da equipe para as estórias de cada item do Product Backlog com uma forma mais dinâmica. Portanto, são os principais responsáveis pelas estimativas das estórias.

> “No Planning Poker, cada integrante tem a sua disposição um baralho de 13 cartas, numeradas numa sequência similar a encontrada nos números de Fibonacci, conforme mostra a Figura 2. De acordo com (SILVEIRA,2012) existe um propósito ao utilizar uma sequência não linear determinação de estimativas. O objetivo de utilizar estes números é que o intervalo entre os valores permite visualizar melhor as diferenças de complexidade entre as funcionalidades. Essa ideia diminui a granularidade das estórias, o que evita uma falsa ideia de precisão de estimativas. Então se para um item foi dada uma estimativa de 20 pontos, não faz muita diferença se pudesse ser dado um valor próximo. Portanto, este valor trata-se de um palpite aproximado”.

Figura 2 - Baralho de Planning Poker

Fonte: Silveira (2012).

A figura acima, mostra o baralho de Planning Poker onde os valores acima de 20 são considerados altos e, portanto, trata-se de uma estória grande, que poderam não ser completamente finalizadas numa única interação.

3.2.1.3 MODELO XP

O *Extreme Programming* (XP) é uma metodologia ágil para o desenvolvimento de software que se direciona, principalmente, a resolver limitações do processo. No final da década dos anos 90, o modelo *Extreme Programming* ou XP, mostra uma maneira diferente de como gerenciar um projeto. Neste modelo muita de suas regras causam certo embate no primeiro momento e outras delas não apresentam sentido caso sejam aplicadas de forma isolada. Porém, o conjunto que o modelo XP apresenta faz dele um sucesso, que visa trabalhar no desenvolvimento rápido garantindo assim a satisfação dos clientes e cumprimento de metas e estimativas. Segundo Beck (1999), XP proporcionam regras, práticas e valores agradáveis em ambientes de desenvolvimento de software para as equipes, que são controladas por quatro valores: comunicação, simplicidade, *feedback* e coragem.

a) **comunicação:** muitos dos problemas que ocorrem no decorrer dos projetos podem estar ligados ao problema da comunicação

entre a equipe ou entre a equipe de projeto com o próprio cliente. As pessoas envolvidas nesse processo devem estar ligadas fisicamente uma das outras, de modo que haja uma comunicação mais fácil afim de acelerar o fluxo de informações. Os projetos que optam em utilizar o XP acabam por ter poucos integrantes, visto que assim acaba por facilitar na troca de informações entre as equipes do projeto, pois, desse modo é mais fácil o integrante saber o que seu companheiro está desenvolvendo no momento e não poder interrompe-lo;

b) **simplicidade**: de acordo com os estudos da *Standish Group*, mais de 50% das funcionalidades dos sistemas não costumam ser utilizadas pelos usuários, ou seja, mais da metade dos softwares não precisariam ser implementado. O modelo XP, se baseia no fato de que é mais barato fazer algo mais simples podendo assim altera-lo conforme as necessidades forem aparecendo do que tentar prever necessidades futuras, adicionando uma complexidade que possa não ser importante no futuro;
c) ***feedback*:** no modelo XP todo o problema deve ser claro o mais cedo possível facilitando assim a resolução de imediata. O *feedback* é o valor primordial dentro do desenvolvimento do modelo ágil, o XP foi o dianteiro em falar de feedback e também afirmar que possibilita que o software evolua;
d) **coragem:** o processo de desenvolvimento de software que é bastante complexo, podendo assim ter existência de vários erros e isso obriga coragem por parte da equipe. No modelo XP, deve se ter coragem de colocar o cliente sempre a par dos acontecimentos, e partir do princípio que podem vir a ocorrer problemas graves. Sendo assim a necessidade de criar uma rede de proteção ao início do projeto com objetivo de reduzir ou eliminar os possíveis problemas que possam surgir.O XP considera que deve-se ter coragem de fazer :Acreditar na capacidade de reagir a mudanças; Trocar de paradigma;

Aprender com os erros; Dar e receber *feedback* sem medo das consequências; Acreditar no *feedback* concreto; Fazer o que precisa ser feito; Jogar fora código ruim; Jogar fora protótipos criados para testar ideias.Fazendo isso a uma importante garantia da rentabilidade e qualidade do projeto final.

Para além dos valores abordados anteriormente XP utiliza um conjunto de princípios que devem ser seguidos por equipes que forem usar XP em projetos. Os princípios servirão para ajudar na escolha de alternativas de solução de problemas durante o curso do projeto. Deve-se preferir uma alternativa que atenda aos princípios de uma forma mais completa do que outra que seja incompleta, ou seja, que esteja mais longe da filosofia de XP. Os princípios fundamentais são: *feedback* rápido, assumir simplicidade, mudança incremental, abraçando mudanças e trabalho de qualidade.

a) ***feedback* rápido:** a ideia de XP é que os participantes de um projeto como clientes, programadores e gerentes devem estar sempre se comunicando para ter maior facilidade no aprendizado coletivo sobre o projeto que está sendo desenvolvido e também de alertar rapidamente sobre dúvidas, riscos e problemas para facilitar eventuais ações eventuais;
b) **assumir simplicidade:** todos problemas devem ser tratados para serem resolvidos da forma mais simples possível. XP afirma que se deve fazer um bom trabalho para resolver hoje os problemas de hoje e confiar na sua habilidade de adicionar complexidade no futuro quando for necessário;
c) **mudança incremental:** quando muitas mudanças são feitas todas de uma vez, não se obtém um bom resultado. as mudanças devem ser acrescentadas e feitas aos poucos. Os programadores têm a liberdade para estar sempre fazendo alterações de melhoria no código e as mudanças de requisitos são incorporadas de forma incremental;
d) **abraçando mudanças** (*Embracing Change*): XP procura facilitar o ato de incluir alterações através do uso de vários princípios e

práticas. As mudanças devem ser sempre bem vindas independentemente do estágio de evolução do projeto. Isso é muito bom em projetos cujos requisitos são bastante voláteis devido aos clientes não saberem o que querem;

e) **trabalho de qualidade**: em XP, qualidade não é um critério opcional, mas sim obrigatório. Embora a definição de qualidade possa variar de pessoa para pessoa, XP trata qualidade no sentido de se ter um sistema que atenda aos requisitos do cliente, que rode 100% dos casos de teste e que agregue o maior valor possível para o negócio do clientes.

Para um bom funcionamento e sucesso de um projeto o XP utiliza treze práticas que são (BECK, 2004):

a) **cliente Presente:** deve ser incluído na equipe uma pessoa da parte do cliente, que irá usar o sistema, para trabalhar junto com os outros e responder as perguntas e dúvidas. Mesmo que não seja possível obter alguém da parte do cliente deve-se ter alguém que tenha conhecimento suficiente do negócio para exercer este papel. O cliente tem um papel importante dentro de um projeto XP já que ele participa do planejamento do projeto escrevendo as histórias e priorizando-as;

b) **jogo do Planejamento:** o planejamento de um *release* e das iterações são feitos com base nas histórias (casos de uso simplificados) e conta com a colaboração de toda a equipe de desenvolvimento, inclusive o cliente, divididos em dois papéis:

a. **negócio:** participam as pessoas que mais entendem sobre o negócio e que possam estabelecer prioridades para as funcionalidades a serem entregues;

b. **técnico:** participam as pessoas que irão implementar as funcionalidades descritas. Os técnicos estimam qual o esforço e riscos envolvidos para implementar as funcionalidades e comunicam ao pessoal de negócios.

No final de um *release* é feita uma determinação rápida do escopo do próximo, através da combinação de estimativas e prioridades do negócio. Um *release* consiste de várias iterações e, em cada iteração, várias histórias (use case simplificados) são implementadas. Os programadores estimam cada estória e dizem quantas eles podem implementar no final do *release*. Baseado nesses dados, os clientes escolhem as principais estórias (que agregam maior valor para o negócio) que serão implementadas. As estimativas podem ser refeitas durante as iterações à medida que os programadores aprenderem mais sobre o sistema.

a) ***stand Up Meeting* ou Reunião Em Pé:** a equipe envolvida do projeto realiza diariamente uma reunião para discutir tudo sobre o que se realizou anteriormente, com isso a equipe fica sabendo dos resultados obtidos e como esta a organização das atividades diários;

b) **programação em Par:** todo o código produzido em XP é escrito por um par de programadores, que possuem papéis distintos, sentados lado-a-lado e olhando para o computador. Um parceiro será responsável pela codificação e pensará nos algoritmos e na lógica de programação. O outro parceiro observa o código produzido e tenta pensa mais estrategicamente em como melhorá-lo e torná-lo mais simples, além de apontar possíveis erros e pontos de falha. Além disso, as duplas são constantemente trocadas e os papéis também com o objetivo de que todos os membros da equipe possam ter conhecimento sobre todas as partes do sistema**;**

c) **código Coletivo:** a programação em pares encoraja duas pessoas a trabalharem juntas procurando atingir o melhor resultado possível. A propriedade coletiva encoraja a equipe inteira a trabalhar mais unida em busca de qualidade no código fazendo melhorias e refatoramentos em qualquer parte do código a qualquer tempo. A princípio pode-se pensar que esta prática possa gerar problemas, mas como todos devem respeitar um

padrão de codificação e devem realizar todos os testes para verificação de erros esta atividade é feita de uma forma controlada e pode ser facilitada com o uso de uma ferramenta de controle de versão;

d) **código Padronizado:** como XP prega a propriedade coletiva de código, onde todos podem alterar e fazer refatoramento de qualquer parte do código a qualquer momento, então é mais do que necessário que se tenha padrões de codificação. O objetivo é que todos programem da mesma forma, facilitando o entendimento do código e as alterações;

e) ***design* Simples:** pode-se explicar esta prática em duas partes: A primeira diz que devem ser projetadas as funcionalidades que já foram definidas e não as que poderão ser definidas futuramente. A segunda diz que deve ser feito o melhor projeto que possa entregar tais funcionalidades. Esta prática tem o intuito de enfatizar que o projeto simples deve se concentrar em soluções simples e bem estruturadas para os problemas de hoje e que não se deve perder tempo investindo em soluções genéricas que procurem atender a funcionalidades futuras, pois como os requisitos mudam constantemente tais soluções genéricas podem não ser mais a realidade do futuro;

f) **desenvolvimento Orientado a Teste:** os testes em XP são feitos antes da programação. Existem dois tipos de teste: teste de unidade e teste funcional. Os testes de unidade são feitos para verificar tudo que possa dar errado. Os testes unitários são automatizados, e toda vez que o programador escrever código, ele irá verificar se o sistema passa em todos os testes. Os testes funcionais são usados para verificação, junto ao cliente, do sistema como um todo. Os testes servem como um mecanismo para assegurar que o sistema está sempre rodando livre de erros, e também servem para dar feedback aos programadores e clientes sobre as falhas encontradas.

g) **refatoração:** São constantes melhorias no projeto do software para aumentar sua capacidade de se adaptar a mudanças. O refatoramento consiste em aplicar uma série de passos, tornando-o mais simples e melhor estruturado, sem alterar sua funcionalidade;

h) **Integração Continua:** o código das funcionalidades implementadas pode ser integrado várias vezes ao dia. Um modo simples de fazer isso é ter uma máquina dedicada para integração. Quando a máquina estiver livre, um par com código a integrar ocupa a máquina, carrega a versão corrente do sistema, carrega as alterações que fizeram (resolvendo as colisões), e rodam os testes até que eles passem (100% corretos). O importante é que na integração as funcionalidades só podem ser integradas se não houver erros, caso contrário os erros devem ser corrigidos;

i) ***releases* Curtos:** cada *release* deve ser tão pequeno quanto possível, contendo os requisitos mais importantes para o negócio". Isso possibilita ter *releases* frequentes o que resulta em maior *feedback* para clientes e programadores, facilitando o aprendizado e a correção dos defeitos do sistema. Beck (2004) sugere que os *releases* sejam de curta duração, normalmente de dois a três meses;

j) **metáforas:** a intenção da metáfora é oferecer uma visão geral do sistema, e num formato simples, que possa ser compartilhada por clientes e programadores. A ideia da metáfora é que seja feita uma analogia entre o sistema que está sendo desenvolvido e um sistema, não necessariamente de software, que todos entendam, com o objetivo de obter um "vocabulário comum" para a posterior criação de nomes de classes, subsistemas, métodos. A metáfora conforme também pode ser vista como uma forma simples de arquitetura do sistema, numa linguagem compreensível tanto por clientes como programadores**;**

k) **ritmo sustentável:** essa não é uma regra que obriga as equipes em projetos XP a trabalharem somente 40 horas por semana. No entanto, não se deve trabalhar duas semanas seguidas além desse tempo, pois o cansaço e a insatisfação de trabalhar horas extras pode resultar numa queda de qualidade do código (BECK,2004).

Se a equipe envolvida no projeto está focada em seguir as praticas, o resultado positivo é garantido, pois os integrantes do projeto devem se envolver com dedicação para que balanço positivo seja alcançado.

Um projeto XP atravessa algumas fases durante o seu ciclo de vida. Essas fases são compostas de várias tarefas que são executadas. O projeto XP passa pelas seguintes fases (BECK, 2004):

a) **a fase de exploração** nela, investigações de possíveis soluções são feitas e verifica-se a viabilidade de tais soluções. Os programadores elaboram possíveis arquiteturas e tentam visualizar como o sistema funcionará considerando o ambiente tecnológico (hardware, rede, software, performance, tráfego) onde o sistema irá rodar. Com isso, os programadores e os clientes vão ganhando confiança, e quando eles possuírem estórias suficientes, já poderão começar a construir o primeiro *release* do sistema;

b) **a fase de planejamento inicial** deve ser usada para que os clientes concordem em uma data para lançamento do primeiro *release*. O planejamento funciona da seguinte forma: Os programadores, juntamente com o cliente, definem as estórias (use case simplificados) a serem implementadas e as descrevem em cartões. Os programadores assinalam uma certa dificuldade para cada estória e, baseados na sua velocidade de implementação, dizem quantas estórias podem implementar em uma iteração. Depois, os clientes escolhem as estórias de maior valor para serem implementadas na iteração – isso é chamado planejamento de iteração. O processo então se repete até

terminar as iterações do *release*. O tempo para cada iteração deve ser de uma a três semanas e para cada *release* de dois a quatro meses;

c) **na fase das iterações do *release*:** são escritos os casos de teste funcionais e de unidade. Os programadores vão seguindo mais ou menos o seguinte fluxo de atividades na seguinte ordem (Em cada iteração): escrita dos casos de testes; projeto e refatoramento; codificação; realização dos testes; e integração. À medida que esse fluxo vai sendo seguido, o sistema vai sendo construído segundo os princípios, valores e práticas apresentados nas seções anteriores. Depois de terminado o primeiro *release*, já se terá uma ideia melhor das tecnologias e do domínio do problema de modo que as iterações poderão ser mais curtas nos *releases* subsequentes e já se podem fazer estimativas mais confiáveis com o que se aprendeu das iterações passadas;

d) **a fase de produção** onde cada *release* subsequente do sistema, depois de construído, é colocado para rodar em um ambiente que simula o ambiente de produção para ver seu comportamento em termos de performance. Pode-se fazer testes de aceitação adicionais para simular o funcionamento real do sistema no ambiente alvo;

e) **a fase de manutenção** pode ser considerada como uma característica inerente a um projeto XP. Em XP você está simultaneamente produzindo novas funcionalidades, mantendo o sistema existente rodando, incorporando novas pessoas na equipe e melhorando o código. Mecanismos como: refatoramento, introdução de novas tecnologias, e introdução de novas ideias de arquitetura podem ser utilizados em um projeto XP. É importante ressaltar que a manutenção dada em um sistema que já está em produção deve ser feita com muita cautela, pois uma alteração errada pode paralisar o funcionamento do sistema resultando em prejuízos para o cliente;

f) a fase de morte corresponde ao término de um projeto XP. Existem duas razões para se chegar ao final de um projeto, uma boa e a outra ruim. A boa razão é quando o cliente já está satisfeito com o sistema existente e não enxerga nenhuma funcionalidade que possa vir a ser implementada no futuro.

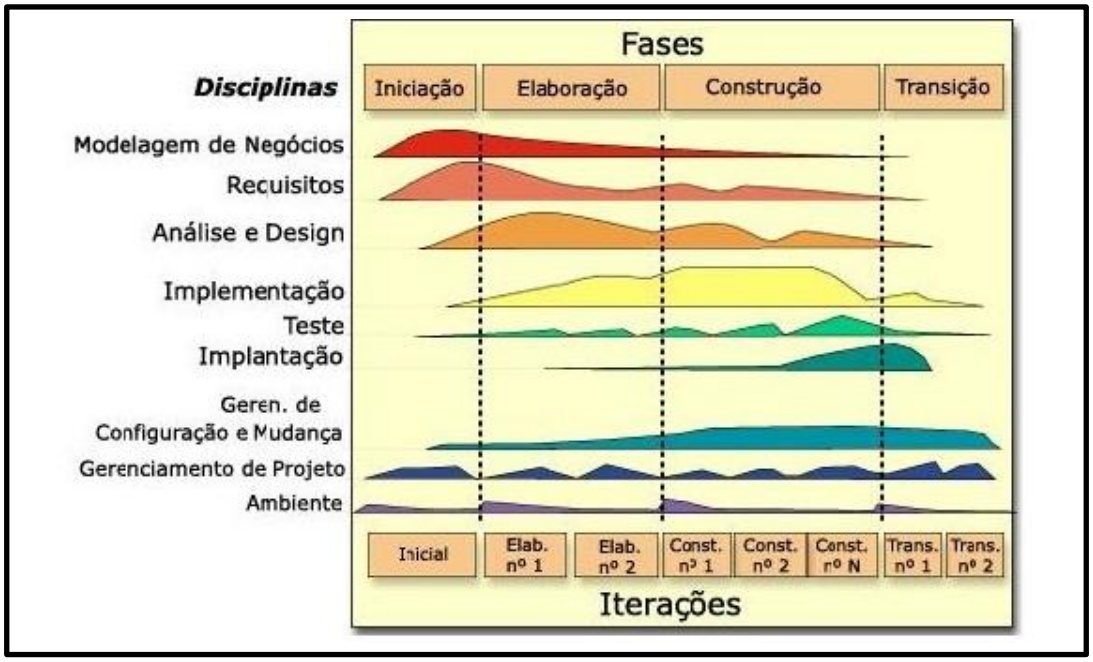
3.3 MODELOS TRADICIONAIS

3.3.1 RUP

O *Rational Unified Process* (RUP), é um processo proprietário de engenharia de software criado pela empresa *Rational Software Corporation* para apoiar desenvolvimentos orientados de objetos e fornecendo uma forma sistemática de obter vantagem no uso de UML, e ele foi posteriormente adquirido pela gerência de Projetos de Software com utilização de práticas tradicionais. Este processo descreve técnicas a serem seguidas pelos membros da equipe de desenvolvimento de software com o objetivo de aumentar a sua produtividade. O RUP oferece uma abordagem baseada em uma matriz de disciplinas e fases que atribui tarefas e responsabilidades para a organização de desenvolvimento de software. A meta é garantir a produção de software de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e de um orçamento previsíveis (RUP, 2003).

As disciplinas de RUP a serem seguidas para obtenção de uma organização de software para atingir qualidade necessária são : Modelagem de Negócios; Requisitos; Análise e Design; Implementação; Teste; Implantação; Gerência de Configuração e Mudança; Gerência de Projeto; Ambiente. estas disciplinas definem gerência de projetos como a arte de equilibrar objetivos concorrentes, gerenciar risco e superar restrições para entregar de forma bem sucedida o produto que satisfaça às necessidades das partes envolvidas (REHMAN, 2007).

Figura 3 - Gráfico RUP



Fonte: (REHMAN, 2007).

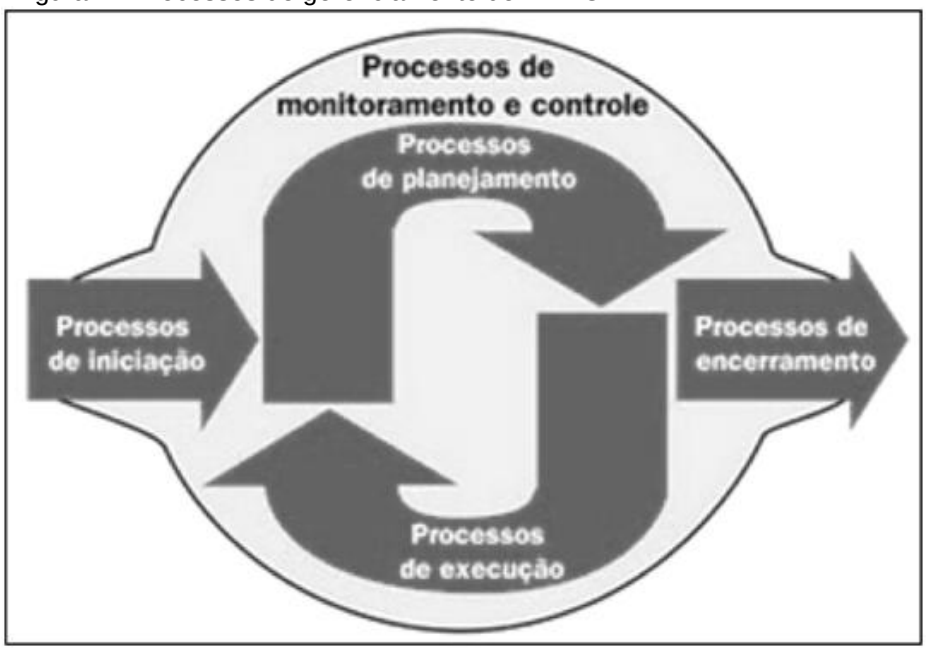
Conforme o gráfico acima na figura 2, ilustra uma ligação entre as quatro fases do modelo RUP e as suas importantes disciplinas conforme dito anteriormente. Atráves deste gráfico é possivel identificar a existencia de cada disciplina ao longo das fases do modelo.

3.3.2 PMBOK

O PMBOK é um guia do conhecimento em gerenciamento de projetos que diz respeito a gerenciamento de projetos. É um documento formal que descreve normas, métodos, processos e praticas estabelecidas, foi criado pelo *Project Management Institute* (PMI), este modelo é criado a partir de boas praticas reconhecidas de profissionais de gerenciamento de projetos que contribuíram para o seu desenvolvimento. Este guia se fundamenta em processos e subprocessos que relatam de forma organizada qual das atividades que devem ser executadas em cada etapa do projeto (PMBOK, 2008).

De acordo com o PMBOK (2008) estes projetos, sejam eles pequenos ou grandes, podem apresentar um ciclo de vida geral representado por quatro fases, consideradas como início do projeto, Organização e preparação, Execução das atividades e Encerramento do projeto. Com isso, o ciclo de vida do projeto apresenta cinco processos de gerenciamento conhecidas como “grupos de processos de gerenciamento de projetos“.

Figura 4 - Processos de gerenciamento do PMBOK



Fonte: PMBOK

A figura 3, mostra de forma clara o fluxo dos processos gerenciais detalhados mais abaixo.

Processo de iniciação: neste ponto as partes interessadas no projeto são identificadas e o escopo inicial é definido. São garantidos os recursos financeiros necessários inicialmente. É necessário o Termo de Abertura do Projeto envolvendo cliente x empresa deve ser criado e aprovado para continuação do projeto.

Processo de planejamento: neste momento do projeto, a equipe e o cliente devem garantir uma relação forte para o desenvolvimento completo e correto do escopo e os 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.

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.

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.

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.

4 TEORIA DA CORRENTE CRÍTICA

De um modo geral, o método da corrente critica(*Critical Chain Project Management*), é um método para gerenciamento de projetos que, visa focar basicamente na administração de prazos e atividades, considerando alocação de recursos, baseado na Teoria das Restrições (TOC). A sua origem é relacionada com a publicação do livro intitulado Corrente Critica, escrita por Eliyahu M. Goldratt, teve seu início através de observações básicas de problemas comuns em projetos de alta probabilidade onde ultrapassava-se o orçamento, tempo e comprometimento de conteúdo.

A corrente critica é um dos métodos mais importante do gerenciamento de projeto, se fundamenta na teoria de restrições, ela tem grande impacto no reconhecimento dos recursos associados tornando assim o método importante para ambientes multiprojetos. Nos anos 90 a corrente critica foi desenvolvida para justificar a existência de problemas crónicos de métodos e abordagens existentes (SILVA; RODRIGUES; LACERDA,2012).

De acordo com o Gordratt (1997) o principal problema no gerenciamento de projetos são as incertezas, elas originam dificuldades e obstáculos. Para solucionar as incertezas estimativas dos tempos das atividades são inflacionados adicionando também segurança.

Existe três fenômenos que interferem nas execuções de atividades: primeiro síndrome de estudante que consiste em esperar até o ultimo momento para iniciar as atividades, estimativas de duração são baseadas em experiências pessimistas; segundo multitarefa consiste na consequência gerada pelo ambiente em que vários projetos são desenvolvidos paralelamente, quanto maior o níveis gerências maior a duração total de estimativa da atividade, assim os diferentes níveis adicionam segurança; terceiro lei de Parkinson consiste no principio que as atividades tendem a se expandir ocupando o tempo disponível, as pessoas estimam as durações protegendo-se de cortes (GOLDRATT,1997).

Com o propósito de eliminar os problemas acima, a Corrente Critica propõe uma quebra de paradigma, diminuindo em média, 50% na estimativa

inserida em cada tarefa do cronograma. Porém, essa ” eliminação de gordura ” torna o projeto mais frágil a possíveis atrasos provocadas na mudança e incertezas a que qualquer projeto se encontra submetido.

Para implementar a corrente critica na gestão de projeto é necessário definir a restrição do sistema ou seja corrente critica, pois esta determina a duração do projeto, decidir como explorar a restrição e não desperdiçar tempo com os três fenómenos, admitir que o projeto precisa de proteção quanto as incertezas, a proteção deve ser destinado no final da corrente critica (SILVA; RODRIGUES; LACERDA, 2012).

O projeto normalmente mede-se conforme o trabalho ou investimentos realizado em relação ao valor a ser desenvolvido, sendo assim não há diferença entre trabalho realizado em caminho critico e trabalhos de outros caminhos. A corrente critica tem como objetivo não programar atividades executadas por um mesmo recurso em paralelo, consequentemente elimina a chance de programar atividades que originam multitarefas (GODRATT, 1997).

Godratt (1997) explica que os princípios devem ser utilizados em ambientes multiprojetos, identificando assim recurso limitante para o conjunto de projeto e dar sequencia as atividades de acordo com prioridade do projeto. Sendo assim, este projeto se originou da necessidade de melhor gerenciar projetos no ambiente estudado junto a uma proposta de estudo da aplicação da corrente crítica.

A corrente crítica da mesma forma que a teoria das restrições busca alcançar de forma significativa o melhor resultado e busca também mudanças utilizando o gerenciamento de projeto como princípios tradicionais. Ela sugere uma diminuição agressiva na estimativa de tempo por tarefas, com isso também a uma diminuição de incidência de Lei de Parkinson e síndrome de estudante (QUELHAS; BARCAUI, 2010).

4.1 LEI DE PARKINSON

A Lei De Parkinson refere que o trabalho se expande para preencher todo o tempo disponível, e também é muito comum encontra-la dentro de organizações de vários tipos de negócios. Uma tarefa tem seu prazo maior do

que o necessário para sua execução, a pessoa responsável pela tarefa ao invés de manter o ritmo para finalizar antes do prazo, diminui o seu ritmo para terminar a tarefa no prazo estipulado, isso é prejudicial para qualquer organização pois o prazo é cumprido com isso a equipe vai continuar trabalhando sempre a baixo de sua capacidade máxima de produção (SILVA; RODRIGUES; LACERDA, 2012).

4.2 MULTITAREFA

Em ambiente de múltiplos projetos, como é visto em várias organizações, muitas vezes o mesmo recurso é colocado e retirada para efetuar inúmeras tarefas. Tendo como consequência por vezes uma má utilização de recursos, uma vez que há uma perda de foco e um “ *delay* ” de tempo quando são necessárias as mudanças entre tarefas diferentes em projetos diferentes. Essa situação geralmente, transporta, atrasos nas tarefas, e consequentemente, no prazo do projeto caso essas tarefas pertençam a caminho crítico.

4.3 SINDROME DO ESTUDANTE

Síndrome de estudante é da natureza humana guardar uma atividade até que ela se torna realmente urgente para então realiza-la, este comportamento é melhor entendido quando associado ao aluno em período letivo, a maioria dos alunos só realiza seu trabalho sempre na última hora possível. Este comportamento também é comum dentro de organizações, as tarefas são distribuídas e sempre tem algumas pessoas que o realizarão no último momento, se houver imprevistos a entrega do trabalho será atrasado provocando prejuízo para as organizações (SILVA; RODRIGUES; LACERDA, 2012).

4.4 TEORIA DAS RESTRIÇÕES

Na década dos anos 70 teve o começo da Teoria das Restrições (TOC) quando o Israelense Eliyahu Goldratt teve problemas com a logística de produção de uma organização.

Com isso Goldratt teve de elaborar um novo método de produção. Este mesmo método elaborado por ele, teve um ótimo desempenho no mercado e assim várias empresas tiveram o interesse de aplicar esta mesma técnica. A partir daí Goldratt se focou em continuar o trabalho podendo assim espalha-lo.

A teoria das restrições (TOC) é definida como abordagem de gestão voltada para melhoria dos processos que limitam o fluxo da produção com intuito de melhorar continuamente o desempenho das produções, buscando assim a filosofia de melhoria por meio de identificação das restrições de um sistema (BRANDÃO et al, 2012).

A restrição de um sistema é qualquer coisa que impeça o sistema de atingir um desempenho maior que a sua meta, por este motivo é necessário conhecer a meta do sistema (GOLDRATT, 1997).

De acordo com a teoria e com a base das ideias inicias a principal meta das empresas é seu resultado financeiro, se a empresa não possuísse restrição seu lucro seria infinito, assim considere-se dois tipos de restrições: física e não física (políticas e emocionais) (NOGUCHI, 2006).

Com isso, a TOC sugere tratar as restrições por meio de seu processo de pensamento: o que mudar? Como provocar a mudança? Mudar para o que? Este processo de pensamento sugerida pela TOC é para uma lista de sintomas para desenvolver análises de causa-e-efeitos com objetivo de identificar a principal causa do problema (MARCANTONIO, 2010).

A teoria aumenta um conjunto de procedimentos para otimizar e identificar as restrições, procedimento estes que abrangem cinco passos que são (BRANDÃO; et al., 2012):

a) Identificar as restrições do sistema por meio da realização de cálculos da capacidade de cada máquina versus a capacidade solicitada para a produção;
b) Exploram-se as decisões e é preciso identificar a melhor forma de explorar as restrições. É preciso atingir a melhor taxa de rendimento possível;
c) Os demais recursos são subordinados, nesta etapa coloca-se que os demais recursos devem trabalhar no mesmo ritmo da restrição, não mais rápido e nem mais devagar, pois não estariam aumentando o nível de produção da linha;
d) Eleva-se a produção, deve-se aumentar a produção da restrição, assim, se a restrição for uma máquina, outra pode ser adquirida, ou parte da produção que passaria pela restrição pode ser enviada para fábricas externas;
e) Elevar a inércia do sistema. É preciso renovar o ciclo de melhoria para elevar a inércia do sistema. Se a restrição dos passos anteriores foi quebrada deve-se começar de novo. Sempre buscando a melhoria contínua.

Os cinco passos da TOC podem ser usados como soluções para resolver diversos problemas na organização, a teoria das restrições apresenta inovadoras técnicas para solucionar problemas de restrições na organização como um todo e pode ser implementada em várias áreas de atuação. As restrições não são apenas ruins, uma organização se não possuísse restrições o seu desempenho seria teoricamente imenso. A restrição é a limitação de qualquer uma organização em conseguir um desempenho no seu fluxo de trabalho. Sendo assim toda organização possui restrições (NOGUCHI, 2006).

5 TRABALHOS CORRELATOS

Durante o desenvolvimento desta pesquisa, foram estudados diferentes trabalhos com abordagem na teoria da corrente crítica e gerenciamento de projetos, semelhantes ao trabalho proposto, mas com foco em diferentes áreas especificas.

A seguir serão descritos alguns desses trabalhos.

5.1 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 de Leandro Daminelli em Ciência da Computação, em 2011, pelo Departamento de Ciência da Computação, da Universidade do Extremo Sul Catarinense.

O trabalho tem como objetivo, propor 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.

5.2 ACORRENTE CRITICA APLICADA NA FERRAMENTA DE GESTÃO DE PROJETOS MS PROJECT

O artigo escrito por Maria Isabel Palmeiro Marcantonio que foi apresentado no encontro nacional de engenharia de produção que ocorreu em 14 e 15 de outubro de 2010 em São Paulo, o presente trabalho tem como objetivo apresentar os conceitos fundamentais da Corrente Crítica desenvolvidos por Eliyahu Goldratt e aplicá-los na ferramenta de gestão de projetos MS Project É um estudo teórico aplicado a ferramenta.

5.3 GUIA PMBOK

Utilizou-se o guia PMBOK que é um livro elaborado pela instituição mais renomada do mundo na área, de gerenciamento de projeto o Project Management Institute PMI, o guia tem como objetivo de abordar sobre os principais aspectos contidos no gerenciamento de um projeto.

5.4 PROPOSTA PARA USO DA CORRENTE CRÍTICA NO GERENCIAMENTO DE MÚLTIPLOS PROJETOS

A tese de Robert Eduardo Cooper Ordoñez para obtenção do grau de Doutor em engenharia mecânica, em 2013, pela Universidade Estadual de Campinas na área de Materiais e Processos de Fabricação.

A tese tem como intuito propor um modelo que utilize os fundamentos do método da corrente crítica para sua aplicação no gerenciamento do tempo das atividades em sistemas de múltiplos projetos.

6 PERSPETIVA PARA ELABORAÇÀO DA TEORIA DA CORRENTE CRÍTICA

A presente pesquisa tem como finalidade criar um método da teoria da corrente crítica para minimizar o gerenciamento de projetos, visto que na área de projetos, existe muito conhecimento guardado sobre maneiras de gerenciar um projeto. Para isso, dentro dos diferentes modelos de gerenciamento, existem inúmeras formas de efetuar com relação ao gerenciamento de custos, prazos, recursos, porém nenhum desses modelos esclarece uma maneira diferente sobre gestão de atividades quanto estimativas de tempo.

A estimativa do tempo da atividade é o processo de estimar o número de horas de trabalho que serão necessários para finalizar as atividades especificas com os recursos estimados. Para isso devem ser usadas as informações sobre atividades do escopo do projeto, tipos de recursos necessários, quantidades estimadas de recursos e calendário de recursos.

6.1 METODOLOGIA

A metodologia utilizada para o desenvolvimento desse trabalho iniciou-se pela etapa do levantamento de matérias bibliográfico para estudo através de pesquisas na biblioteca da Universidade do Extremo Sul Catarinense (UNESC), no município de Criciúma, e em artigos retirados na internet.

Após o levantamento do material necessário, foram estudadas metodologias de gerenciamento de projetos, para melhor entendimento do assunto em estudo. Em seguida foi retratado sobre Teoria da Corrente Critica e Teoria das Restrições para dar inicio ao documento.

O presente estudo atinge uma pesquisa aplicada. Onde tende como objetivo produzir conhecimentos para função prática aplicado à saída de problemas específicos, onde é abordada a implantação da Corrente Crítica em ambiente multiprojeto. Assim sendo, o estudo e proposta do método, não se

preocupa em relatar a natureza dos inconvenientes presentes na execução ou nas relações causais que importunam no uso da Corrente Crítica em um ambiente multiprojeto.

Nesse fundamento, este trabalho, pelo propósito que se propôs, se determina pela utilidade de uma abordagem quantitativa. Finalizado o estudo, foi iniciado o acompanhamento do trabalho, contemplando o conhecimento obtido através dos estudos realizados.

6.2 RESULTADOS, ANÁLISES E DISCUSSÕES

Este capítulo apresenta a estruturação da pesquisa, o método planejado, os dados referentes ao projeto analisado e o desempenho do modelo no contexto do projeto avaliado.

6.3 APLICAÇÃO DO MODELO DE PROCESSO PROPOSTO

O projeto proposto objetiva avaliar o modelo definido nesta pesquisa com a aplicação do mesmo em um projeto de desenvolvimento de software real. Neste contexto, foi estabelecida uma parceria com a Empresa CONTARE TECNOLOGIA a qual forneceu os dados necessários para validação do modelo de processo proposto, aplicando-o em um de seus projetos de desenvolvimento.

Inicialmente observou-se que a empresa em questão utiliza um modelo de desenvolvimento ágil, separando seus projetos em sprints semanais, facilitado assim o micro-gerenciamento diário das atividades do projeto. Desta forma, a aplicação da Teoria da Corrente Crítica, pode ser medida de forma sistemática, apontando os desvios do planejamento em relação a execução, fazendo com que o curso do projeto possa ser corrigido dentro do tempo planejado.

Para aplicação das condições observadas no desenvolvimento da pesquisa, foram seguidas as etapas listadas abaixo:

a) **Início:** nesta etapa são planejados os procedimentos internos da organização do projeto, dados de entrada, análise das

funcionalidades e definição dos fatores de incerteza, que, neste projeto, foram adotados de acordo com a regra de pontuação da escala de Fibonacci, sendo que quanto mais complexa a funcionalidade, maior é o fator de incerteza associado a ela;

b) **Planejamento:** para a definição do escopo de cada sprint do projeto, foram utilizadas duas técnicas; a primeira consiste na realização de reuniões de planejamento denominadas *Sprint Planing Meeting*, onde é feita uma análise crítica do projeto com a participação de todo o time de desenvolvimento. A segunda etapa refere-se ao refatoramento das funcionalidades com o cliente ou *owner* do projeto. Nesta etapa, são realizados procedimentos de análise de requisitos e priorização, apontando um planejamento com entregáveis bem definidos. Após estas etapas, é composta a EAP (Estrutura Analítica do Projeto), separada em Sprints de 5 dias de desenvolvimento, onde, ao final do quinto dia útil sempre é acompanhado um Marco ou (Milestone) de entrega;

c) **Execução:** esta etapa consiste em executar e acompanhar as atividades planejadas em cada *sprint*. Para a validação do modelo proposto, são utilizadas reuniões diárias de 15 minutos denominadas *daily scrum*, onde é atualizado o esforço empregado em cada funcionalidade;

d) **Controle:** o controle é feito através da atualização de um gráfico que mostra o esforço planejado e executado, estabelecendo indicadores de corrente crítica como gatilhos para o gerenciamento do consumo do pulmão do projeto;

e) **Encerramento**: nesta etapa é realizada uma reunião de encerramento onde são discutidos os fatores críticos do projeto, bem como as oportunidades de melhoria para a próxima sprint.

Para apoiar a gestão deste projeto, foi utilizada a ajuda do gestor do projeto bem como o time de desenvolvimento que, com isso, pode também validar um novo método de gerenciamento e controle para seus projetos em tempo de produção.

Por se tratar de uma pesquisa aplicada em um projeto real, algumas informações relevantes a empresa parceira foi omitida da pesquisa por serem consideradas confidenciais, não podendo ser divulgadas. Desta forma foram analisados somente os dados pertinentes a conclusão da pesquisa científica. Contudo, o trabalho será apresentado na sequência proposta contemplando a fase de Início, Planejamento. Execução, Controle e Encerramento.

6.4 ETAPA DE INÍCIO

Conforme mencionada anteriormente, a empresa em questão utiliza um modelo de processo de desenvolvimento híbrido, implementando conceitos do *scrum* associados com modelos tradicionais de documentação e processo. Desta forma utilizamos os artefactos de entrada sugeridos pela equipe de desenvolvimento do projeto contare. Além disso, foi criada a tabela contendo a escala de mensuração dos atributos relacionados a complexidade e incerteza desenvolvido por Pinto (2012). Esta tabela é aplicada com a intenção de estabelecermos uma métrica que auxilie na definição das prioridades, associada a solicitação do *Owner* (Cliente), servindo assim como ponto de referência para a realização das estimativas de tempo das atividades planejadas.

Os dados obtidos com a aplicação desta escala são apresentados abaixo:

Apresentação do mapeamento de complexidade e incerteza

Tabela 1 - Mapeamento do esforço por complexidade e incerteza

Sprint	Complexidade		Incerteza	
	Esforço	Classificação	Esforço	Classificação
SP01	36.5	Baixa	45.00	Média
SP02	9.00	Baixa	20.00	Média
SP03	20.00	Média	40.00	Alta
SP04	35.00	Alta	20.00	Média
SP06	30.00	Alta	35.00	Alta
SP07	40.00	Baixa	45.00	Média
SP08	25.00	Média	28.00	Média
SP09	25.00	Baixa	30.00	Média
SP10	20.00	Baixa	30.00	Média

SP11	25.00	Alta	30.00	Alta
SP12	25.00	Alta	28.00	Alta
SP13	30.00	Alta	35.00	Alta

Fonte: Do autor (2017).

A classificação das sprints do projeto por incerteza e complexidade, auxilia o gerente do projeto ou *Scrum* Master na classificação e ordenação das sprints. Alguns fatores externos devem ser considerados nesta fase, um deles é a classificação pela priorização do *Owner*, que, geralmente absorve demandas externas do mercado.

6.5 ETAPA DE PLANEJAMENTO

Nesta etapa, os requisitos do projeto são planejados e as sprints são compostas de acordo com a prioridade e complexidade, levando em consideração a velocidade do time. Além das atividades relacionadas ao desenvolvimento de novas funcionalidades, também são incluídas tarefas relacionadas a melhorias, bugs e suporte. O projeto analisado nesta fase apresenta o seguinte *backlog*:

Figura 5 - Tarefas Relacionadas

Fonte: Do autor (2017)

De acordo com a tabela acima, nesta fase foram planejadas 13 Sprints com um conjunto de tarefas priorizando o tamanho de acordo com

o esforço alocado para o projeto Contarte. A figura abaixo apresenta a lista de Sprints com a quantidade de atividades planejadas e finalizadas.

Figura 6 - Sprints com a quantidade de atividades planejadas e finalizadas

Versão

	Abertas	Fechadas	Total
Sprint 1	-	13	13
Sprint 2	-	6	6
Sprint 3	-	3	3
Sprint 4	-	7	7
Sprint 5	-	2	2
Sprint 6	-	4	4
Sprint 7	-	5	5
Sprint 8	-	5	5
Sprint 9	-	11	11
Sprint 10	-	5	5
Sprint 11	-	6	6
Sprint 12	-	11	11
Sprint 13	-	9	9

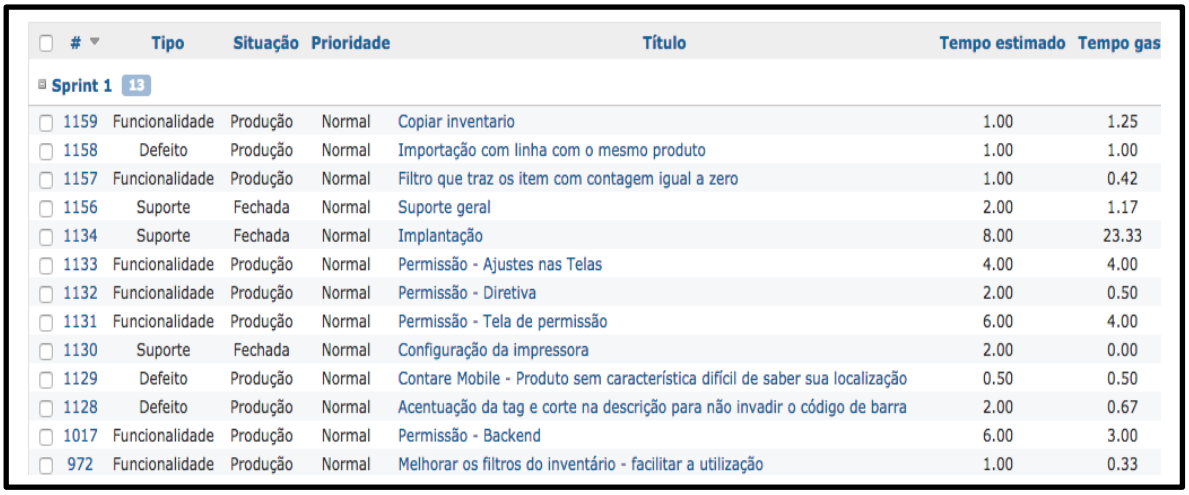
Fonte: Do autor (2017)

De acordo com o planejamento proposto, foram executadas as 13 Sprints com um planejamento de 5 dias para cada Sprint. Uma visão detalhada das sprints pode ser observada no subcapítulo da etapa de execução.

6.6 ETAPA DE EXECUÇÃO E CONTROLE

Nesta etapa, as Sprints planejadas são executadas e o controle é feito utilizando a teoria da corrente crítica. O fator base deste processo é a observação do consumo do pulmão. Neste contexto, associamos cada sprint a um micro projeto, sendo ele gerenciado e controlado com a teoria básica da corrente crítica abordada neste trabalho. Para melhor visualizarmos o trabalho executado, é importante apresentarmos o planejamento de cada sprint. Estes dados podem ser observados abaixo:

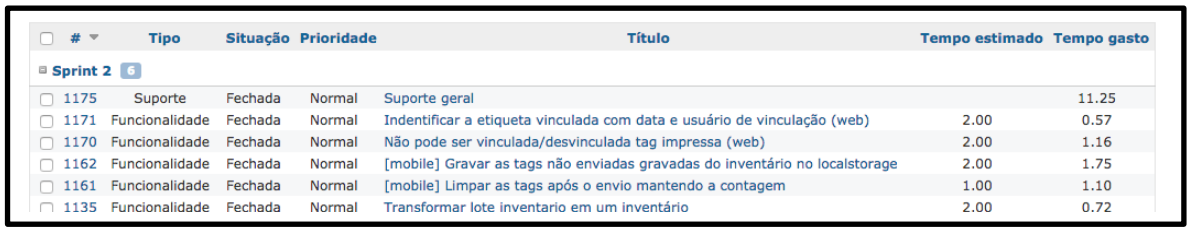
Figura 7 - Projeto Contare Sprint 1.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gas
Sprint 1 13						
1159	Funcionalidade	Produção	Normal	Copiar inventario	1.00	1.25
1158	Defeito	Produção	Normal	Importação com linha com o mesmo produto	1.00	1.00
1157	Funcionalidade	Produção	Normal	Filtro que traz os item com contagem igual a zero	1.00	0.42
1156	Suporte	Fechada	Normal	Suporte geral	2.00	1.17
1134	Suporte	Fechada	Normal	Implantação	8.00	23.33
1133	Funcionalidade	Produção	Normal	Permissão - Ajustes nas Telas	4.00	4.00
1132	Funcionalidade	Produção	Normal	Permissão - Diretiva	2.00	0.50
1131	Funcionalidade	Produção	Normal	Permissão - Tela de permissão	6.00	4.00
1130	Suporte	Fechada	Normal	Configuração da impressora	2.00	0.00
1129	Defeito	Produção	Normal	Contare Mobile - Produto sem característica difícil de saber sua localização	0.50	0.50
1128	Defeito	Produção	Normal	Acentuação da tag e corte na descrição para não invadir o código de barra	2.00	0.67
1017	Funcionalidade	Produção	Normal	Permissão - Backend	6.00	3.00
972	Funcionalidade	Produção	Normal	Melhorar os filtros do inventário - facilitar a utilização	1.00	0.33

Fonte: Do autor (2017)

Figura 8 - Projeto Contare - Sprint 2.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 2 6						
1175	Suporte	Fechada	Normal	Suporte geral		11.25
1171	Funcionalidade	Fechada	Normal	Indentificar a etiqueta vinculada com data e usuário de vinculação (web)	2.00	0.57
1170	Funcionalidade	Fechada	Normal	Não pode ser vinculada/desvinculada tag impressa (web)	2.00	1.16
1162	Funcionalidade	Fechada	Normal	[mobile] Gravar as tags não enviadas gravadas do inventário no localstorage	2.00	1.75
1161	Funcionalidade	Fechada	Normal	[mobile] Limpar as tags após o envio mantendo a contagem	1.00	1.10
1135	Funcionalidade	Fechada	Normal	Transformar lote inventario em um inventário	2.00	0.72

Fonte: Do autor (2017)

Figura 9 - Projeto Contare - Sprint 3.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Spri Ordenar por "#"						
1191	Defeito	Produção	Normal	Erro cor database la moda		5.00
1190	Funcionalidade	Produção	Normal	Suporte Geral	4.00	9.50
1167	Funcionalidade	Produção	Normal	[mobile] Melhorias integração com o leitor	16.00	19.50

Fonte: Do autor (2017)

Figura 10 - Projeto Contare - Sprint 4.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gast
Sprint 6 4						
1289	Funcionalidade	Fechada	Normal	Alterar inventário de nota ser bloqueado para produtos externos	0.33	0.33
1287	Suporte	Fechada	Normal	[Suporte] Suporte geral	8.00	9.00
1282	Funcionalidade	Fechada	Normal	Disponibilizar tela de configuração	4.00	4.50
1166	Funcionalidade	Fechada	Normal	[agente] Desenvolver calibração automática	16.00	10.00

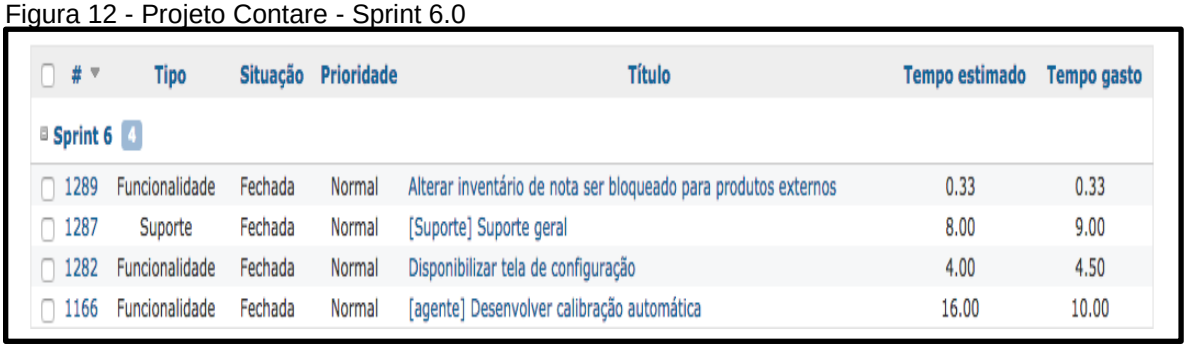
Fonte: Do autor (2017)

Figura 11 - Projeto Contare - Sprint 5.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 5 2						
1254	Funcionalidade	Produção	Normal	Suporte geral	3.00	3.00
1173	Funcionalidade	Produção	Normal	[MOBILE] Homologar o leitor BTL1000 (web e mobile)	24.00	26.50

Fonte: Do autor (2017)

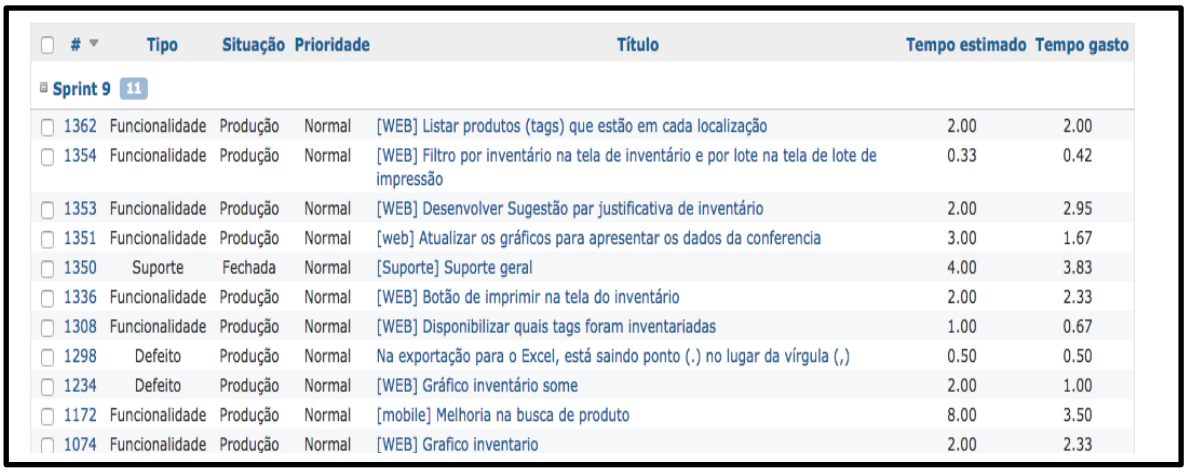
Figura 12 - Projeto Contare - Sprint 6.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 6 4						
1289	Funcionalidade	Fechada	Normal	Alterar inventário de nota ser bloqueado para produtos externos	0.33	0.33
1287	Suporte	Fechada	Normal	[Suporte] Suporte geral	8.00	9.00
1282	Funcionalidade	Fechada	Normal	Disponibilizar tela de configuração	4.00	4.50
1166	Funcionalidade	Fechada	Normal	[agente] Desenvolver calibração automática	16.00	10.00

Fonte: Do autor (2017)

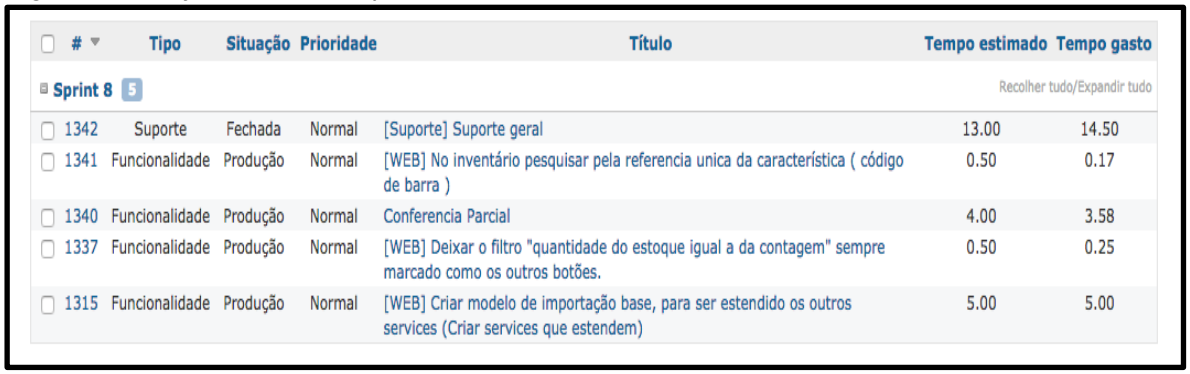
Figura 13 - Projeto Contare - Sprint 7.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 9 11						
1362	Funcionalidade	Produção	Normal	[WEB] Listar produtos (tags) que estão em cada localização	2.00	2.00
1354	Funcionalidade	Produção	Normal	[WEB] Filtro por inventário na tela de inventário e por lote na tela de lote de impressão	0.33	0.42
1353	Funcionalidade	Produção	Normal	[WEB] Desenvolver Sugestão par justificativa de inventário	2.00	2.95
1351	Funcionalidade	Produção	Normal	[web] Atualizar os gráficos para apresentar os dados da conferencia	3.00	1.67
1350	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	3.83
1336	Funcionalidade	Produção	Normal	[WEB] Botão de imprimir na tela do inventário	2.00	2.33
1308	Funcionalidade	Produção	Normal	[WEB] Disponibilizar quais tags foram inventariadas	1.00	0.67
1298	Defeito	Produção	Normal	Na exportação para o Excel, está saindo ponto (.) no lugar da vírgula (,)	0.50	0.50
1234	Defeito	Produção	Normal	[WEB] Gráfico inventário some	2.00	1.00
1172	Funcionalidade	Produção	Normal	[mobile] Melhoria na busca de produto	8.00	3.50
1074	Funcionalidade	Produção	Normal	[WEB] Grafico inventario	2.00	2.33

Fonte: Do autor (2017)

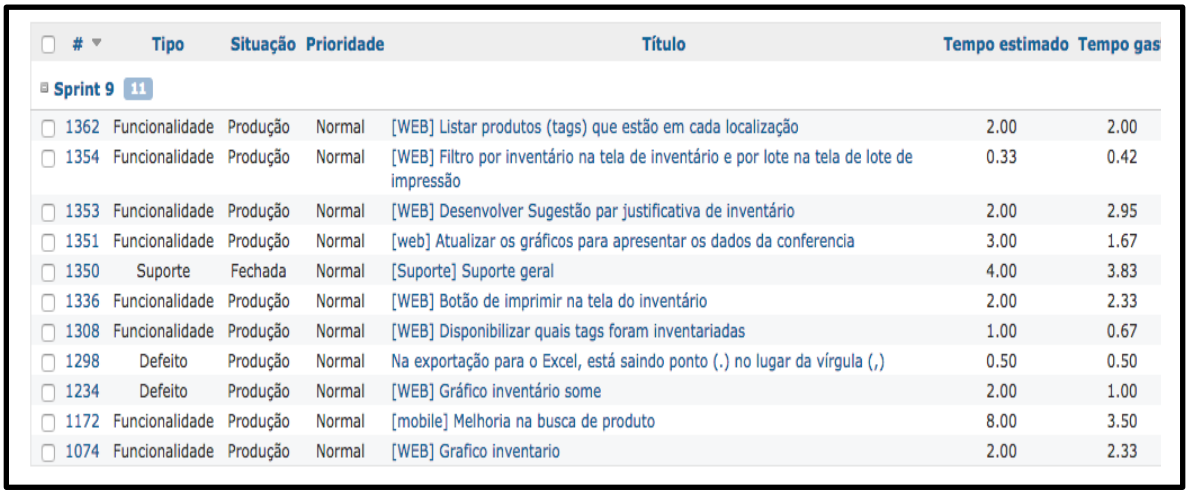
Figura 14 - Projeto Contare - Sprint 8.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 8 5						Recolher tudo/Expandir tudo
1342	Suporte	Fechada	Normal	[Suporte] Suporte geral	13.00	14.50
1341	Funcionalidade	Produção	Normal	[WEB] No inventário pesquisar pela referencia unica da característica (código de barra)	0.50	0.17
1340	Funcionalidade	Produção	Normal	Conferencia Parcial	4.00	3.58
1337	Funcionalidade	Produção	Normal	[WEB] Deixar o filtro "quantidade do estoque igual a da contagem" sempre marcado como os outros botões.	0.50	0.25
1315	Funcionalidade	Produção	Normal	[WEB] Criar modelo de importação base, para ser estendido os outros services (Criar services que estendem)	5.00	5.00

Fonte: Do autor (2017)

Figura 15 - Projeto Contare - Sprint 9.0



#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gas
Sprint 9 11						
1362	Funcionalidade	Produção	Normal	[WEB] Listar produtos (tags) que estão em cada localização	2.00	2.00
1354	Funcionalidade	Produção	Normal	[WEB] Filtro por inventário na tela de inventário e por lote na tela de lote de impressão	0.33	0.42
1353	Funcionalidade	Produção	Normal	[WEB] Desenvolver Sugestão par justificativa de inventário	2.00	2.95
1351	Funcionalidade	Produção	Normal	[web] Atualizar os gráficos para apresentar os dados da conferencia	3.00	1.67
1350	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	3.83
1336	Funcionalidade	Produção	Normal	[WEB] Botão de imprimir na tela do inventário	2.00	2.33
1308	Funcionalidade	Produção	Normal	[WEB] Disponibilizar quais tags foram inventariadas	1.00	0.67
1298	Defeito	Produção	Normal	Na exportação para o Excel, está saindo ponto (.) no lugar da vírgula (,)	0.50	0.50
1234	Defeito	Produção	Normal	[WEB] Gráfico inventário some	2.00	1.00
1172	Funcionalidade	Produção	Normal	[mobile] Melhoria na busca de produto	8.00	3.50
1074	Funcionalidade	Produção	Normal	[WEB] Grafico inventario	2.00	2.33

Fonte: Do autor (2017)

Figura 16 - Projeto Contare - Sprint 10.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 10 5						
1384	Funcionalidade	Produção	Normal	[WEB] Codificação renner	3.00	3.00
1367	Defeito	Produção	Urgente	[WEB] Alterar inventario, lote impressão, gestão de localização para ser por unidade	2.00	2.28
1365	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	8.00
1364	Funcionalidade	Produção	Normal	[WEB] Cadastro de sugestões para inventário	4.00	7.58
1352	Funcionalidade	Produção	Normal	[WEB] Criar comparativo de inventário	5.00	9.50

Fonte: Do autor (2017)

Figura 17 - Projeto Contare - Sprint 11.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 13 9						
1420	Suporte	Fechada	Normal	[SUPORTE] REMOVER TESTES FLOR LINDA	1.00	1.00
1419	Funcionalidade	Produção	Normal	[web] União de lote de inventário para gerar inventário	5.00	5.17
1418	Funcionalidade	Produção	Normal	Alteração na tag da Flor Linda	1.00	1.17
1415	Funcionalidade	Produção	Normal	[WEB] Criar importação de arquivo de inventário SMART	5.00	4.50
1414	Suporte	Fechada	Normal	[Suporte] Implantação Flor Linda	6.00	4.83
1413	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	6.17
1375	Defeito	Produção	Alta	[MOBILE] LEITOR 900 ESTÁ COM A LEITURA RUIM	5.00	1.00
1366	Defeito	Produção	Alta	Atualização do Aplicativo Automaticamente	2.00	1.58
1309	Funcionalidade	Produção	Normal	[Mobile] botão de Bluetooth	2.00	0.67

Fonte: Do autor (2017)

Figura 18 - Projeto Contare - Sprint 12.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 12 11						
1409	Funcionalidade	Produção	Normal	[WEB] Desenvolver leitura de arquivo de etiquetas da SMART	5.00	5.00
1408	Suporte	Fechada	Normal	[SUPORTE] Escrita do memorial descritivo do sistema contare	1.50	1.67
1407	Funcionalidade	Produção	Normal	[IMPRESSORA] fazer impressora ZD500 Series Imprimir	2.00	1.83
1406	Suporte	Fechada	Normal	[SUPORTE] Ligação para flor linda	2.00	1.78
1405	Defeito	Produção	Normal	[WEB] Tela de conferencia não deixa salvar, e não é possivel clicar no campo observação	1.00	0.50
1404	Funcionalidade	Produção	Normal	Criar designer tag Renner	1.00	2.50
1403	Funcionalidade	Produção	Normal	[WEB] Disponibilizar ambiente de teste Renner e efetuar a impressão de tags de teste	1.00	0.50
1402	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	8.00
1401	Funcionalidade	Produção	Normal	[WEB] Desenvolver parametro para Tyco serial	3.00	5.50
1400	Funcionalidade	Produção	Normal	[WEB] Arrumar parâmetros contare	2.00	1.00
1325	Defeito	Produção	Normal	[AGENT IMPRESSÃO] no windows não aparece o logo da contare	1.00	0.50

Fonte: Do autor (2017)

Figura 19 - Projeto Contare - Sprint 13.0

#	Tipo	Situação	Prioridade	Título	Tempo estimado	Tempo gasto
Sprint 13 9						
1420	Suporte	Fechada	Normal	[SUPORTE] REMOVER TESTES FLOR LINDA	1.00	1.00
1419	Funcionalidade	Produção	Normal	[web] União de lote de inventário para gerar inventário	5.00	5.17
1418	Funcionalidade	Produção	Normal	Alteração na tag da Flor Linda	1.00	1.17
1415	Funcionalidade	Produção	Normal	[WEB] Criar importação de arquivo de inventário SMART	5.00	4.50
1414	Suporte	Fechada	Normal	[Suporte] Implantação Flor Linda	6.00	4.83
1413	Suporte	Fechada	Normal	[Suporte] Suporte geral	4.00	6.17
1375	Defeito	Produção	Alta	[MOBILE] LEITOR 900 ESTÁ COM A LEITURA RUIM	5.00	1.00
1366	Defeito	Produção	Alta	Atualização do Aplicativo Automaticamente	2.00	1.58
1309	Funcionalidade	Produção	Normal	[Mobile] botão de Bluetooth	2.00	0.67

Fonte: Do autor (2017)

De acordo com as trezes sprints avaliados na empresa Contare Tecnologia seis delas tiverem um atraso considerável, portanto, esse atraso serviu como reforço para que a equipe usada no projeto fosse melhorar nos próximos projetos. As outras sete sprints foram entregues dentro do prazo,

mesmo consumindo uma parte da margem de segurança que é feito através do cálculo das horas planejadas multiplicada por 20% definido pelo gerente de projeto, portanto, o projeto teve uma confiabilidade aceitavel. Nas sprints projeto Contare da primeira até a décima terceira a equipe executou da seguinte maneira: cada sprint foi definida tarefas a serem executadas, além das tarefas com seus respetivos códigos foi determinado o tipo, situação, prioridade o tempo estimado e o tempo gasto a ser executado.

Com base no planejamento acima, foi executado o método de controle baseado em reuniões diárias, uma vez que este, é o método utilizado pela empresa para o acompanhamento de seus projetos. Desta forma, foram avaliadas as evoluções diárias sobre cada projeto em desenvolvimento e, ao final, podemos analisar os desvios de execução conforme tabela abaixo:

Utilização do pulmão do projeto CONTARE

Tabela 2 - Apresentação dos resultados

Projeto - Sprint	Esforço inicial (Horas)	Duração sem o pulmão (Horas)	Pulmão - 20% (Horas)	Duração total (Horas)
SP01	40.00	32.00	8.00	40.17
SP02	10.00	8.00	2.00	16.55
SP03	20.00	16.00	4.00	34.00
SP04	35.00	28.00	7.00	29.50
SP05	30.00	24.00	6.00	29.50
SP06	40.00	32.00	8.00	23.83
SP07	25.00	20.00	5.00	33.67
SP08	25.00	20.00	5.00	23.50
SP09	20.00	16.00	4.00	21.20
SP10	25.00	20.00	5.00	30.37
SP11	25.00	20.00	5.00	22.00
SP12	30.00	24.00	6.00	28.78
SP13	30.00	24.00	6.00	26.08

Fonte: Do autor(2017)

A tabela acima apresenta uma visão geral sobre o andamento do projeto, mostrando um panorama referente a Sprint, o esforço inicial planejado em horas, a duração do projeto sem o pulmão, o pulmão alocado e a duração total de cada sprint.

Algumas informações importantes são relevantes para análise dos resultados. Uma delas é o Percentual utilizado para a estimativa do pulmão. Seguindo as orientações de Leach (2000), basicamente calculamos 20 % a partir do tempo estimado para cada Sprint. A análise crítica deste projeto pode ser observada na etapa de conclusão.

6.7 ETAPA DE CONCLUSÃO

Nesta etapa, são analisados os fatos importantes ocorridos durante o projeto, bem como as oportunidades de melhoria. Desta forma, apresentamos neste item, uma análise dos micro Projetos abordados neste estudo. Conforme observado na Tabela 2, podemos avaliar que as Sprints 01,02,03,07,09 e 10 entraram na corrente crítica e consumiram o pulmão do projeto, extrapolando o tempo planejado. Porém, a data planejada para a liberação do projeto foi mantida, fator este que propõe que o acompanhamento dos projetos utilizando a corrente crítica proporciona uma visão importante para o gestor que consegue tomar decisões referentes aos desvios, de forma rápida podendo replanejar atividades a fim de manter a *release*.

Outro fator importante que pode ser observado, é que após a execução de 3 sprints utilizando o modelo de gestão pela corrente crítica, houve um ganho na forma de estimativa, a qual proporcionou que as 3 próximas sprints, embora tenha havido consumo do pulmão, foram entregues abaixo do tempo estimado. Observamos uma tendência após a execução histórica utilizando o método, pois o aprendizado absorvido pela equipe, proporciona um ganho no sentido de estimativa e execução do projeto tanto no que se refere a estimativas iniciais quanto na entrega.

7 CONCLUSÃO

Com base nesse assunto é extremamente importante que se entenda que nem sempre os riscos que um projeto tem são levados em consideração, ou seja, podendo assim comprometer o sucesso do projeto. Portanto o gerenciamento de projetos tem a finalidade de precaver juntamente com sua equipe, em uma visão abrangente com facilidade na identificação dos riscos.

A implementação da metodologia da Corrente Crítica vem ganhando cada vez mais adeptos, com muitos artigos publicados hoje em dia as organizações não se vem mais nos formatos tradicionais de gerenciar projetos.

Com o conhecimento de uma nova metodologia, a Corrente Crítica não é considerada complexa, porém seu raciocínio lógico exige muita observação, tendo como promessas reduzir prazos gerando menos caos nas equipes executoras e ganho de credibilidade através dos orçamentos.

Essa nova linha de pensamento pode estrategicamente significar a diferença entre o sucesso e o fracasso de um projeto.

Pode-se concluir então que na figura dois (2) teve sprints que entraram em corrente crítica com o uso do pulmão, portanto, a data para a liberação do projeto foi mantida; ou seja é notável que o uso da teoria da corrente crítica proporciona um ganho no sentido da estimativa de tempo e execução do projeto.

REFERÊNCIAS

AZANHA, Raphael Barban; ANTONIOLLI, Pedro Domingos; BENEVIDES, Gustavo; NEVES, Angela Fernanda Naves; JUNIOR, Adonival Coelho de Souza. Aplicação da abordagem corrente crítica da teoria das restrições em projetos de máquinas. **Revista de Administração do Sul do Pará (REASP) – FESAR** – v. 1, n. 3, Set/Dez – 2014. Disponível em <http://www:http://reasp.fesar.com.br/index.php/REASP/article/view/30/24. Acesso em 25 mar. 2016.

BECK, Kent. **Programação Extrema Explicada.** Bookman, 2004 Http://www.cin.ufpe.br/~gamr/FAFICA/Desenvolvimento%20de%20sistemas/XP.pdf .acesso em 4 de mai. 2016

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

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

BRANDÃO, Barbara Parreira; NASCIMENTO, Gracielly Ribeiro do; ABADIA, Ludymilla Santos de; PACO, Tatiany da Rocha; REZENDE, Ricardo Caetano. Teoria das restrições aplicada a uma linha de produção de ketchup.**ABEPRO.** Bento Gonçalves, RS, outubro de 2012. Disponível em <http://www:abepro.org.br/biblioteca/enegep2010_TN_STO_113_739_15576.pdf. Acesso em 29 mar. 2016.

CARVALHO, Bernardo Vasconcelos de; MELLO, Carlos Henrique Pereira. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica.**Gest. Prod.**, São Carlos , v. 19, n. 3, p. 557-573, 2012 . Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000300009&lng=pt&nrm=iso>. acessos em 26 abr. 2016.

CLELAND, David I.; IRELAND, Lewis R. **Gerenciamento de projetos.** 2. ed Rio de Janeiro: LTC, 2007. xii, 371 p.

GOLDRATT, E. M.; **A Corrente Crítica.** São Paulo: Nobel. 1998

GOLDRATT, Eliyahu. **Corrente Crítica**. São Paulo: Ed. Nobel. 1997.

LEACH, L. P. **Critical Chain Project Management**. Norwood, Artech House, Inc., 330 p. 2000.

MARCANTONIO, Maria Isabel Palmerio.A corrente critica aplicada na ferramenta de gestão de projetos ms project. **ABEPRO.** São Carlos, SP, outubro de 2010.Disponível em

<http://www:abepro.org.br/biblioteca/enegep2010_TN_STO_113_739_15576.pdf. Acesso em 25 mar. 2016.

MARTINS, José Carlos Cordeiro. **Gerenciamento projetos de desenvolvimento de software com PMI, RUP e UML**. Ed. 5. Rio de Janeiro: Brasport, 2010.

NERUR S; MAHAPATRA, R; MANGALARA, G. **Challenges of Migratingto Agile Methodologies**. Communications of the ACM, v.48, n.5, Maio/2005.

NOGUCHI, Julio Celso. Corrente crítica – a teoria das restrições aplicada à gestão de projetos. **Revistado Centro Universitário Planalto do DistritoFederal – UNIPLAN**. Volume 3 No 1, 2006. Disponível em http://www.uniplandf.edu.br/revista/vol3n1.pdf#page=20.Acesso em 31 mar. 2016.

ORDOÑEZ, Robert Eduardo Cooper. Proposta para uso da corrente crítica no gerenciamento de múltiplos projetos. **UNICAMP** ,Campinas, SP. 2013. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?view=000911341> Acesso em 05 mar 2016.

PINTO, J. S. Variáveis dos Atributos Complexidade e Incerteza em Projetos: Proposta de Criação de Escala de Mensuração. Universidade Estadual de Campinas, Campinas, SP, **Tese de Doutorado**, 216 p, 2012.

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

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®),**Pennsylvania, 2008, ed 4, 459p.

QUINTELLA, Heitor Luiz Murat de Meirelles; ROCHA, Henrique Martins. Avaliação da maturidade do processo de desenvolvimento de veículos automotivos. **Gest. Prod.**, São Carlos , v. 13, n. 2, p. 297-310, May 2006 . Disponivel em : <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2006000200011&lng=en&nrm=iso>. acessado em 26 June 2017.

QUELHAS, Osvaldo; BARCAUI, André B. A teoria das restrições aplicada a gerência de projeto: uma introdução a corrente crítica. **UFF passos da Pátria,** RJ, 2010. Disponível em <http://www:pmtech.com.br/newsletter/Marco_2005/TOC_e_CCPM_em_GP.pdf . Acesso em 25 mar. 2016.

REHMAN, Asim Ur. Software **Project Management Methodologies/Frameworks Dynamics "A Comparative Approach**". In

Proceedings of the IEEE International Conference on Information and Emerging Technologies (ICIET), 2007.

RUP. **Rational Unified Process.** Version 2003.06.00.65, CD-ROM. Rational Software Corporation, Cupertino, California.

SAMARANI, Paulo R. de M..**Um modelo de implementação do Capabilit Maturity Integration nível 2**. 2005. 135f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porta Alegre. Disponível em: <http://www.lume.ufrgs.br/handle/10183/5698>. Acesso em: 18 abril de 2016.

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

SILVEIRA, Dennis Williams Alves da. **Ferramenta para apoio à estimativa baseada em Planning Poker utilizando a metodologia Scrum**. 2012.Trabalho de Conclusão de Curso (Graduação em ciência da computaçãoo) - UNIVERSIDADE FEDERAL DE PERNAMBUCO, Recife. Disponivel em: < http://www.cin.ufpe.br/~tg/2012-1/dwas.pdf>. Acesso em: 27 de jun de 2017.

SILVA, Éverton Maurer da; RODRIGUES, Luis Henrique; LACERDA, Daniel Pacheco. Aplicabilidade da corrente crítica da teoria das estrições no gerenciamento de projetos executivos de engenharia: um estudo de caso em uma refinaria de petróleo. **Gest. Prod.**, São Carlos , v. 19, n. 1, p. 1-16, 2012 . Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000100001&lng=pt&nrm=iso>. acessos em 22 mar. 2016.

SOFTEX. MPS.BR - Melhoria de Processo do Software Brasileiro. **Copyright ©,** Dezembro de 2012. Disponível em <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf> Acessos em 15 mai. 2016.

TUKEL, O..; ROM, W.; EKSIOGLU, S. An investigation of buffer sizing techniques in Critical Chain Scheduling. **EuropeanJournalofOperationalResearch**, v. 172, p. 401-416, 2006.

XAVIER, Carlos Magno da Silva. **Gerenciamento de projetos:** como definir e controlar o escopo do projeto. São Paulo: Saraiva, 2008. 259 p.

APÊNDICE A – ARTIGO

Teoria da corrente crítica para minimizar o problema de gerenciamentos de projetos de Softwares

Edson de Oliveira Branco[2], Gustavo Bisognin[2]

[1]Acadêmico do Curso de Ciência da Computação

Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brasil

[2]Professor do Curso de Ciência da Computação

Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brasil

oliveira200_6@yahoo.com, gbisog@gmail.com

Abstract. The theory of the critical current, originated by Eliyahu Goldratt in 1997, with the evolution of the business and of the companies coming from a globalized world Goldratt noted three problems for management of projects, Parkinson's law, the Syndrome of the Student and the Multitasking. Thus, some adaptation of the knowledge, skills, and techniques facilitates a better vision of the results to be delivered within the criteria of time, budget, scope and quality. The theory of the current critique addresses a different way to treat the project management in the context of software. The theory of constraints seeks the optimization of the performance of the activities of the project; models of software processes tends to make the projects more manageable, CMMI and MPS.BR quality models; currently we highlight the models of project management, the Scrum model, the model Xtreme programming (XP) and traditional models like the RUP and the PMBOK. Generally, the current critique is a model for project management that your focus is primarily on the administration of deadlines and activities, considering resource allocation based on the TOC. In this context, the present research seeks to present a better technique of time management in projects is considered as one of the most innovative advances in the area of project management; for being an applied research in a real project, where they were analyzed only the data relevant to the completion of the search, so the proposal includes the stage of initiation, planning, execution, control and closing.

Resumo. A teoria da corrente crítica originou-se por Eliyahu Goldratt em 1997, com a evolução dos negócios e das empresas vindo de um mundo globalizado Goldratt observou três problemas para gestão de projetos, a lei de Parkinson, a Síndrome de Estudante e a Multitarefa. Assim a certa a adaptação de conhecimento, habilidades e técnicas facilita a melhor visão dos resultados a serem entregues dentro dos critérios de tempo, orçamento, escopo e qualidade. A teoria da corrente critica aborda uma forma diferente de tratar o gerenciamento de projeto no contexto de software. A teoria das restrições busca a otimização dos desempenhos das atividades do projeto; os modelos de processos de software tende a tornar os projetos mais gerenciáveis, CMMI e MPS.BR modelos de qualidade; atualmente destaca-se os modelos de gerenciamento de projetos,

o modelo Scrum, o modelo *Xtreme programming* (XP) e modelos tradicionais como o RUP e PMBOK. De modo geral a corrente critica é um modelo para gerenciamento de projetos que seu foco é basicamente na administração de prazos e atividades, considerando alocação de recursos baseado na TOC. Nesse contexto a presente pesquisa busca apresentar uma melhor técnica de gerenciamento do tempo em projetos considerada como um dos mais inovadores avanços na área de gerenciamento de projetos; por se tratar de uma pesquisa aplicada em um projeto real, onde foram analisados somente os dados pertinentes para a conclusão da pesquisa, assim a proposta contempla a fase de início, planejamento, execução, controle e encerramento.

1. INTRODUÇÃO

Com base em estudos históricos, identificou-se que a teoria da corrente crítica nasceu nos anos de 1997, tal técnica, apoia a constante busca por produtividade associada a uma redução nos custos de gerenciamento de projetos de engenharia de software. Esta técnica, foi escrita por Eliyahu M. Goldratt, e é responsável pela criação de novos conceitos aplicados ao gerenciamento de projetos e suas respectivas atividades e tarefas.

O gerenciamento de projeto existe a alguns anos, e vem sendo exercido em vários setores, no qual, práticas têm sido divulgadas e documentadas a um período de tempo, contribuindo assim para evolução dos projetos uma vez que abordam conceitos claros e ágeis em seu gerenciamento (CLELAND; IRELAND, 2007). Os projetos possuem distintas dimensões a serem gerenciadas, a organização é uma das peças importante para desenvolvimento de produtos, contudo, o gerenciamento de projetos busca a redução de custo, prazo e entregas para o cliente final (SILVA; RODRIGUES; LACERDA, 2012).

A Teoria das Restrições visa o tratamento das restrições de um projeto, sejam elas internas ou externas ou de mercado. Tal teoria busca a otimização do desempenho das atividades do projeto, aumentando a probabilidade de entrega no prazo definido. Neste contexto, identifica-se que a teoria das restrições pode ser aplicada em diversas áreas de conhecimento, como projetos de engenharia civil, mecânica, elétrica e principalmente na engenharia de software.

No entanto a teoria da corrente crítica aborda uma forma diferenciada de tratar o gerenciamento de projetos no contexto de software. Fortemente baseada na teoria das restrições, a corrente crítica enfatiza o gerenciamento das atividades do projeto e objetiva a aplicação de metodologia agressiva no que se refere as estimativas de tempo, diminuindo consideravelmente o tempo estimado para a realização de uma determinada atividade do projeto.

2. MODELOS DE PROCESSO DE SOFTWARE

De modo que as ações de melhorias possam ser desligadas, as empresas que precisam melhorar seus processos precisam adotar-se de normas e modelos capazes de arriscar possíveis padronizações afim de conseguir melhores resultados que leva a uma melhoria contínua da qualidade do processo de desenvolvimento de software. Com isso alguns modelos e normas e qualidades de software vêm ganhando importância, seja por sua qualidade ou aplicação. O modelo de processo de software tem como objetivo melhorar a qualidade do sistema para tornar os projetos mais gerenciáveis, as datas de entregas e os custos mais previsíveis para que a equipe passa guiar e realizar o trabalho necessário para construir o sistema (PRESSMAN, 2006).

2.1 Modelos de Maturidade

Segundo Quintella e Rocha (2006), a definição de níveis de maturidade é baseada em princípios de qualidade de produtos, de acordo com os autores Crosby adaptou a estrutura de maturidade da qual esses princípios foram mostrados como “Aferidor de Maturidade da Gerência de Qualidade”, o qual citou os cinco estágios na adoção das práticas de qualidade: Incerteza, Despertar, Esclarecimento, Sabedoria e Certeza. Esses Esses princípios foram adaptados pelo SEI, Software Engineering Institute, da Carnegie Mellon University, para o processo de desenvolvimento de software em 1986, pelo CMM (Capability Maturity Model).

2.1.1 Modelo CMMI

O modelo *Capability Maturity Model Integration* (CMMI) é um modelo de qualidade criado pelo *Software Engineering Institute*(SEI) que desenvolveu um abrangente metamodelo de processo que se dedica a um conjunto de forças de engenharia de software que devem estar presente à medida que as empresas atingem níveis de capacidade e maturidade de processo. Para alcançar essas capacidades, o SEI afirma que uma empresa deve construir um modelo de processo que siga as normas do CMMI (PRESSMAN, 2006). A ideia do CMMI é de abastecer-se no desenvolvimento e manutenção de produtos e serviços e também de uma estrutura extensível de modo que outros conhecimentos de corpo possam ser salvos (SAMARANI, 2005).

2.1.2 Modelo MPS

O MPS.BR é um programa mobilizador, de longo prazo, criado em dezembro de2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), o MPS.BR tem como objetivo melhorar o Processo de Software e Serviços, com duas metas a alcançar a médio e longo prazos, metas essas que são:

. a) meta técnica, que visa à criação e aprimoramento do Modelo MPS;

. b) meta de negócio, que visa à disseminação e adoção do Modelo MPS em todo país, em um intervalo de tempo justo, a um custo razoável, tanto em micro, pequenas e médias empresas.

.

3. GERENCIAMENTO DE PROJETOS

A gerência de projetos é um ramo bastante usada em várias empresas de variados tipos de negócio. Com o tempo foram comprovando os benefícios que ela proporciona, trazendo como principal resultado a redução de custos, que vem se destacando de acordo com o tamanho dos projetos.

Os gerenciamentos de projetos são práticas que remontam a antiguidade, à medida que teoria e prática de gerenciamento de projeto se ampliam surgem novas áreas de conhecimento proporcionando assim alicerces de um futuro promissor (CLELAND; IRELAND, 2007).

Segundo o Guia PMBOK (2008), o gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto afim de atender aos seus requisitos. O mesmo é realizado através da aplicação e da interação dos seguintes processos de gerenciamento de projetos: iniciação, planejamento, execução, monitoramento, controle e encerramento.

3.1 MODELOS DE GERENCIAMENTO DE PROJETOS.

3.1.1 MODELOS ÁGEIS

Segundo Boehm (2006), enfatiza que o desenvolvimento de software tem despertado grandes interesses entre as organizações do mundo inteiro. Pois, gerenciamento ágil foi criado através dos conceitos de desenvolvimento rápido como uma das alternativas afim de poder tratar com ambiente competitivo do mercado atual que exige resultados imediatos, sob condições de grandes incertezas e constantes mudanças.

3.1.2 MODELO SCRUM

Criado por Ken Schwaber e Jeff Sutherland (1996), o modelo *Scrum* proposto por eles, defende que o desenvolvimento de software é imprevisível e que deve ser separado por se destacar dos demais métodos ágeis por este estar mais focado ao processo de gerenciamento de projetos.

Segundo Schwaber (2004), o modelo *Scrum* reúne atividades de monitoramento e *feedback*, fazendo reuniões rápidas e diárias com toda a equipe, visando na identificação e correção de quaisquer deficiências ou impedimentos no processo de desenvolvimento.

3.2 MODELOS TRADICIONAIS

3.2.1 RUP

O *Rational Unified Process* (RUP), é um processo proprietário de engenharia de software criado pela empresa *Rational Software Corporation* para apoiar desenvolvimentos orientados de objetos e fornecendo uma forma sistemática de obter vantagem no uso de UML, e ele foi posteriormente adquirida pela gerência de Projetos de Software com utilização de práticas tradicionais. Este processo descreve técnicas a serem seguidas pelos membros da equipe de desenvolvimento de software com o objetivo de aumentar a sua produtividade.

4. CORRENTE CRÍTICA

De modo geral, o método da corrente critica(*Critical Chain Project Management*), é um método para gerenciamento de projetos que, visa focar basicamente na administração de prazos e atividades, considerando alocação de recursos, baseado na Teoria das Restrições (TOC). A sua origem é relacionada com a publicação do livro intitulado Corrente Critica, escrita por Eliyahu M. Goldratt, teve seu início através de observações básicas de problemas comuns em projetos de alta probabilidade onde ultrapassava-se o orçamento, tempo e comprometimento de conteúdo.

De acordo com o Gordratt (1997) o principal problema no gerenciamento de projetos são as incertezas, elas originam dificuldades e obstáculos.

Existe três fenômenos que interferem nas execuções de atividades: primeiro síndrome de estudante que consiste em esperar até o ultimo momento para iniciar as atividades, estimativas de duração são baseadas em experiências pessimistas; segundo multitarefa consiste na consequência gerada pelo ambiente em que vários projetos são desenvolvidos paralelamente, quanto maior o níveis gerências maior a duração total de estimativa da atividade, assim os diferentes níveis adicionam segurança; terceiro lei de Parkinson consiste no principio que as atividades tendem a se

expandir ocupando o tempo disponível, as pessoas estimam as durações protegendo-se de cortes (GOLDRATT,1997).

4.1 TEORIA DAS RESTRIÇÕES

Na década dos anos 70 teve o começo da Teoria das Restrições (TOC) quando o Israelense Eliyahu Goldratt teve problemas com a logística de produção de uma organização.

Com isso Goldratt teve de elaborar um novo método de produção. Este mesmo método elaborado por ele, teve um ótimo desempenho no mercado e assim varias empresas tiveram o interesse de aplicar esta mesma técnica. A partir daí, Goldratt se focou em continuar o trabalho podendo assim espalha-lo.

A teoria das restrições (TOC) é definida como abordagem de gestão voltada para melhoria dos processos que limitam o fluxo da produção com intuito de melhorar continuamente o desempenho das produções, buscando assim a filosofia de melhoria por meio de identificação das restrições de um sistema (BRANDÃO et al, 2012).

A restrição de um sistema é qualquer coisa que impeça o sistema de atingir um desempenho maior que a sua meta, por este motivo é necessário conhecer a meta do sistema (GOLDRATT, 1997).

5. RESULTADOS E DISCUSSÕES

O projeto proposto objetiva avaliar o modelo definido nesta pesquisa com a aplicação do mesmo em um projeto de desenvolvimento de software real. Neste contexto, foi estabelecida uma parceria com a Empresa CONTARE TECNOLOGIA a qual forneceu os dados necessários para validação do modelo de processo proposto, aplicando-o em um de seus projetos de desenvolvimento.

Para aplicação das condições observadas no desenvolvimento da pesquisa, foram seguidas as etapas seguintes: Início, Planejamento, Execução,

Controle e Encerramento.

Tabela 1 - Mapeamento do esforço por complexidade e incerteza

Sprint	Complexidade		Incerteza	
	Esforço	Classificação	Esforço	Classificação
SP01	36.5	Baixa	45.00	Média
SP02	9.00	Baixa	20.00	Média
SP03	20.00	Média	40.00	Alta
SP04	35.00	Alta	20.00	Média
SP06	30.00	Alta	35.00	Alta
SP07	40.00	Baixa	45.00	Média
SP08	25.00	Média	28.00	Média
SP09	25.00	Baixa	30.00	Média
SP10	20.00	Baixa	30.00	Média
SP11	25.00	Alta	30.00	Alta
SP12	25.00	Alta	28.00	Alta
SP13	30.00	Alta	35.00	Alta

Fonte: Do autor (2017).

A classificação das sprints do projeto por incerteza e complexidade, auxilia o gerente do projeto ou *Scrum* Master na classificação e ordenação das sprints.

Tabela 2 - Apresentação dos resultados

Projeto - Sprint	**Esforço inicial (Horas)**	**Duração sem o pulmão (Horas)**	**Pulmão - 20% (Horas)**	**Duração total (Horas)**
SP01	40.00	32.00	8.00	40.17
SP02	10.00	8.00	2.00	16.55
SP03	20.00	16.00	4.00	34.00
SP04	35.00	28.00	7.00	29.50
SP05	30.00	24.00	6.00	29.50
SP06	40.00	32.00	8.00	23.83
SP07	25.00	20.00	5.00	33.67
SP08	25.00	20.00	5.00	23.50
SP09	20.00	16.00	4.00	21.20
SP10	25.00	20.00	5.00	30.37
SP11	25.00	20.00	5.00	22.00
SP12	30.00	24.00	6.00	28.78
SP13	30.00	24.00	6.00	26.08

Fonte: Do autor(2017)

A tabela acima apresenta uma visão geral sobre o andamento do

projeto, mostrando um panorama referente a Sprint, o esforço inicial planejado em horas, a duração do projeto sem o pulmão, o pulmão alocado e a duração total de cada sprint.

Algumas informações importantes são relevantes para análise dos resultados. Uma delas é o Percentual utilizado para a estimativa do pulmão. Seguindo as orientações de Leach (2000), basicamente calculamos 20 % a partir do tempo estimado para cada Sprint.

5.1 ETAPA DE CONCLUSÃO

Nesta etapa, são analisados os fatos importantes ocorridos durante o projeto, bem como as oportunidades de melhoria. Desta forma, apresentamos neste item, uma análise dos micro Projetos abordados neste estudo. Conforme observado na Tabela 2, podemos avaliar que as Sprints 01,02,03,07,09 e 10 entraram na corrente crítica e consumiram o pulmão do projeto, extrapolando o tempo planejado. Porém, a data planejada para a liberação do projeto foi mantida, fator este que propõe que o acompanhamento dos projetos utilizando a corrente crítica proporciona uma visão importante para o gestor que consegue tomar decisões referentes aos desvios, de forma rápida podendo replanejar atividades a fim de manter a *release*.

Outro fator importante que pode ser observado, é que após a execução de 3 sprints utilizando o modelo de gestão pela corrente crítica, houve um ganho na forma de estimativa, a qual proporcionou que as 3 próximas sprints, embora tenha havido consumo do pulmão, foram entregues abaixo do tempo estimado. Observamos uma tendência após a execução histórica utilizando o método, pois o aprendizado absorvido pela equipe, proporciona um ganho no sentido de estimativa e execução do projeto tanto no que se refere a estimativas iniciais quanto na entrega.

6. CONCLUSÃO

A implementação da metodologia da Corrente Crítica vem ganhando cada vez mais adeptos, com muitos artigos publicados hoje em dia as organizações não se vem mais nos formatos tradicionais de gerenciar projetos.

Com o conhecimento de uma nova metodologia, a Corrente Crítica não é considerada complexa, porém seu raciocínio lógico exige muita observação, tendo como promessas reduzir prazos gerando menos caos nas equipes executoras e ganho de credibilidade através dos orçamentos.

Essa nova linha de pensamento pode estrategicamente significar a diferença entre o sucesso e o fracasso de um projeto.

Pode-se concluir então que na figura dois (2) teve sprints que entraram em corrente crítica com o uso do pulmão, portanto, a data para a liberação do projeto foi mantida; ou seja é notável que o uso da teoria da corrente crítica proporciona um ganho no sentido da estimativa de tempo e execução do projeto.

REFERENCIAS

BRANDÃO, Barbara Parreira; NASCIMENTO, Gracielly Ribeiro do; ABADIA, Ludymilla Santos de; PACO, Tatiany da Rocha; REZENDE, Ricardo Caetano. Teoria das restrições aplicada a uma linha de produção de ketchup.**ABEPRO.** Bento Gonçalves, RS, outubro de 2012. Disponível em <http://www:abepro.org.br/biblioteca/enegep2010_TN_STO_113_739_15576.pdf. Acesso em 29 mar. 2016.

CLELAND, David I.; IRELAND, Lewis R. **Gerenciamento de projetos.** 2. ed Rio de Janeiro: LTC, 2007. xii, 371 p.

CLELAND, David I.; IRELAND, Lewis R. **Gerenciamento de projetos.** 2. ed Rio de Janeiro: LTC, 2007. xii, 371 p.

GOLDRATT, Eliyahu. **Corrente Crítica**. São Paulo: Ed. Nobel. 1997.

LEACH, L. P. **Critical Chain Project Management**. Norwood, Artech House, Inc., 330 p. 2000.

PRESSMAN, Roger S. **Engenharia de Software**. 6.ed. São Paulo: McGraw-

Hill, 2006.

SAMARANI, Paulo R. de M..**Um modelo de implementação do Capabilit Maturity Integration nível 2**. 2005. 135f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porta Alegre. Disponível em: <http://www.lume.ufrgs.br/handle/10183/5698>. Acesso em: 18 abril de 2016.

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

SILVA, Éverton Maurer da; RODRIGUES, Luis Henrique; LACERDA, Daniel Pacheco. Aplicabilidade da corrente crítica da teoria das restrições no gerenciamento de projetos executivos de engenharia: um estudo de caso em uma refinaria de petróleo. **Gest. Prod.**, São Carlos , v. 19, n. 1, p. 1-16, 2012 . Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000100001&lng=pt&nrm=iso>. acessos em 22 mar. 2016.