AVALIANDO A INFLUÊNCIA DAS PRÁTICAS DE DESENVOLVIMENTO DE *SOFTWARE* PARA OTIMIZAÇÃO DA GERÊNCIA DE PROJETOS

ASSESSING THE INFLUENCE OF PRACTICE FOR SOFTWARE DEVELOPMENT PROJECT MANAGEMENT OPTIMIZATION

Jenifer Vieira Toledo
jenifer@jenifer.eti.br

Adicinéia Aparecida de Oliveira
adicineia@ufs.br

Universidade Federal de Sergipe, Brasil.

Resumo: Sediando eventos, como a Copa do Mundo, os Jogos Olímpicos e os Jogos Paraolímpicos o Brasil passará por um período de forte demanda no desenvolvimento de *software*. Havendo uma maior necessidade do apoio de técnicas, métodos ou modelos que auxiliem no cumprimento dos prazos e orçamentos, questões relevantes para Gerência de Projetos de *software*. Neste contexto, este trabalho tem por objetivo discutir a otimização da Gerência de Projetos de *software* com o uso das boas práticas de métodos de desenvolvimento de *software*. A otimização no gerenciamento de projetos é fundamental às empresas que visam aproveitar as oportunidades de negócios que estes eventos promoverão nos próximos anos.

Palavras-chave: Processos; Métodos; Projetos e Gerência.

***Abstract*:** Hosting events *like the World Cup, the Olympic* and Paralympic Games Brazil will for a period of strong demand in software development. If there is a greater need for support of techniques, methods or models to assist in meeting deadlines and budgets, issues relevant to Project Management software. In this context, this paper aims to discuss the optimization of Project Management software with the use of good software development methods. The optimization project management is key to companies seeking to take advantage of business opportunities that will promote these events in the coming years.

Keywords: Processes; Methods; Projects and Managem.

1 INTRODUÇÃO

Sediando a Copa do Mundo em 2014, os Jogos Olímpicos e os Jogos Paraolímpicos em 2016, o Brasil terá eventos com demandas específicas relacionadas aos serviços em Tecnologias da Informação e Comunicação (TIC), precisando de técnicas, métodos e modelos para melhorar a garantia dos prazos e da qualidade dos produtos de *software* desenvolvidos especificamente para tais eventos, destacando que estes produtos não poderão ser submetidos a problemas como o mal planejamento de custos e tempo, já que estes eventos não deixarão de ocorrer ou poderão ser adiados.

Frequentes no desenvolvimento de *software*, problemas relacionados a custos e tempo, juntamente com a pressão do mercado, algumas organizações desenvolvedoras passam a terem resultados negativos quanto ao desenvolvimento de *software*. De acordo com o *Standish Group* (1), dados sobre o sucesso e falha de projetos de Tecnologia da Informação (TI) da pesquisa realizada em 2010 informam que, 21% dos projetos desenvolvidos nas organizações são considerados como fracassados, 42% dos projetos são contrariados ao planejamento inicial e 37% apenas são realizados com sucesso. Assim, as organizações se preocupam cada vez mais com a qualidade do processo de desenvolvimento de *software*, tendo em vista a qualidade do produto de maneira a atender as expectativas do cliente (2).

Em um ambiente de mudanças rápidas e complexas para o desenvolvimento de *software*, as organizações carecem também de algum fator para melhoria da Gerência de Projetos, pois, esta possui um grau de rigidez, no qual na maioria das vezes, organizações não conseguem o adaptar ou o manter, gerando uma sequência de conflitos e dificuldades para gerenciar seus projetos, fato que tem como consequência, projetos de *software* falhos, seja porque não cumprem o orçamento, o cronograma, faltam funcionalidades que não

atendem às necessidades dos usuários, ou ainda, porque estes fatores estão presentes em conjunto.

Neste contexto, percebe-se que a Gerência de Projetos necessita de um apoio, diante as dificuldades enfrentadas em suas fases. Este artigo tem como objetivo discutir a otimização da Gerência de Projetos de *software* com o uso das boas práticas dos métodos de desenvolvimento de *software*. O trabalho esta dividido em oito seções, sendo esta a primeira. A segunda apresenta um resumo sobre Processos de Desenvolvimento de *Software* (PDS). Na seção três compara-se os métodos de desenvolvimento de *software*. Seguindo, na seção quatro, busca-se identificar a influência das práticas dos métodos de desenvolvimento de *software* diante alguns fatores críticos organizacionais. Na seção cinco alguns trabalhos relacionados são apresentados. Já na seção seis discute-se sobre a Gerência de Projetos e suas dificuldades. Na seção sete faz-se uma análise da utilização de boas práticas dos métodos de desenvolvimento de *software* para otimização da Gerência de Projetos. Finalizando o artigo a seção oito apresenta as conclusões do trabalho.

2 PROCESSOS DE DESENVOLVIMENTO DE *SOFTWARE*

Define-se processo de *software* como um arcabouço para as tarefas que são necessárias para construir *software* de alta qualidade, servindo como base para a utilização de métodos e ferramentas. O processo determina a sequência em que os métodos serão aplicados, os produtos serão entregues, a relação entre os usuários e a equipe de desenvolvimento e as ferramentas para se atingir o objetivo (3).

Preocupadas cada vez mais com o processo, a indústria e a academia, buscam desenvolver produtos com qualidade, aumentando a produtividade de suas equipes, diminuindo os custos e melhorando a previsibilidade dos seus projetos, investindo assim em PDS e apostando em recursos de apoio como um meio para alcançar algumas das exigências do mercado. Dentre estes recursos, as organizações vêm buscando o apoio e utilização dos métodos de desenvolvimento de *software*.

Porém, há uma discussão sobre a identificação de qual método deve-se utilizar diante determinados projetos e problemas enfrentados no cotidiano do desenvolvimento de *software* destas organizações. Na próxima seção discute-se sobre os métodos mais citados na literatura, em busca da identificação de suas principais práticas.

3 MÉTODOS DE DESENVOLVIMENTO DE *SOFTWARE*

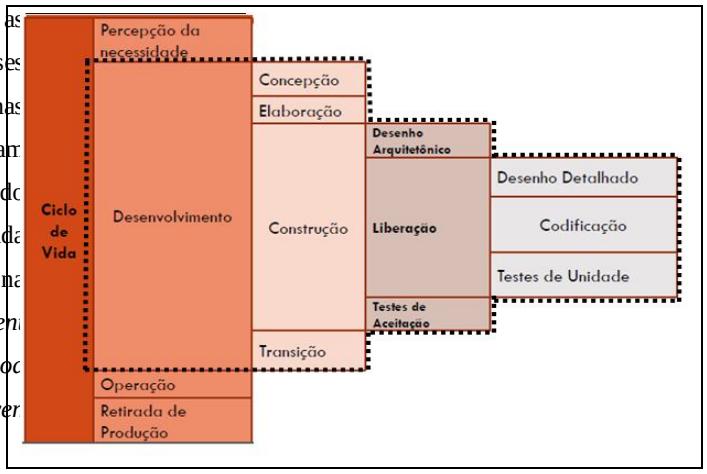
Para desenvolver um produto de *software* deve-se passar por um processo determinando várias atividades de forma planejada, visando custos e prazos estimados, utilizando teorias, métodos, técnicas, modelos e ferramentas adequadas. Porém, parte das empresas brasileiras de desenvolvimento de *software* não possui um processo de desenvolvimento de *software* definido e institucionalizado (4).

Além disso, considerando que um dos fatores para o sucesso da melhoria do processo de *software* são as pessoas, estas precisam ser treinadas e motivadas para tal melhoria. Podendo também apresentar dificuldades em áreas específicas envolvidas na execução do processo, como por exemplo, no conhecimento dos métodos disponíveis para o desenvolvimento de *software* (5). A adoção de um método de desenvolvimento depende em parte de um fator-chave, ou seja, da clareza de suas exigências.

Segundo (6), os Métodos Dirigidos por Planejamento (MDP) são focados na disciplina, ou seja, utilizam uma abordagem sistemática onde o *software* é construído a partir dos requisitos até o código final, passando por uma série de fases, estas nas quais são bem documentadas. Como os principais MDP há o *Unified Process* (UP) e o *Rational Unified Process* (RUP). Já os Métodos Ágeis (MA) não rejeitam os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas simplesmente mostram que eles

estão em segundo plano quando comparados com os indivíduos e interações, com a colaboração do cliente e as respostas rápidas a mudanças e alterações. Esses conceitos aproximam-se melhor, da forma que pequenas companhias de Tecnologia da Informação (TI) trabalham e respondem às mudanças (7). MA vêm ganhando crescente popularidade desde o início da última década (8), existindo determinados métodos mais citados na literatura, sendo eles: *Adaptative software development* (ASD), *Crystal*, *Dynamic Systems Development Method* (DSDM), *Extreme Programming* (XP), *Feature Driven Development* (FDD) e *Lean Development* (LD).

Figura 1 - Visão do Ciclo de vida do *Software* (10).



3.1 Comparando MDP e MA

Escolher um método adequado para o desenvolvimento de *software* em uma organização não é uma tarefa trivial, mas é algo importante para que possa disciplinar o desenvolvimento com o objetivo de tornar o processo mais previsível e eficiente (9). Através de estudos e pesquisas é possível perceber que os métodos apresentam atividades semelhantes em relação aos seus processos de desenvolvimento. Porém, alguns apresentam atividades com particularidades, como por exemplo: a programação em pares, as estórias escritas pelos clientes do XP; as inspeções de código de FDD e as sessões de *workshops* sugeridas pelo ASD; entre outras particularidades.

Cada método possui características que determinam a forma de desenvolvimento de um *software*, estas características podem determinar o sucesso ou o fracasso dependendo da maneira de como são utilizadas e o conhecimento da equipe.

Além disso, para entender a aplicação dos métodos no processo de desenvolvimento de *software* é importante visualizar a localização deste, no Ciclo de Vida do *Software* (CVS). Na Figura 1 é possível visualizar as etapas que compreendem o Ciclo de Vida do Processo de Desenvolvimento de *Software* (CVPS).

O Quadro 1 exibe um comparativo entre MDP com os MA citados anteriormente, neste comparativo apresenta-se como critérios de confiança as principais práticas utilizadas em cada método diante as fases do ciclo de vida do PDS. Com esta comparação, há outro ponto interessante sobre a visão dos utilizadores de tais métodos e práticas, enquanto alguns defensores “radicais” dos MA são categóricos em criticar e apontar as falhas dos MDP outros especialistas, têm uma postura mais amena, enxergando até mesmo uma possibilidade de integração entre as duas abordagens (11). Pois, ambos os lados têm pontos importantes a serem analisados e balanceados, gerando os Métodos Híbridos.

Quadro 1- Comparação entre MDP e MA.

Métodos	**Fases do Ciclo de Vida do Processo de Desenvolvimento de *Software***			
	Concepção	**Elaboração**	**Construção**	**Transição**
***Unified Process* (UP)**	Documento Visão, Plano de Projeto, Avaliação Inicial de risco, Modelo de negócio, Modelo de Caso de Uso, Protótipos.	Descrição da arquitetura do *software*, Requisitos Funcionais, Protótipos executáveis, Lista de risco revisada, Manual preliminar do usuário.	Modelo de projeto, Codificação, Plano de procedimento de teste, Documento de apoio ao usuário.	Relatório de teste beta, Incremento de *software* entregue, Realimentação geral do usuário.
***Rational Unified Process* (RUP)**	Documento Visão, Modelo de Caso de Uso, Plano de Projeto, Plano de Custos, Modelo de Caso de Negócios, Protótipos	Arquitetura, Projeto do sistema, Modelagem de Negócio, *Design*, Protótipos executáveis, Lista de riscos.	Lista de Itens de trabalho, Implementação, Caso de Teste, Plano de Teste, Teste de Desenvolvedor.	Acompanhamento da Qualidade, Capacidade dos Usuários, Alocação de recursos, Controle de versões e mudanças, Ferramentas de infraestrutura.
***Adaptative software development* (ASD)**	Definição dos objetivos do projeto e requisitos necessários através de *workshops*.	Definição de prazos, Componentes simultaneamente.	Implementação, Qualidade do ponto de vista técnico e do cliente.	Testes e *Feedbacks* do cliente.
Crystal	Modelagem do Método, Reflexão de *Workshops*. Documento Visão.	Planejamento a jato, Estimativas por experiência, Descobertas de iterações do usuário.	Reuniões diárias em pé, Programação lado a lado, Andamento do projeto. Mine processos. Integração contínua e	Não definido.
***Dynamic Systems Development Method* (DSDM)**	Acompanhamento de requisitos, Protótipos, Cooperação entre os *stakeholders*, *Workshops*.	Alinhamento com os interesses de negócio, Revisões durante o processo.	Usuário ativo envolvido, Equipes com autonomia, Mudanças reversíveis e Testes.	Entregas frequentes, Testes, Gerência de configuração, Manuais, Revisão do projeto.
***Feature Driven Development* (FDD)**	Lista de funcionalidades e prioridades, Diagramas simplificados.	Projeto das funcionalidades, Revisão e validação de requisitos.	Construção das funcionalidades, Inspeções, Integração Regular, Testes.	Gerência de configuração.
***Extreme Programming* (XP)**	Planejamento, Estórias, Cliente Presente.	*Design* simples, Teste antecipado.	Refatoração, Programação em par, Propriedade Coletiva de Código, Integração Contínua, Padrão de código, Teste unitário, Reuniões Diárias, 40 horas semanais.	Teste de aceitação.
***Lean Development* (LD)**	Propor soluções, Tomar decisões com conhecimento, Otimização.	Eliminar gastos.	Testes, Cliente envolvido, Melhoria contínua.	Testes, Entrega rápida.

4 INFLUÊNCIAS DAS PRÁTICAS DOS MÉTODOS DE DESENVOLVIMENTO DE *SOFTWARE* DIANTE FATORES CRÍTICOS

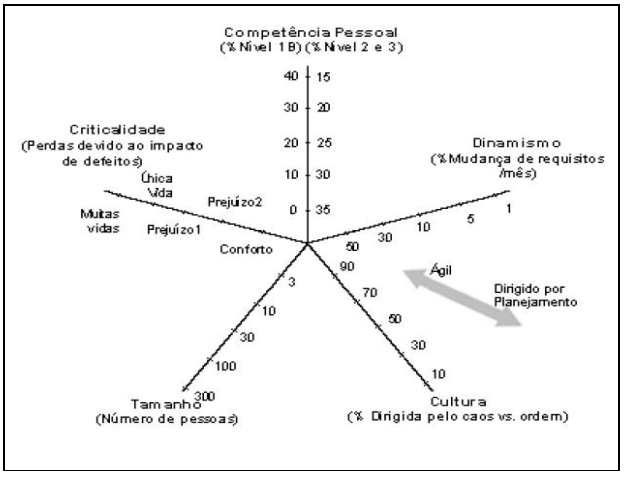
Define-se cinco fatores críticos, através dos quais a equipe e o contexto de desenvolvimento de *software* podem ser caracterizados, servindo como parâmetros para determinar a adequação de MA e MDP para o desenvolvimento de *software*, sendo eles (6):

- tamanho: refere-se ao tamanho da equipe envolvida e ao esforço necessário no desenvolvimento;
- criticalidade: refere-se às perdas ocasionadas em função de erros presentes no produto, relaciona-se ao risco de vidas humanas pelo uso do *software* e a outros riscos menos graves;
- dinamismo: refere-se à possibilidade de mudanças nos requisitos em função de mudanças no ambiente do problema, ou em função de um trabalho de extração mal executado, implicando em alterações no *software*;

- competência pessoal: refere-se à competência dos profissionais envolvidos no desenvolvimento, variando de pouco experiente a especialista; e,
- cultura: refere-se à cultura dos desenvolvedores em relação ao desenvolvimento e Gerência de Projetos de *software*. Os desenvolvedores podem possuir a cultura na qual se sentem mais confortáveis tendo mais liberdade para produzir o *software*, ou podem ter a cultura na qual se sentem mais confortáveis trabalhando com regras, procedimentos e políticas bem definidas e estabelecidas.

Estes fatores são organizados graficamente conforme apresentado no Gráfico 1, permitindo simular o nível de cada fator crítico avaliado nas organizações e sugerir qual método mais adequado para o desenvolvimento de *software*.

Gráfico 1 - Adequação de métodos para desenvolvimento de *software* (6).



O gráfico é organizado em uma escala em forma de estrela, de modo que os itens mais centrais da escala apontam para o uso de MA, enquanto que os itens mais externos apontam para o uso de MDP. Em busca não exatamente de qual método mais indicado a ser utilizado no desenvolvimento de *software*, mas de quais práticas mais indicadas para o desenvolvimento de *software*, busca-se verificar na próxima subseção a influência das principais práticas dos métodos de desenvolvimento de *software* utilizadas em cada fase do CVDS, diante os fatores críticos avaliados nas organizações indicados por (5).

4.1 Justificando a influência de práticas dos métodos de desenvolvimento de *software* diante fatores críticos

A análise sistêmica, ou seja, a análise do todo e das relações de suas partes poderão levar o tomador de decisões a entender melhor o problema que está sendo avaliado (12). Em *software*, pode-se afirmar que a falta de uma visão sistêmica do processo de desenvolvimento leva os gerentes a tomarem decisões reativas e pontuais, considerando apenas o problema presente sem relacioná-lo com o seu ambiente, suas variáveis e demais problemas correlacionados (13).

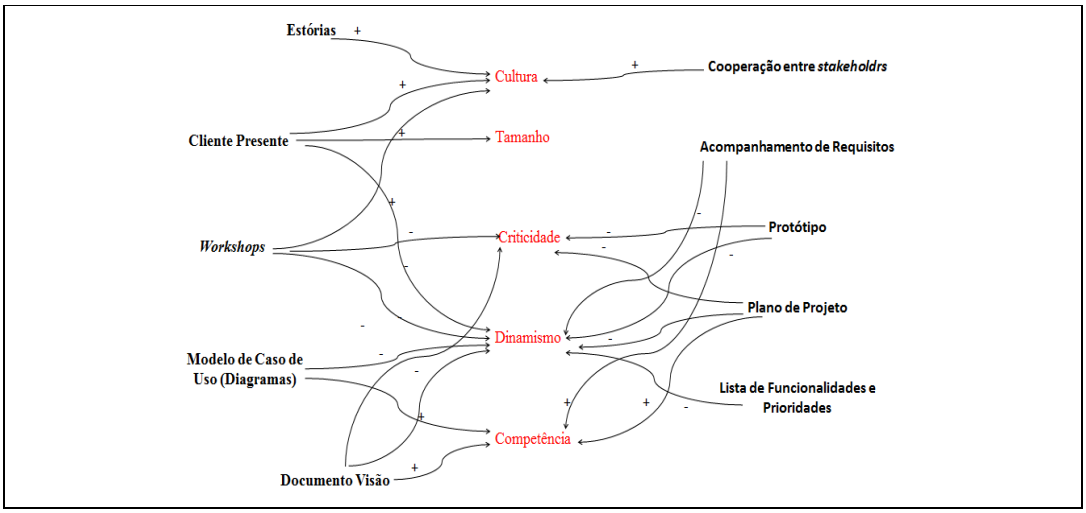
As práticas, ou seja, a maneira de se executar algo, a depender do contexto em que esta inserida pode ser boa ou ruim, deste modo práticas dos MA e MDP se mostram úteis em uma grande diversidade de situações (14). Com o objetivo de filtrar este contexto de situações, buscando facilitar as decisões de quem o analisa, será verificada a influência das práticas dos métodos de desenvolvimento de *software* diante os fatores críticos avaliados nas organizações, de acordo com (6), através da análise sistêmica, gerando diagramas de influência. O primeiro diagrama de influência se refere à fase de Concepção do CVPDS, com as principais práticas dos métodos nesta fase, sendo elas:

- Documento Visão, Plano de Projeto (Avaliação de Riscos, Estimativas de Custo e Prazo), Modelo de Caso de Uso, Protótipos, *Workshops*, Acompanhamento de requisitos,

Cooperação entre os *stakeholders*, Lista de funcionalidades e prioridades, Estórias e Cliente Presente.

A influência de tais práticas, diante os fatores críticos possibilita desenvolver o diagrama de influência, conforme ilustrado na Figura 2.

Figura 2 - Relacionamento entre Práticas da fase de Concepção com Fatores Críticos.



Já a análise do diagrama apresentado pode ser feita da seguinte forma:

- em um primeiro momento, nota-se que a utilização de *Estórias* pode afetar a cultura da equipe em relação à Gerência de Projetos e ao desenvolvimento de *software;*
- ter o *Cliente presente*, ou seja, mais envolvidos na equipe, afeta também a cultura da equipe, tendo pessoas com *perfis* diferenciados no cotidiano do trabalho. Mas podendo diminuir as mudanças constantes diante a facilidade do esclarecimento de dúvidas sobre o que deve ser produzido;
- da mesma forma a utilização de *Workshops* também afeta aos costumes da equipe, ou seja, a cultura. Por outro lado, diminui o dinamismo, ou seja, a possibilidade de mudança de requisitos e possibilidade da identificação de erros;
- utilizar *Modelo de Caso de Uso* ou diagramas, também diminui o dinamismo, mas necessita de pessoas capacitadas para elaboração do mesmo;
- a elaboração de um *Documento Visão* traz um menor índice de dinamismo e criticidade, porém havendo necessidade de pessoas mais capacitadas para tal produção;
- ter *Cooperação entre stakeholders* compromete a cultura dos envolvidos, nem sempre as pessoas estão dispostas a enfrentar e lidar com diferenciados *perfis;*
- já o *Acompanhamento de Requisitos*, diminui o dinamismo, necessitando de pessoas mais experientes para realizar o acompanhamento e ter resultados válidos do mesmo;
- criar *Protótipos* traz menos riscos e dinamismo podendo visualizar de melhor forma o que será produzido;
- um *Plano de Projeto* gera menos dinamismo das funcionalidades e

criticidade do projeto, porém necessitando de pessoas mais capacitadas; e,

- uma *Lista de Funcionalidades* assim como outras práticas traz um menor índice de mudanças.

É possível perceber através da Figura 2, que na fase de Concepção do ciclo de vida do processo de desenvolvimento de *software*, há um maior fluxo das práticas dos métodos relacionadas ao fator crítico Dinamismo. Concluindo que a implantação de tais práticas na fase de Concepção traz menores mudanças de requisitos e geração de alterações futuras. Ao realizar tal análise diante cada fase do CVPDS, é interessante observar que nas fases iniciais de Concepção e Elaboração teve-se um maior fluxo sob a influência das práticas dos métodos de desenvolvimento de *software* em relação ao fator crítico Dinamismo. Já nas fases de Construção e Transição a maior influência foi em relação ao fator crítico Cultura e Competência. Fatores estes nos quais são sempre discutidos e levantados como problemas no desenvolvimento de *software*. É importante deixar claro que para este trabalho buscou-se avaliar a influencia das práticas dos métodos de desenvolvimento de *software* apenas diante os fatores críticos citados por (6). Como um trabalho futuro ou aprofundamento da pesquisa é interessante verificar a influência de tais práticas diante outros fatores críticos identificados na literatura ou no mercado para o desenvolvimento de *software.*

5 TRABALHOS RELACIONADOS

Pereira (15) apresenta um debate teórico, com um enfoque sobre a Gestão de Projetos na produção de *software* em micro e pequenas empresas.

Já Fagundes (16) mostra uma comparação somente entre os métodos ágeis XP, FDD e ASD, bem como o modelo de gestão *Scrum* buscando auxiliar a equipe de desenvolvimento na escolha do método que melhor se adapte a suas expectativas.

Chagas (17) buscou apresentar um cenário atual dos processos de desenvolvimento de *software* e mostrar com clareza, métodos e *frameworks* existentes para auxiliar a construção de *software*. Nesse âmbito abordou-se: *Crystal*, DSDM, FDD, LD, ASD, XP, *Scrum* e RUP.

Vasco *et. al.* (18) comparam os métodos de desenvolvimento de *software* RUP e XP, buscando a redução de riscos desenvolvendo o sistema de forma incremental.

E Trecanni *et. al*. (19) apresentam resultados parciais de um estudo empírico que visa identificar como são aplicadas os MA no processo de desenvolvimento e manutenção de sistemas. Todas as empresas avaliadas utilizam o modelo de gestão *Scrum*, como base de seu processo de desenvolvimento, incorporando também práticas de MA como: XP e FDD.

Tal avaliação proposta diferencia-se dos demais trabalhos já realizados, pois busca identificar as melhores práticas dos métodos de desenvolvimento mais citados na literatura através da análise sistêmica, bem como, verificar qual a contribuição destas práticas para as áreas de conhecimento da Gerência de Projetos.

6 GERÊNCIA DE PROJETOS DE SOFTWARE

Do mesmo modo que o processo de desenvolvimento de *software*, a Gerência nos últimos anos tem sido estudada em busca de uma disciplina estabilizada e compreendida para os projetos (20). Apesar das organizações de *software* buscarem por processos bem definidos para o desenvolvimento de seus produtos e serviços, como já citado, estes processos existentes ainda precisam da Gerência de Projetos, como a gerência das pessoas e de outros recursos a fim de planejar, analisar, projetar, construir, testar e manter um produto de *software*. Mesmo sendo uma atividade relevante para o processo de desenvolvimento de *software*, às organizações mostram um perfil imaturo, sendo que, de acordo com a Pesquisa Nacional de Qualidade e Produtividade no Setor de *Software* Brasileiro (21),

apenas 30% a 35% das empresas brasileiras possuem atividades de gerência. Já Kerzner (22), enumera uma lista de fatores que podem comprometer o gerenciamento de projeto, tais como:

- um trabalho mal feito que precise ser refeito;
- pobres canais de comunicação;
- falha na delegação de tarefas;
- metas e objetivos vagos;
- restrições burocráticas;
- gerenciamento exagerado; e,
- falta de adequadas ferramentas de projeto.

Segundo os autores (23), as organizações precisam adaptar suas estruturas, estratégias e políticas para satisfazerem aos novos ambientes e a crescente demanda da sociedade contemporânea por sistemas de informação. As organizações devem buscar alternativas sobre como gerenciar seus projetos de *software*, visando à diminuição dos fracassos e a melhoria na qualidade de seus produtos e serviços. Desta forma, busca-se analisar se com a utilização de práticas dos métodos de desenvolvimento de *software* é possível otimizar a Gerência de Projetos de forma que se possa atingir seus objetivos.

No processo de desenvolvimento de *software* existem os métodos de desenvolvimento conforme já citados e na Gerência existem as áreas de conhecimento, descritas a seguir (24):

- gestão da integração: tem como objetivo fazer o controle geral das mudanças e monitorar a execução do plano do projeto, desde seu inicio com o termo de abertura do projeto até seu final com o encerramento do projeto, realizando negociações dos objetivos conflitantes, dando alternativas ao projeto com a finalidade de atender as necessidades e expectativas de todas as partes interessadas;
- gestão de escopo: contextualizada por definir de forma periódica, as necessidades, os produtos e os *stakeholders* do projeto, procurando somente o necessário para entregar o projeto com sucesso;
- gestão de tempo: inclui todos os procedimentos necessários para realizar o projeto dentro do prazo estipulado;
- gestão de custos: são considerados todos os atributos envolvidos no planejamento, estimativa, orçamento e controle de custos de modo a encerrar o projeto dentro do orçamento previsto;
- gestão da qualidade: baseado tanto na qualidade do produto, sendo necessário verificar se as necessidades pré-estabelecidas pelos clientes estão implementadas no produto, quanto na qualidade do processo, sendo importante analisar se o processo está organizado, estruturado para produzir subprodutos de qualidade;
- gestão de recursos humanos: tem como objetivo administrar a mão de obra humana, atribuir funções e responsabilidades, relações interpessoais e de equipe, buscando sempre o melhor aproveitamento das pessoas envolvidas no projeto;
- gestão das comunicações: reúne os processos necessários para garantir a geração, a coleta, a distribuição, o armazenamento, a recuperação e o destino final das informações do projeto, de forma oportuna e adequada;
- gestão de aquisição: o objetivo principal é obter bens e serviços externos à organização executora. Consiste do planejamento de aquisição, planejamento de solicitação, solicitação de propostas,

seleção de fornecedores, e administração e encerramento de contratos; e,

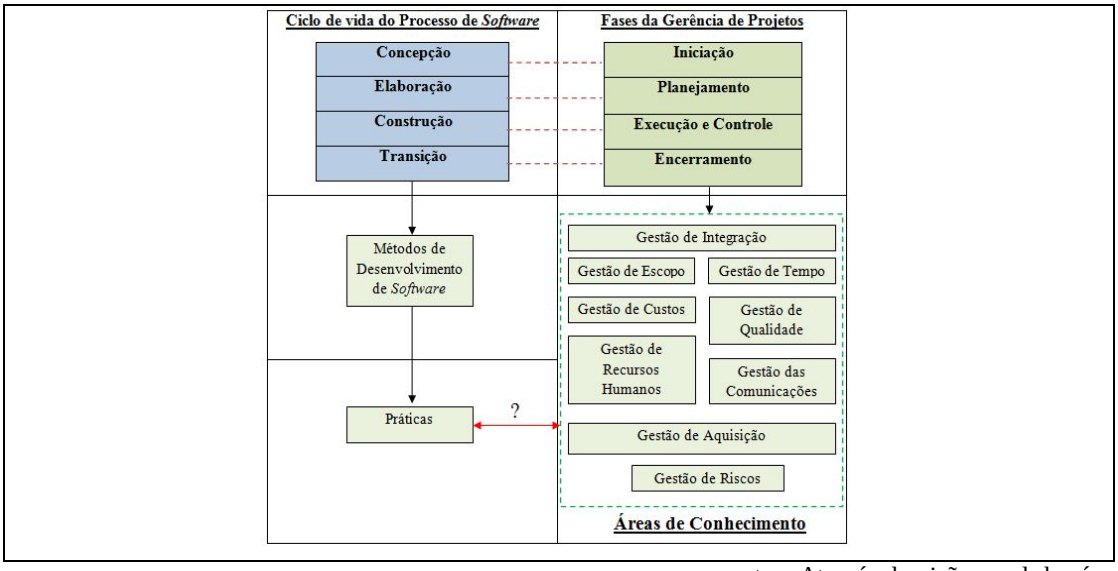
- gestão de risco: o objetivo principal é maximizar os resultados de ocorrências positivas e minimizar as consequências negativas ou até mesmo eliminar eventos adversos, tratando e controlando os riscos.

7 OTIMIZANDO A GERÊNCIA DE PROJETOS COM PRÁTICAS DE DESENVOLVIMENTO DE *SOFTWARE*

A Gerência de Projetos de *software* surge como uma das atividades mais importantes do processo de desenvolvimento de *software*. No início da década de 70, o fracasso no desenvolvimento de grandes projetos de *software* se dava pela falta de gerenciamento das atividades, o que acarretava atrasos na entrega dos produtos, sistemas que não refletiam a verdadeira necessidade do cliente e problemas com as equipes no ambiente de desenvolvimento (25).

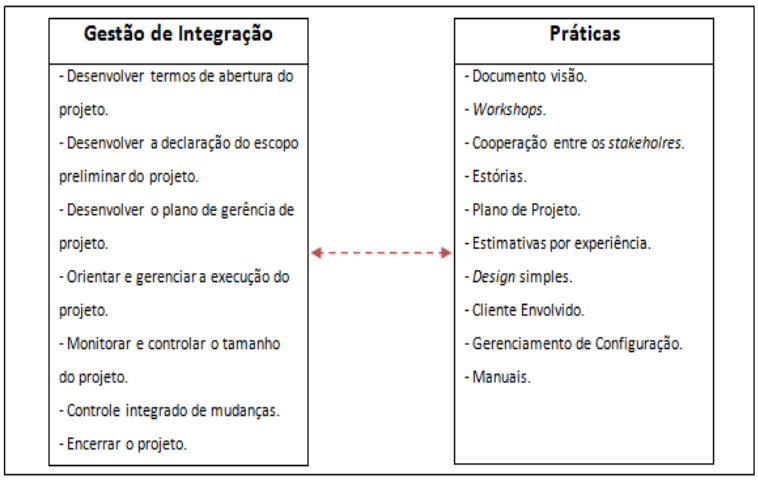
Apesar dos PDS identificarem a importância das atividades relacionadas à Gerência, não tratam de forma adequada estes aspectos. Sendo notória uma relação constante entre o PDS e a Gerência de Projetos, observa-se que, as fases relacionadas ao PDS são realizadas de forma paralela as fases da Gerência de Projetos, ou seja, estão basicamente alinhadas durante o desenvolvimento de *software*. Através desta relação, como um ponto do trabalho a ser verificado, ilustrado na Figura 3, pretende-se analisar a possibilidade de otimizar a Gerência de Projetos de *software* com o apoio das práticas já utilizadas no PDS.

Figura 3 - Análise quanto as práticas dos métodos de desenvolvimento de *software* e áreas de conhecimento da Gerência de Projetos.



Realizar o gerenciamento das áreas de conhecimento conforme a literatura acaba sendo um fator distante da realidade das organizações diante as exigências do mercado de desenvolvimento de *software*, que buscam cada vez mais um desenvolvimento rápido e com menores custos. Através da visão geral das áreas de conhecimento da Gerência de Projetos e o PDS ilustrados na Figura 3, cria-se uma visão crítica sobre determinados pontos. Um ponto inicial é sobre a Gestão de Integração

que traz como principal objetivo controlar as mudanças e monitorar a execução do plano do projeto, as com a utilização das práticas dos métodos de desenvolvimento de *software*, é possível atingir o objetivo da Gestão de Integração, conforme ilustrado na Figura 4.



Para sustentar tal objetivo da Gestão de Integração, práticas dos métodos de desenvolvimento como: documento visão, *workshops*, cooperação entre os *stakeholres*, estórias, plano de projeto, estimativas por experiência, *design* simples, cliente envolvido, gerência de configuração e manuais são possíveis de serem aplicadas de forma a auxiliar ou substituir os processos da Gestão de Integração. Tal procedimento foi verificado em cada área de conhecimento da Gerência de Projetos, sendo possível verificar e analisar que as práticas dos métodos de desenvolvimento de *software*, podem a vir contribuir para as áreas de conhecimento da Gerência de Projetos.

8 CONCLUSÃO

Não basta apenas o PDS, também é necessário gerenciar o projeto, ou seja, controlar o desenvolvimento de modo que o aproveitamento seja satisfatório exige um acompanhamento, além da utilização de fatores que qualifiquem o andamento do projeto. Para cumprir estes objetivos, um gerente de projetos necessita de algum tipo de suporte, geralmente baseado em um modelo de desenvolvimento, que permita lidar com diferentes variáveis do projeto, responsabilidades e tarefas, sendo um trabalho extenso e cansativo. Com o uso das práticas nos métodos de desenvolvimento estas tarefas podem ser realizadas de forma menos complexa e segura, trazendo maior garantia quanto ao cumprimento de cronogramas e planejamento dos produtos de *software* a serem desenvolvidos.

Com o crescimento na demanda por desenvolvimento de soluções de *software*, devido aos grandes eventos esportivos que o Brasil sediará nos próximos anos, forçará as organizações e equipes de desenvolvimento ajustar seus processos de desenvolvimento para atender as solicitações em tempo. Neste artigo discutiu-se a Gerência de Projetos, buscando a adoção de práticas alinhadas aos métodos de desenvolvimento auxiliando aos gestores de projetos de *software* a maximizarem seus recursos e assim como as chances de um produto de *software* de qualidade.

Como trabalhos futuros podem ser desenvolvidos, como por exemplo, ferramentas para acompanhamento dos projetos alinhados às práticas adotadas; o estabelecimento de métricas para acompanhamento e avaliação dos projetos, permitindo mensurar e validar a efetividade da adoção das práticas.

REFERÊNCIAS

1. STANDISH GROUP**. New Standish Group report shows more project failing and less successful projects.** Disponível <http://www.standishgroup.com/newsroom/chaos_2010.php>. Acesso em: 01 Dez.2011.

2. SANTOS, M.A. **Agile UBPM FOR SCRUM: Modelo de Aprimoramento do Gerenciamento e Desenvolvimento Ágil Baseado na Percepção de Valor do Usuário**. Trabalho de Conclusão de curso. Universidade Federal de Lavras. 2011.

3. PRESSMAN, Roger S.. **Engenharia de software**. 6ª Edição. Porto Alegre: Editora Mc Graw- Hill, 2010.

4. FABRI, J.A.; ERARIO, A.L.; DOMINGUES, A.S. **Institucionalizando Práticas nos Cursos de Ciência da Computação e Tecnologia em Análise e Desenvolvimento de Sistemas**. Universidade Tecnológica Federal do Paraná. 2011. Disponível em: <www.elsevier.nl/locate/entcs>. Acesso em: 20 Dez. 2011.

5. RICHARDSON, I.; WANGENHEIM, Chrisiane Gresse von. **Why are small software Organization Different**? IEEE Computer Society. 2007.

6. BOEHM, B. e TURNER, R. **Balancing Agility and Discipline:** A Guide for the Perplexed. Addison – Wesley, Boston, 2004.

7. SOARES, L. **Obtenção de requisitos para customização de processo de desenvolvimento de software**. Master's thesis, Universidade Federal de Viçosa CCE/DPI, dissertação de Mestrado, 2007.

8. MELO, C. O.; Santos Jr., C. D.; FERREIRA, G. R. M e Kon, F. **Um estudo exploratório dos fatores associados ao estímulo do aprendizado em times ágeis na indústria**. In: VII Experimental Software Engineering Latin American Workshop, Goiânia, 2010.

9. FOWLER, Martin. **The New Methodology**. 2005. Disponível em: <www.martinfowler.com/articles/newMethodology.html>. Acesso em: 12 Jan. 2011.

10. OLIVEIRA, A.A. Engenharia de Software. **Processos de Software e SWEBOK.** 2010.

11. GLASS, R., "**Agile Versus Traditional**: Make Love, Not War," *Cutter IT Journal* , pp. 12-18, Dec. 2001.

12. SENGE.P.M. **A quinta disciplina:** teoria e prática da organização de aprendizagem. 3 ed. Editora Nova Cultura LTDA. São Paulo. 1990.

13. AMBRÓSIO, B.G.MSc. Universidade Federal de Viçosa. **Modelagem da fase de requisitos em processo de desenvolvimento de software: uma abordagem utilizando dinâmica de sistemas**. 2008.

14. HIGHSMITH, J. **Agile Project Management** – Creating Innovative Products. Addison – Wesley, 2004.

15. PEREIRA, S.L. e VIEIRA, T.P.B. **Estudo da aplicação de um processo gerenciado de produção de software em MPEs.** Centro de Ciências Sociais Aplicadas. Universidade Federal da Paraíba. João Pessoa. 2006.

16. FAGUNDES, P. **Framework para Comparação e Análise de Métodos Ágeis**. Florianópolis. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Santa Catarina, Florianópolis, 2008.

17. CHAGAS, M.V.L. **Processo de Desenvolvimento de Software:** O estado da arte. Universidade Estadual de Londrina. Londrina, 2008.

18. VASCO, Carlos G.; VITHOFT, Marcelo Henrique; ESTANTE, Paulo Roberto C. **Comparação entre Metodologias RUP e XP**. PUCPR, 2009.

19. TRECCANI, P.J.F e SOUZA, C.R.B. **Utilização de Metodologias Ágeis no Desenvolvimento de Software:** Resultado de um Estudo Empírico. Universidade Federal do Pará. VII Experimental Software Engineering Latin American Workshop. 2010.

20. FREITAS, J. A. S. B. **A dimensão tácita do conhecimento e o trabalho dos gerentes no varejo bancário.** In: Anais do XXX Encontro Nacional da Associação Nacional de Pós-Graduação em Administração. Salvador: ANPAD, 2006.

21. MCT. Ministério de Ciência e Tecnologia. **Pesquisa Nacional de Qualidade e Produtividade no setor de software Brasileiro.** 2009.

22. KERZNER, H. **Project Management**. 7ª. Edição. John Wiley & Sons.2001.

23. NERUR S.; MAHAPATRA, R.; MANGALARAJ, G. **Challenges of Migrating to Agile Methodologies**. Communications of the ACM, v. 48, n. 5.2005.

24. PMBOK – "**A Guide to the Project Management Body of Knowledge**", 2008.

25. SOMMERVILLE. I. **Engenharia de Software** / Ian Sommerville; tradução André Maurício de Andrade Ribeiro; revisão técnica Kechi Hirama. São Paulo: Addison Wesley, 2007.