UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
DEPARTAMENTO DE CIÊNCIAS ADMINISTRATIVAS

Ygor Bruxel

FATORES DE SUCESSO EM PROJETOS DE SOFTWARE QUE UTILIZAM MÉTODOS ÁGEIS EM UMA CONSULTORIA INTERNACIONAL

Porto Alegre
2010

Ygor Bruxel

FATORES DE SUCESSO EM PROJETOS DE SOFTWARE QUE UTILIZAM MÉTODOS ÁGEIS EM UMA CONSULTORIA INTERNACIONAL

Trabalho de conclusão do curso de graduação apresentado ao Departamento de Ciências Administrativas da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Bacharel em Administração.

Orientador: Prof. Dr. Antônio Carlos Gastaud Maçada

Porto Alegre
2010

Ygor Bruxel

FATORES DE SUCESSO EM PROJETOS DE SOFTWARE QUE UTILIZAM MÉTODOS ÁGEIS EM UMA CONSULTORIA INTERNACIONAL

Trabalho de conclusão do curso de graduação apresentado ao Departamento de Ciências Administrativas da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Bacharel em Administração.

Conceito final :

Aprovado em …….. de …… de …….

BANCA EXAMINADORA:

_____
Prof.ª Dr.ª Angela Brodbeck
Universidade Federal do Rio Grande do Su

_____
Orientador – Prof. Dr. Antonio Carlos Gastaud Maçada
Universidade Federal do Rio Grande do Sul l

Lista de Ilustrações

Lista de Tabelas

Sumário

1. Introdução

Desenvolvimento de software é um processo complexo, caro e demorado, e, muitas vezes, o que seria uma vantagem competitiva para a empresa no mercado, acaba se tornando uma fonte de custos. Chow e Cao (2007) colocam que o processo de desenvolvimento de software não tem sido consistentemente de sucesso, resultando em projetos atrasados, falhos, abandonados e rejeitados. Larman (2004) cita o relatório feito pelo Standish Group em 1994 que revela que apenas 16% (dezesseis por cento) dos projetos de software são considerados um sucesso, 31% (trinta e um por cento) são cancelados e 53% (cinquenta e três por cento) tiveram seu custo aumentado, atrasaram ou tiveram seu escopo reduzido.

Baseado na experiência de dezessete desenvolvedores experientes em projetos de software e nas melhores práticas de desenvolvimento de software do mercado o pensamento ágil propõe uma forma diferente de se desenvolver software, voltado à colaboração com cliente, a responder a mudanças, a interação entre os envolvidos no projeto e a entrega de software funcionando como valores a serem seguidos a fim de se obter o sucesso do projeto. Uma pesquisa feita pela Shine Technologies (2003) revelou que 88% (oitenta e oito por cento) dos respondentes notaram uma melhora na qualidade das aplicações desenvolvidas. a mesma pesquisa revelou que cerca de 49% (quarenta e nove por cento) relataram uma diminuição nos custos dos projetos. Assim, identificou-se uma lacuna importante que é investigar quais são os fatores do sucesso na aplicação e uso do pensamento ágil visando auxiliar na gestão deste processo que poderá trazer para a organização vantagens competitivas, principalmente se atender fatores de qualidade, escopo, tempo e custo.

2. Justificativa

Uma empresa, para ser competitiva no mercado atualmente, precisa de um sistema de informação para ajudar na tomada de decisão, gerenciar informações relevantes para a organização, entre outros muitos benefícios que a tecnologia de informação proporciona. Apesar dos esforços para se aplicar metodologias de engenharia de software, o desenvolvimento de software não tem sido constantemente de sucesso, frequentemente resultando em atrasos, falhas, abandonos e rejeição de projetos de software (CHOW e CAO, 2007).

O foco de um projeto de software ágil é a entrega de valor para o cliente, nesse contexto, uma pesquisa sobre métodos ágeis, pela Shine Technologies (2003), mostrou que 93% (noventa e três por cento) das organizações citaram um aumento na produtividade. As metodologias mais usadas pelas empresas que responderam a pesquisa foi Scrum e XP, estas serão estudadas nesse trabalho. A mesma pesquisa ainda revelou que 83% (oitenta e três por cento) dos clientes afirmaram ter uma satisfação maior com o produto entregue.

Visto que os métodos de engenharia de software ágil surgiram recentemente como uma nova forma de se desenvolver software, seu sucesso é na sua maior parte anedótico (CHOW e CAO, 2007). Para entender e verificar esse sucesso, esse estudo visa à identificação de elementos de sucesso em um projeto de software para entender como os ensinamentos desse pensamento contribuem para o sucesso de projetos. Baseado em um estudo feito por Chow e Cao (2007), o presente trabalho foca como seu alvo de pesquisa equipes de desenvolvimento de software, ou seja, pessoas que aplicam os ensinamentos ágeis diariamente no seu trabalho para entregar software de qualidade para o cliente.

A pesquisa também fornecerá um *feedback* para os times de desenvolvimento da empresa sobre os fatores de sucesso percebido pelos próprios integrantes. O resultado dará uma visão sobre o que é considerado importante em seus projetos, e pontos que podem ser focados para ter sucesso em projetos de software ágeis. F*eedbacks* se alinham fortemente com o pensamento ágil, assim como a forma de trabalhar da empresa.

A questão a ser respondia nesse trabalho é: Quais os principais fatores que contribuem para o sucesso um projeto de software que usa metodologias ágeis? Por ano, são investidos milhões em software, e a tendência é os projetos ficarem cada vez mais caros e complexos, é necessária uma forma eficiente de desenvolver software que suporte as mudanças do modelo de negócios do mundo atual e com foco na entrega de valor para o cliente, para isso, é necessário analisar o que faz um projeto de software ter sucesso. A resposta dessa pesquisa será útil para guiar os investimentos das empresas na adoção de metodologias ágeis de software e na execução de projetos que usem esse pensamento como parâmetro para as atitudes do time de desenvolvimento.

3. Objetivos

3.1. Objetivo Geral

Identificar os fatores de sucesso que contribuem para o sucesso de projetos de software baseado em metodologias ágeis.

3.2. Objetivos Específicos

a. Identificar os fatores de sucesso em projetos de software baseado em metodologias ágeis;
b. Elaborar um instrumento para identificar os fatores de sucesso em projetos de software baseado em metodologias ágeis;

4. Referencial Teórico

4.1. Pensamento ágil

O pensamento ágil, também chamado de Agile, é uma filosofia que propõe uma nova forma de desenvolver software, esse pensamento traz novos valores para as equipes de desenvolvimento. O manifesto ágil, publicado em 2001, por Kent Beck e outros desenvolvedores, possui valores bem definidos, diferentes dos utilizados nos projetos de software anteriores. Segundo Pressman (2006, p. 59), Agile é uma filosofia que: "encoraja a satisfação do cliente e a entrega incremental de software logo de início; equipes de projeto pequenas, altamente motivadas; métodos informais; produtos de trabalho de engenharia de software mínimos e simplicidade global do desenvolvimento."

Outros autores colocam que Agile se trata de uma forma de pensar, pois manifesto ágil não fornece regras para serem seguidas pelo time, mas sim valores e princípios para guiar as práticas que o time irá usar no projeto. Segundo Larman (2003, p. 25), "ao invés de um conjunto de regras sobre diversos papéis, a organização do time, responsabilidades, relacionamentos, e atividades, o time e o gerente são guiados em primeiro lugar pelos princípios contidos no Manifesto Ágil" (tradução própria).

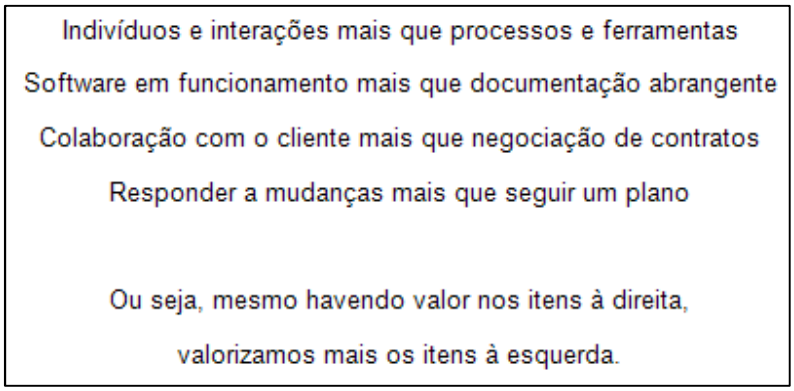


Figura 1: Agile Manifesto

Fonte: http://agilemanifesto.org/iso/ptbr/

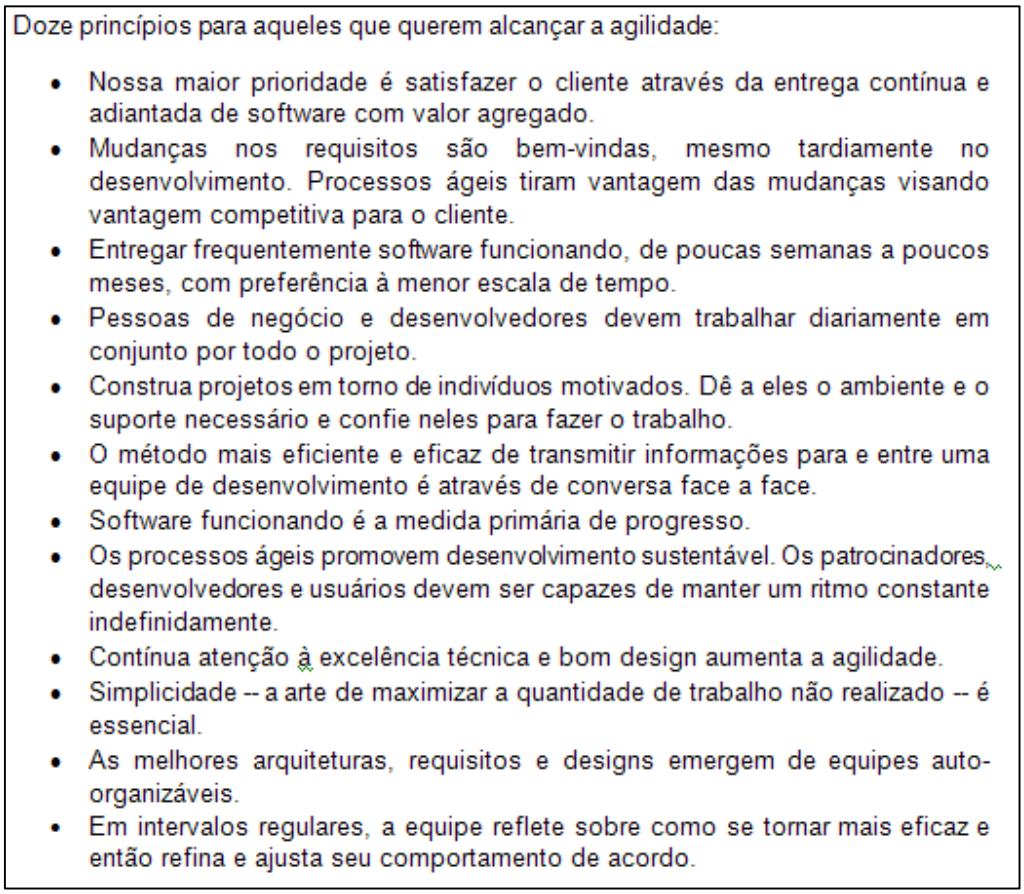
Indivíduos motivados, competentes, comunicativos, e com o desejo de fazer o melhor trabalho para o cliente são essenciais para um projeto de software de sucesso,

as pessoas são o ingrediente mais importante para o esse sucesso ser alcançado (MARTIN e MARTIN, 2006). A interação entre os membros do time também não pode ser subestimada, segundo Beck (2005) "problemas com projetos podem ser invariavelmente rastreados para alguém não falando com outro alguém sobre algo importante". O compartilhamento de informação e o trabalho em equipe são considerados pelo pensamento ágil tão importante que a habilidade de se comunicar, é tão, ou mais, valorizada que a capacidade técnica dos programadores. Um time de programadores medianos que se comunica bem tem mais chance de ter sucesso em um projeto do que um time de ótimos programadores que falham em interagir como um time (MARTIN e MARTIN, 2006). Vale ressaltar que : o pensamento ágil não exclui a existência de processo, apenas considera os indivíduos e a interação entre elas mais importantes.

A relação entre o time de desenvolvimento e o cliente deve ser de colaboração, e não de disputas contratuais (HIGHSMITH, 2004). A comunicação frequente com o cliente, para conhecer os rumos do projeto, é feita através de demonstrações periódicas ao longo do projeto, o cliente diz o que precisa ser modificado na aplicação que está sendo desenvolvida, assim, o cliente, no final do projeto, tem um produto que continuou crescendo e sendo modificado ao longo do projeto. O cliente é quase incorporado ao time, sendo tanto este como aquele responsáveis pelo sucesso do projeto. As conversas, preferencialmente são face a face, ou seja, o cliente conhece a equipe de trabalho e vice-versa, criando uma relação mais próxima entre as partes. Projetos de sucesso envolvem feedback frequente e regular do cliente. Em vez de depender de um contrato, ou uma afirmação de trabalho, o cliente do software trabalha junto do time de desenvolvimento, fornecendo *feedback* frequente sobre seus esforços (MARTIN e MARTIN, 2006).

A documentação é um subproduto da interação entre cliente e o time de desenvolvimento e esta deve ser uma ferramenta a mais de comunicação e não uma barreira (HIGHSMITH, 2004). Grandes projetos, que duram meses, até mesmo anos, coletando requisitos e um design de produto, são suscetíveis a falhas. Boas idéias são criadas, mas falham na hora de entrar no mercado, pois não recolhem *feedback* sobre o produto (HIGHSMITH, 2004). Tradicionalmente, a forma que o cliente especifica os

requisitos do software é através de uma extensa documentação, assim, o time de desenvolvedores precisa desenvolver o que está escrito. O problema nessa abordagem é que ela é estática, pois não considera as mudanças que podem ocorrer durante o projeto. Colaboração com o cliente, e a entrega contínua para avaliação em contraposição a análise de projeto realizam o mesmo papel de contratos e documentação respondem melhor a mudanças de requisitos que são constantes em projetos de software (MARTIN e MARTIN, 2006).


Doze princípios para aqueles que querem alcançar a agilidade:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Contínua atenção à excelência técnica e bom design aumenta a agilidade.
- Simplicidade -- a arte de maximizar a quantidade de trabalho não realizado -- é essencial.
- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Figura 2: Princípios por trás do Manifesto Ágil

Fonte: http://agilemanifesto.org/iso/ptbr/principles.html

Um fato corriqueiro no desenvolvimento de software é a mudança de requisitos, muitas vezes, partes do projeto são adicionadas, modificadas ou até mesmo retiradas. O modelo mais utilizado anteriormente, o modelo cascata, não suporta esse tipo de mudança depois do fim da fase de análise. Uma equipe ágil deve responder as

adaptações que o cliente deseja fazer no projeto antes da entrega final do produto, para isso, equipe deve criar demonstrações para o cliente, para este ver como o software irá funcionar. Segundo Pressman (pág 61), "uma estratégia de desenvolvimento incremental deve ser instituída. Incrementos de software (protótipos executáveis ou partes de um sistema operacional) devem ser entregues em curtos períodos de tempo de modo que a adaptação acerte o passo com as modificações.", ainda segundo o mesmo autor, "Essa abordagem iterativa habilita o cliente a avaliar o incremento de software regularmente, fornecer o *feedback* necessário a equipe de software e influenciar as adaptações do processo feitas para acomodar o feedback". Enquanto metodologias anteriores necessitavam de uma extensa documentação de especificações para o software, o método Ágil entrega pequenos incrementos no programa por vez, permitindo o cliente avaliar se o que está sendo feito é realmente algo que irá agregar valor ao seu negócio.

Uma equipe ágil realiza reuniões internas frequentemente para avaliar como está o seu desempenho, o que está bom e ruim e, adapta sua forma de trabalhar de acordo com o conhecimento adquirido na reunião. Conhecidas também como retrospectivas, essas reuniões tem como propósito discutir como estão às práticas do time e, caso seja necessário, modificá-las, seguindo um dos valores do manifesto ágil de responder a mudanças, esse valor não se refere somente a mudanças nos requisitos do sistema, mas também na estrutura do time. Os próprios métodos ágeis são adaptativos e necessitam de *feedback* para guiar e avaliar o seu desempenho e realizar mudanças para melhorar sua produtividade (LARMAN, 2003). Através da auto-reflexão e adaptação, um time ágil ajusta constantemente sua organização, regras, convenções, relacionamentos, e assim por diante. Um time ágil sabe que o ambiente está em mudança constante e sabe que deve mudar com o ambiente para continuar ágil (MARTIN e MARTIN, 2006).

4.1.1. **Scrum**

Larman (2003, p. 111) define Scrum como "um método iterativo e incremental que enfatiza um conjunto de valores e práticas de gerenciamento de projeto, ao invés que os [métodos] em requerimentos, implementações..." (tradução própria). Baseado na

teoria de controle de processos empíricos, também tem como fundamento os três pilares dessa teoria, transparência, inspeção e adaptação. Quando um processo é muito complicado para a abordagem definida, a abordagem empírica é a escolha apropriada. (tradução própria) (OGUNNAIKE apud MARTIN e MARTIN, 2006). A abordagem de processo empírico se aplica ao desenvolvimento de software, pois este é um processo complexo e ainda não se conhece uma forma única de agir que resulte sempre em sucesso. Para lidar com esse problema é necessário um processo empírico, flexível que pode ser modificado de acordo com as especificidades de cada projeto.

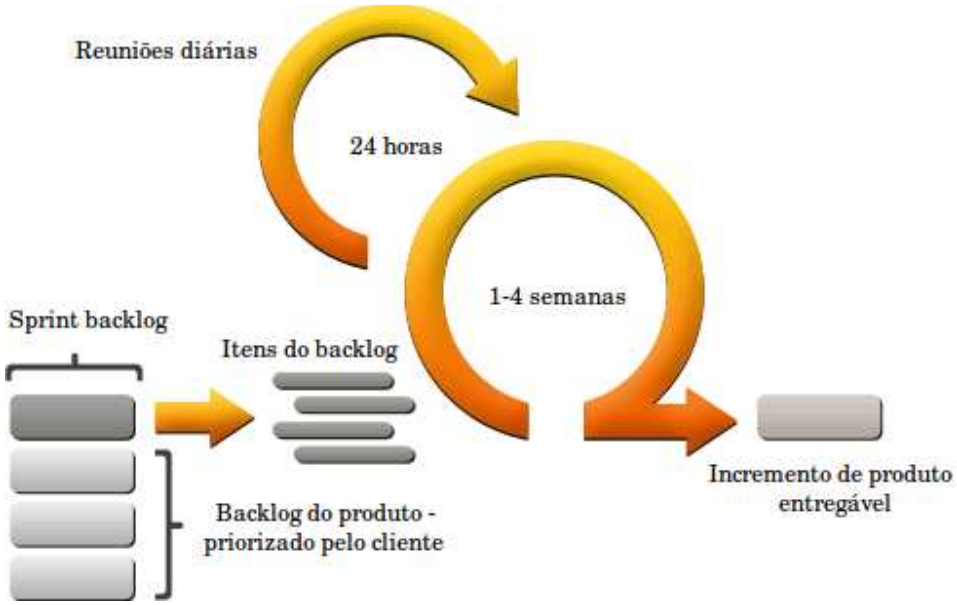


Figura 3: Ciclo de trabalho do Scrum

Fonte: Adaptado de: http://www.reaktor.fi/web/en/technology-and-research/scrum

Existem vários princípios do Scrum que são importantes, como o time auto-gerenciável, manter pequenos times, entre outros, mas esses não são tão relevantes no estudo do fluxo de informação de um projeto de software. As principais práticas, reuniões diárias e sprints, serão explicadas de forma mais detalhada.

4.1.1.1. Reuniões diárias

Essa prática consiste em se realizar reuniões diariamente com a equipe sempre no mesmo local e horário, originalmente, o nome dessas reuniões é "Stand-up meeting"

(tradução literal para o português: reuniões em pé, também chamadas de reuniões diárias), o motivo desse nome é que os membros participantes devem ficar em pé, essa prática tem como objetivo evitar o prolongamento da reunião, que, segundo Schwaber (2004) não devem demorar mais do que quinze minutos. Nessa reunião, cada membro deve responder três perguntas levando em consideração suas atividades no projeto:

- O que fez ontem?
- O que fará hoje?
- Tem algo o impedindo de trabalhar?

A reunião acaba assim que todos os membros do time responderem essas três perguntas.

Essas reuniões são um meio de compartilhar a informação entre os membros do time, o cliente também pode participar da reunião como ouvinte, assim, ele tem um status diário do andamento do projeto direto dos desenvolvedores, assim como todos os membros do time têm uma visão do andamento não apenas de suas tarefas, mas de todo o projeto. A disseminação do conhecimento ajuda a remover impedimentos, o nivelamento da equipe assim como aumenta a motivação e o desempenho geral do time. Por fim, as reuniões diárias permitem a inspeção frequente que é necessária nos métodos empíricos, e adaptação rápida caso seja preciso.

4.1.1.2. Sprint

Sprint é o ciclo de trabalho de uma equipe de software, também são conhecidos como iteração. Schwaber (2004) determina que a duração do Sprint seja de 30 dias, porém, com a evolução da metodologia, muitos times preferem realizar iterações mais curtas, com durações de três, duas, e há relatos de iterações que duram somente uma semana. Sprints com mais que 30 dias são considerados muito longos esse tempo é o máximo de tempo que pode ser utilizado sem que o time de desenvolvimento faça tanto trabalho que, o mesmo precise de artefatos e documentação para sustentar o seu processo de pensamento. (SCHWABER, 2004), esse também é o período de tempo máximo que os clientes irão esperar sem perder o interesse no progresso do time sem perder a crença de que este está fazendo um trabalho significativo para eles (ibidem).

Um ponto importante no Sprint é que tenha sempre a mesma duração para se criar um ritmo de trabalho dentro da equipe.

O Sprint começa com uma reunião de planejamento de Sprint, Schwaber coloca que essa reunião deve durar 8 horas, dividida em duas etapas iguais. Na primeira parte, o time define quais funcionalidades se comprometerão a desenvolver na iteração dentre as definidas como prioridade pelo cliente para o Sprint. Na segunda parte, o time analisa cada item que se comprometeu em desenvolver, separa em tarefas, e verificam se terão algum impedimento na execução de cada uma delas e, caso necessário, planejam e realizam ações para os problemas terem o mínimo impacto no decorrer do Sprint, o resultado dessa reunião é uma lista (chamada de Sprint Backlog) de funcionalidades e tarefas que o time trabalhará na próxima iteração (SCHWABER, 2004).

No final do Sprint são feitas duas reuniões. A primeira é a revisão de Sprint que, segundo Schwaber (2004), deve durar 4 horas. Nessa reunião, são mostradas as funcionalidades que foram adicionadas na iteração, Larman (2003, p. 120) coloca que "apresentações em "Power Point" são proibidas." - Tradução própria, o autor complementa que a ênfase da reunião é mostrar o produto. Enquanto apresentam as funcionalidades, os desenvolvedores respondem perguntas dos clientes sobre as mesmas e anotam mudanças requisitadas. Vale ressaltar que os próprios desenvolvedores apresentam o produto para o cliente, pois eles têm mais conhecimento sobre a funcionalidade e facilidade para responder a perguntas e entender as mudanças no software. Schwaber (2004) enfatiza que os clientes podem necessitar de mudanças nas funcionalidades, mesmo que elas tenham sido desenvolvidas corretamente de acordo com o que foi pedido, e adicioná-las como prioridades no projeto, assim como, a partir do que foi visto, eles podem perceber que precisam de novas funcionalidades, e, também podem adicioná-las como prioridade.

A segunda reunião consiste em uma retrospectiva do Sprint, segundo Schwaber (2004), deve durar no máximo 3 horas. Essa reunião serve como uma auto-reflexão do time, nela, o time considera o que funcionou bem na iteração, o que não funcionou bem e o que pode ser melhorado para trabalhar melhor na próxima iteração. Schwaber

(2004), afirma que retrospectivas que não resultam em mudanças são inúteis e frustrantes para o time.

4.1.2. **Extreme Programming**

Extreme Programming é uma metodologia de desenvolvimento de software que tem valores que se alinham com os valores do pensamento ágil. Os valores do *Extreme Programming* são: comunicação, simplicidade, *feedback*, coragem e respeito. Esses valores são base para as práticas dessa metodologia e, consideradas por Beck(1999), essenciais para o sucesso de um projeto de software. Segundo o mesmo autor, *Extreme Programming* se distingue das outras metodologias por, entre outros motivos, confiar na comunicação oral como um meio de comunicar a estrutura e intenção do sistema. *Extreme Programming* tem práticas mais técnicas e menos gerenciais, Larman (2004, p. 139), coloca que *Extreme Programming* "fornece métodos explícitos para programadores" (tradução própria). A maioria dessas práticas se refere especificamente ao desenvolvimento de software, ainda assim, o estudo dessas práticas facilitando entendimento de como uma equipe deve trabalhar para não ser um grupo de desenvolvedores trabalhando no mesmo projeto, mas sim, uma equipe de desenvolvimento.

O primeiro valor, a comunicação, é essencial para uma equipe aplicar XP. Seja a comunicação interna para o compartilhamento de conhecimento, através da programação em pares, onde o código produzido é sempre um trabalho feito por dois programadores juntos, seja pela comunicação externa, com o cliente trabalhando junto do time de desenvolvimento, sempre disponível para tirar dúvidas sobre o domínio do projeto. Além de aumentar o conhecimento de toda equipe sobre os diversos pontos da aplicação, o que contribui para a capacitação da equipe de trabalhar no projeto. A comunicação é importante para criar um sendo de time e de cooperação efetiva. (BECK, 2004).

Tendo em vista que é necessário um processo empírico para o desenvolvimento de software, a simplicidade se torna essencial para facilitar as mudanças recorrentes em projetos de software. "Adote a mudança ao invés de lutar contra ela" (tradução própria) (LARMAN, 2004, p. 153). A simplicidade não se entende apenas ao do

software que está em desenvolvimento, mas também a práticas que facilitam o funcionamento diário do time, como por exemplo, a utilização de *post-its* para descrições de tarefas e funcionalidades.

Mudanças são inevitáveis em um projeto de software, mas junto delas, surge a necessidade de *feedback* (BECK, 2004). O *feedback* ocorre diariamente na execução de testes pelos programadores sobre o software desenvolvido, também vem do cliente através de testes das versões do software criadas durante o desenvolvimento, para, assim, o time poder adaptar e melhorar a qualidade do produto (LARMAN, 2004). A comunicação é essencial para se obter feedbacks frequentes, visto que o resultado da comunicação é uma análise mais precisa das funcionalidades necessárias, e, assim, alterar o software para atender melhor as necessidades do cliente.

É necessário coragem para ter disciplina (HIGHSMITH, 2002, p. 171). Coragem é necessária para suportar os outros valores, e na capacidade do time de realizar as ações necessárias para resolver problemas que surgem durante o desenvolvimento do produto.

O último valor, respeito, é necessário para o bom relacionamento entre os membros do time e para o bom andamento do projeto. Se os membros do time não se importam uns com os outros e com o que estão fazendo, ou se o time não se importa com o projeto, *Extreme Programming* não irá funcionar (BECK, 2004).

Figura 4: Ambiente de trabalho XP

Fonte: http://www.extremeprogramming.org/rules/space.html

4.2. Modelos de elementos de sucesso em projetos de software ágeis

Durante a pesquisa, foram encontrados dois artigos que apresentam diferentes idéias de como definir os elementos de sucesso em um projeto ágil de software: Chow e Cao (2007) e Misra et al. (2009). Essa parte do trabalho visa analisar brevemente os artigos encontrados e justificar a escolha do artigo usado na presente pesquisa.

4.2.1. **Modelo de Chow e Cao (2007)**

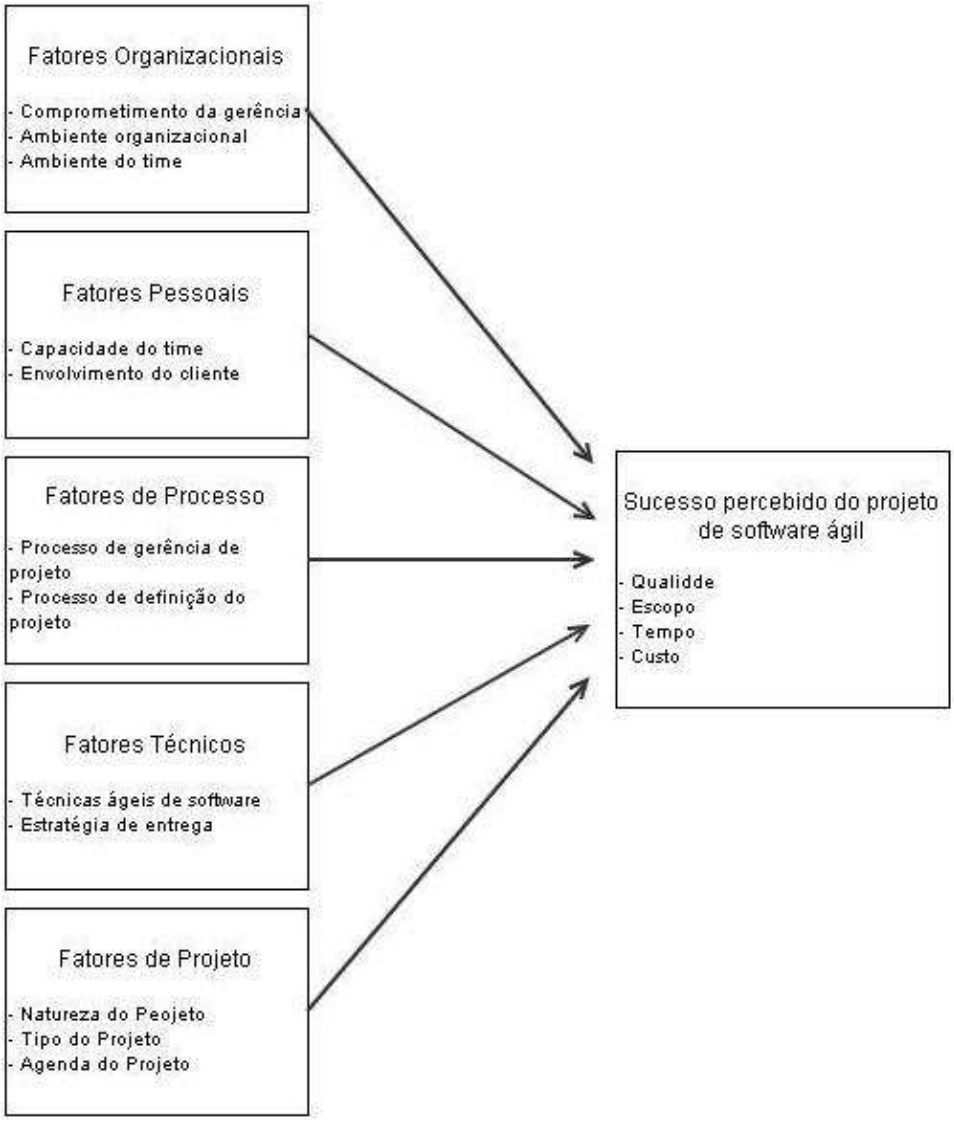


Figure 5: Modelo de Chow e Cao (2007)

Fonte: Chow e Cao (2007)

A pesquisa de Chow e Cao (2007), através de referencial teórico, primeiramente sumarizou os fatores de falha e de sucesso em projetos ágeis de software, dividindo esses fatores em cinco categorias: organizacional, pessoas, processo, projeto e técnico.

O artigo também apresenta quatro fatores de sucesso de projetos de software: qualidade, escopo, tempo e custo. Assim, o estudo analisa os elementos de sucesso referentes fatores de sucesso em projetos de software.

4.2.2. **Modelo de Misra et al. (2009)**

O artigo de Misra et al. (2009) tem suas hipóteses divididas em dois grupos: organizacional e pessoas. Esse estudo consiste em uma análise mais aprofundada sobre essas dimensões de projetos ágeis de software e analisa como cada fator contribui para o sucesso de um projeto de software.

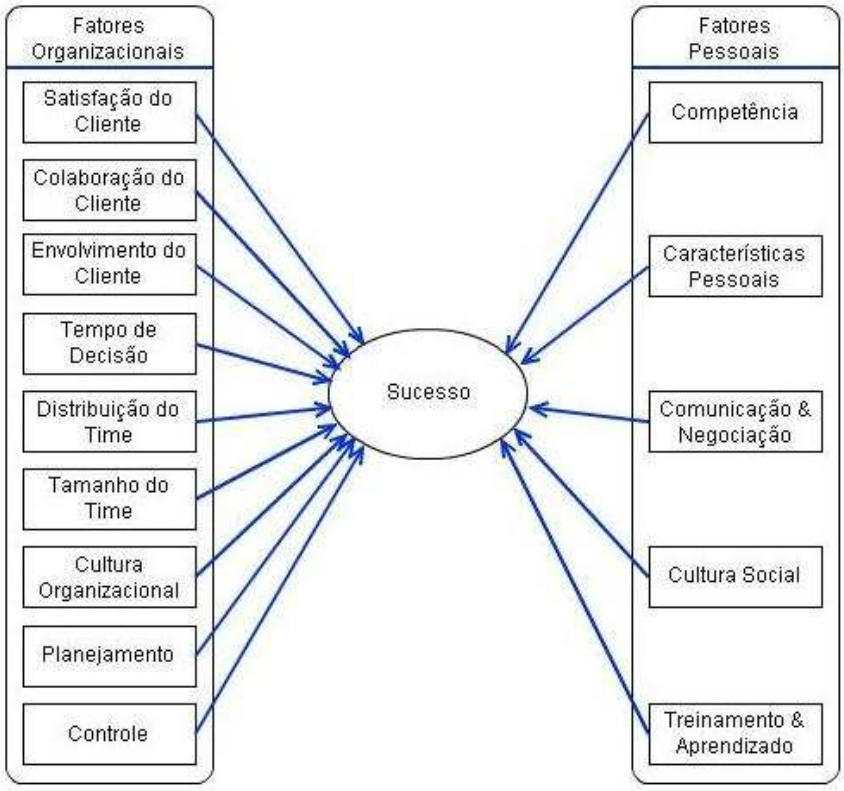


Figura 6: Modelo de Misra et al. (2009)

Fonte: Misra et al. (2009)

4.2.3. **Comparação entre os modelos de pesquisa**

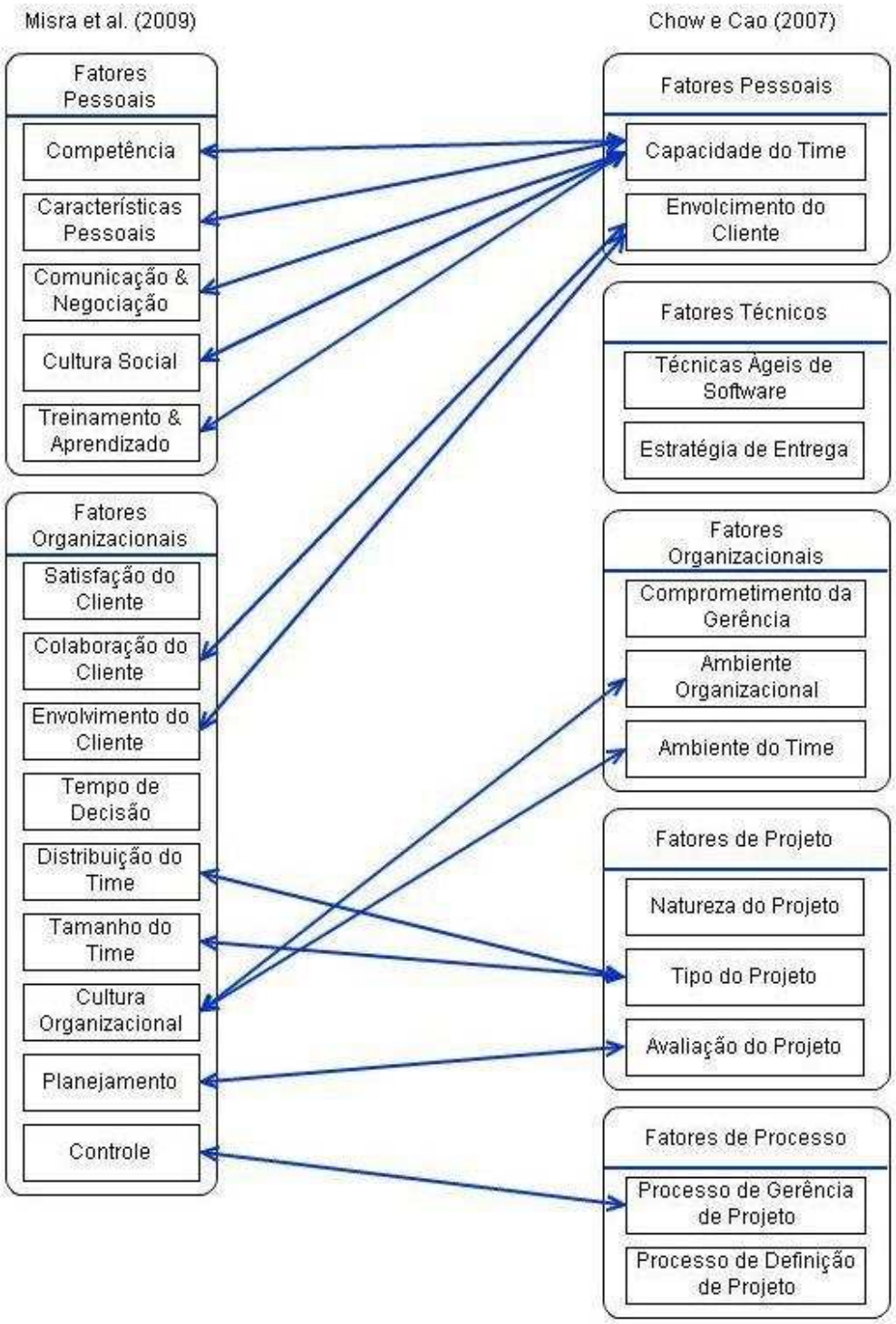


Figura 7: Comparação entre o modelo de Chow e Cao (2007) e Misra et al. (2009)

Fonte: Desenvolvido pelo autor

As pesquisas de Chow e Cao (2007) e Misra et al. (2009) apesar de apresentarem modelos diferentes, ambas possuem pontos que fazer referência a mesma atividade, mas sob um ponto de vista diferente. A figura 7 mostra a equivalência entre os fatores de sucesso encontrados pelas pesquisas.

4.2.4. **Escolha do modelo de pesquisa**

O artigo de Chow e Cao (2007) foi escolhido, primeiramente, por ter uma abrangência maior das hipóteses a serem testadas, se caracterizando como uma pesquisa mais superficial, porém que analisa diferentes elementos de projetos de software ágeis. O artigo de Misra et al. (2009) seria um ótimo modelo na continuação da pesquisa caso as hipóteses referentes a pessoas e organização sejam consideradas fatores de sucesso.

A possibilidade de análise da relação dos elementos de sucesso com diferentes fatores de sucesso de um projeto de software também teve influência na escolha do modelo de pesquisa a ser utilizado. Por Chow e Cao (2007) considerarem diferentes as influências dos elementos em cada fator de sucesso, em elemento pode ser preferida a outra dependendo da necessidade do projeto.

O modelo de Chow e Cao (2007) foi escolhido pois se alinhar melhor com os objetivos da presente pesquisa.

4.3. Elementos de sucesso de um projeto de software

Avaliar o sucesso de um projeto de software é uma tarefa complexa, pois o conceito de sucesso de um projeto de software é visto de diferentes maneiras por diferentes pessoas, dependendo do papel que ela exerce. Para o cliente, se todas as funcionalidades requisitadas estiverem implementadas ao final do projeto, este pode ser considerado um sucesso, porém o mesmo projeto pode estar com um custo acima do esperado, e ser considerado um fracasso pelo gerente. Beck (1999) menciona quatro variáveis que podem ser controladas e utilizadas como parâmetros para definir sucesso em um projeto de software, custo, tempo, qualidade e escopo.

No começo do projeto geralmente é definida a data de entrega do produto final, se o produto ainda está em desenvolvimento ao passar dessa data o projeto é

considerado um fracasso no ponto de vista de tempo, pois passou da estimativa feita. Situação parecida ocorre com o custo do projeto, se ao final o que foi gasto excedeu o valor estimado, o projeto é considerado um fracasso em termos de custo.

O sucesso do escopo de um projeto é variável, pois, como o pensamento ágil coloca, é normal existir variações no escopo durante o decorrer do projeto. As funcionalidades elaboradas no início do projeto sofrem mudanças, algumas são retiradas e, nas demonstrações do produto ao cliente, surgem novos requisitos. Isso leva as organizações a procurarem processos flexíveis capazes de se adaptarem a requerimentos voláteis. Mostrar progresso para o cliente e a gerência rapidamente e criar versões funcionais do software promovem esse comportamento (LINDVALL et al, 2004).

A qualidade de um software pode ser medida, principalmente, de dois pontos de vista, o externo, que se refere à percepção do cliente, ou usuários da aplicação, em relação ao produto entregue e a interna, que se a qualidade do código escrito pelos desenvolvedores. A qualidade interna não é vista pelo cliente final, por isso ela só é avaliada pelos programadores (BECK, 1999). Se a qualidade interna do software for baixa, o cliente levará mais tempo para ter as mudanças necessárias no seu software. Segundo Lindvall et al (2004), times de desenvolvimento de software enfrentam uma batalha contínua para aumentar a produtividade, ou seja, diminuir o custo e o tempo do projeto, enquanto mantém ou aumentam a qualidade interna do software.

Qualidade	Entregar um bom produto ou resultado do projeto
Escopo	Atender os requerimentos e objetivos
Tempo	Entregar no tempo estimado
Custo	Entregar com o custo estimado

Tabela 1: Fatores de sucesso de projetos de software

Fonte: Adaptado de: Chow e Cao (2007)

Sacrificar a qualidade interna do software para reduzir o *time to market* e esperar que a qualidade externa não seja afetada é uma estratégia que funciona muito bem a curto prazo. Porém, ela só será vantajosa por algumas semanas, no máximo meses. No

médio prazo, os problemas com a qualidade interna do software irão afetar o desenvolvimento, tornando este mais demorado e mais caro, ou, até mesmo, impossível de atingir um nível aceitável de qualidade externa (BECK, 1999).

4.4. Dimensões de fatores de sucesso em um projeto de software

Conhecimento técnico não é o suficiente para ter projetos de software de sucesso, é importante entender como se desenvolve sistemas usando a tecnologia da informação, mas também se deve considerar um ponto de vista mais abrangente: para se obter sucesso, é necessário muito mais do que apenas uma boa engenharia de software (BYTHEWAY, 1999). O manifesto ágil deixa isso bem claro em sua recomendação para dar mais valor aos indivíduos envolvidos no projeto e suas interações do que processos e ferramentas, porém, as interações da uma equipe de desenvolvimento de software não são apenas internas, elas se estendem ao ambiente em que a equipe está inserida. Nesse contexto, Chow e Cao (2008) dividem os elementos de sucesso em cinco categorias: organização, pessoas, processos, técnicos e de projeto. Lindvall et al (2004) complementa esse pensamento e coloca que para se ter um benefício pleno das práticas ágeis, as organizações têm que definir as interfaces entre um time de desenvolvimento ágil e o ambiente.

4.4.1. **Organização**

Uma cultura organizacional dinâmica e adaptativa a mudanças é considerada quase unamimamente por *experts* do pensamento ágil como um elemento necessário para o sucesso da introdução de práticas ágeis na organização (MISRA et al, 2009). Visto que é necessária colaboração de áreas da empresa que estão precisando de produtos de tecnologia da informação para automatizar tarefas, essas também precisam estar alinhadas com o pensamento ágil. O *feedback* e as necessidades dos usuários do sistemas, que atuam como o cliente do equipe de desenvolvimento, devem participar ativamente do projeto, e, para isso ter esse suporte de outras partes da organização, deve estar inserido na cultura organizacional um pensamento que aceite mudanças facilmente e que apóie a colaboração entre diferentes áreas da empresa para o bem maior da organização (ibidem).

O comportamento, as ações e as decisões das pessoas são influenciados pela estrutura social da organização em que estão inseridas, por sua vez, essa é afetada significativamente pela cultura organizacional (NERUR et al, 2005), sendo essa considerada um elemento de sucesso para projetos ágeis.

A cultura da organização influencia o comportamento das pessoas, se tiver uma cultura de cargos verticalizada, gerentes vão mandar, se tiver uma cultura de colaboração, gerentes vão colaborar. As metodologias ágeis precisam da uma mudança no papel de gerente, ao invés de comandar o time e controlá-lo, o gerente deve atuar como um líder e colaborar para o desenvolvimento do time (NERUR et al, 2005). Larman (2004) sustenta que o papel do gerente, em um projeto ágil de software, não deve ser de planejador e controlador, responsável por estruturar o time, fazer estimativas de tempos, agendar reuniões para o time ou atribuir responsabilidades, o gerente deve atuar como um facilitador que provê recursos para o time, mantém a visão do projeto, ajuda a remover impedimentos que impeçam o time de trabalhar e promover princípios ágeis. O gerente deve contribuir para o time tomar as decisões como um grupo, e dar liberdade para todos contribuírem com idéias que podem influenciar a decisão final, fazendo a tomada de decisão uma tarefa do grupo inteiro através da colaboração e do diálogo, e não por imposição. Times de sucesso de projetos ágeis normalmente têm liberdade para tomarem suas próprias decisões para contribuir para seu sucesso (MISRA et al, 2009).

A organização também deve apoiar o time e prover um bom ambiente de trabalho para este poder tomar decisões de forma mais eficiente, a área de desenvolvimento deve ser grande o bastante para todo o time (BECK, 2005), isto é, o time de desenvolvimento, e o cliente, que é a pessoa responsável por tirar dúvidas frequentes do time. Times co-localizados é um importante fator para uma comunicação eficiente (MISRA et al, 2009) que, por sua vez, é um dos principais valores da metodologia XP.

Ambientes produtivos contém quadros brancos, áreas formais e informais de reuniões, espaço para o time de desenvolvimento descansar, máquinas de café, assim, será criado um ambiente onde os desenvolvedores possam focar no seu trabalho e não se incomodar com problemas externos (REEL, 1999). O ambiente deve ser altamente

informativo, quadros brancos e paredes devem ser utilizados para transmitir informações. Um observador interessado deve conseguir entrar no ambiente de desenvolvimento e ter uma idéia geral do andamento do projeto, através da observação, em quinze segundos (BECK, 2005).

4.4.2. **Pessoas**

As pessoas envolvidas no projeto também são fatores de sucesso de um projeto de software, isso é evidenciado mais ainda em um projeto que utilize metodologias e o pensamento ágil, como dito no manifesto existe uma ênfase nos indivíduos e interação, colaboração com o cliente e responder a mudanças sugeridas por estes (MISRA et al, 2009).

A competência de um indivíduo para um projeto de software refere-se à experiência do indivíduo com a tecnologia que está sendo usada, se ele construiu sistemas na mesma área anteriormente e se ele possui boas habilidades interpessoais e de comunicação (MISRA et al, 2009). No quesito de contratar pessoas com excelência técnica, deve usar o princípio de Boehm "use pessoas melhores e menos pessoas" (COHN e FORD, 2003; MISRA et al, 2009), assim, o time pode ser mantido pequeno, o que se alinha com o pensamento ágil e manter a produtividade elevada, segundo Cohn e Ford (2003), a diferença de produtividade entre o melhor e o pior programador do time pode exceder a proporção documentada de dez para um.

O conhecimento de domínio também é importante, visto que alguém que já trabalhou em sistemas parecidos pode conhecer os problemas que podem surgir e como resolvê-los.

A cooperatividade caracterizada pela comunicação e colaboração entre os indivíduos que valorizam e confiam uns nos outros é crítico para o sucesso das metodologias ágeis (NERUR et al, 2005). Os indivíduos do time devem se comprometer a sempre ajudar seus colegas quando solicitados, assim como não podem ter receio de pedir ajuda ou de admitir quando está com dificuldade de resolver algum problema, a *stand-up* diária contribui bastante nesse processo, pois todos indivíduos do projeto sabem no que os outros estão trabalhando. Os membros de um time ágil devem estar sempre buscando aprender uns com os outros e valorizar a honestidade, a

colaboração, assim como assumir responsabilidade por funcionalidades e tarefas do projeto (MISRA et al, 2009).

As pessoas que terão suas vidas influenciadas pelo sistema que está em desenvolvimento devem ser parte do time (BECK, 2005), assim, elas terão conhecimento sobre as funcionalidades necessárias e mais facilidade para explicá-las aos desenvolvedores. Assim como os outros integrantes do time, o cliente deve ter um espírito de colaboração e sempre estar motivado a ajudar no desenvolvimento do software. O cliente que também deve ter um bom conhecimento sobre o sistema que está sendo desenvolvidos, ou seja, para conseguir orientar os desenvolvedores.

A função do cliente em uma equipe de desenvolvimento ágil inclui a explicação detalhada das funcionalidades para os programadores, a criação de critérios de aceitação junto com os testes e a definição de que uma parte do software está finalizada ou não (LARMAN, 2004). Essas tarefas necessitam que o cliente não esteja apenas presente na mesma sala que a equipe de desenvolvimento, mas que ele esteja altamente motivado, ativo e considere-se como um dos elementos responsáveis pelo projeto (MISRA et al, 2009). Quanto melhor for relacionamento entre o cliente e o time mais valor será agregado ao desenvolvimento (BECK, 2005).

4.4.3. **Processos**

Apesar dos processos serem consideradas menos importantes do que indivíduos e interação, eles não deixam de ser importante para o sucesso de um projeto de software. O processo de definição do projeto segundo, Highsmith (2004), consiste de:

- Coletar amplamente os requerimentos iniciais do projeto
- Definir o trabalho como uma lista de funcionalidades do produto
- Criar um plano de entrega do software
- Elaborar estratégias de redução de riscos para o projeto
- Estimar os custos do projeto e criar informações financeiras e administrativas necessárias

Em projetos ágeis, planos funcionam como guias, e não como uma forma de remover incertezas. Os requerimentos e funcionalidades iniciais funcionam como uma base para o andamento do projeto, os planos devem servir de suporte às mudanças

durante o decorrer dos projetos, e não impeli-las. Estimativas variam, o time varia, funcionalidades são alteradas, adicionadas e retiradas do projeto, o desenvolvimento gera nova informação que por sua vez cria a necessidade de re-planejar o projeto constantemente (HIGHSMITH, 2004).

A gerência do projeto durante todo o processo de desenvolvimento também é considerado um elemento para o sucesso. O processo deve ser alterado para se adequar às habilidades do time e às características do projeto ao invés de usar um rígido processo padrão igual para todos os projetos. Os modelos tradicionais de gestão são guiados por regulamentos e pela medição de progresso para dar segurança, já as metodologias ágeis acreditam na especulação e na incerteza para guiar um ambiente que seja flexível e se adapte rapidamente às mudanças existentes no andamento de um projeto de software (NERUR et al, 2005). Chow e Cao (2008) citam alguns princípios para se seguir que contribuem para o sucesso em projetos ágeis de software: foco na comunicação face a face diariamente; honrar os horários de trabalho, ou seja, sem horas extras; o cliente deve ter autoridade sobre as funcionalidades do sistema.

4.4.4. **Técnico**

As técnicas usadas pela equipe de desenvolvimento tem um papel crítico na implementação de softwares. Organizações que planejam adotar a metodologia ágil devem apoiar as técnicas ágeis que facilitam o desenvolvimento rápido e iterativo. (NERUR et al, 2005)

Um valor da metodologia *Extreme Programming* é a simplicidade, nesse contexto, o foco é resolver o problema atual, e não tentar resolver todos os problemas do sistema de uma vez. Manter o design da aplicação simples e fácil de fazer manutenção é um dos objetivos do XP. Fazer a coisa mais simples que pode funcionar é um trabalho árduo. Adquirir conhecimento sobre o projeto pode mudar a sua implementação futuramente, e isso é reconhecido pelos programadores ágeis que tentam deixar a aplicação o mais simples possível para que ela possa ser alterada posteriormente com mais facilidade (BECK, 2005).

A estratégia de entrega do software é importante para o projeto, tanto para a redução de riscos quanto para a coleta de *feedback*. A estratégia de entrega define

quantas entregas do software será feito durante o projeto e define funcionalidades para cada uma dessas entregas. Cada entrega deve ser o menor possível e conter as funcionalidades de maior valor para o cliente, além disso, o grupo de funcionalidades deve fazer sentido como um grupo. (BECK, 1999)

Em uma entrega, o time deve escolher aas funcionalidades que representam um maior risco para o projeto para serem desenvolvidas o quanto antes, a fim de diminuir o risco do projeto. Mesmo com essa restrição, o time deve desenvolver funcionalidades que tenha mais valor para o cliente, para evitar que estas sejam retiradas e adiadas para a próxima entrega. (BECK, 1999)

Com as entregas, o time recebe *feedback* do cliente que avalia se o que está sendo produzido é realmente o que ele deseja, visto que projetos de software sofrem mudanças constantemente, é preciso uma forma do cliente avaliar o que esta sendo desenvolvido pelo time. Essa avaliação é feita pelo cliente presente junto com o time de desenvolvimento, e em cada release há uma avaliação maior do cliente sobre as funcionalidades que estão sendo desenvolvidas e se elas atendem às necessidades do cliente. (HIGHSMITH, 2004)

4.5. Modelo de Pesquisa

O modelo de pesquisa foi baseado no modelo desenvolvido por Chow e Cao (2007). A tabela 2 mostra os elementos da pesquisa.

Fator de Sucesso	Dimensão de Sucesso	Elemento de Sucesso	Definição
Qualidade Escopo Tempo Custo	Organizacional	Comprometimento da Gerência	A gerência apóia as práticas ágeis na empresa
		Ambiente Organizacional	Cultura da organização, como ter conversas oralmente e uma cultura de cooperação
		Ambiente do Time	O time de desenvolvimento tem um ambiente adequado para trabalhar
	Pessoal	Capacidade do Time	O time tem profissionais competentes para realizar as tarefas
		Envolvimento do Cliente	O envolvimento do cliente no projeto e sua colaboração para o mesmo
	Processo	Processo de Gerência de Projeto	Ter um gerente com um pensamento ágil, que haja como um *coach*
		Processo de Definição de Projeto	O escopo do projeto está definido
	Técnica	Técnicas Ágeis de Software	Design simples, *refactoring*, pouca documentação
		Estratégia de Entrega	Entregas frequentes para o cliente dar *feedback* sobre o produto

Tabela 2: Elementos de sucesso de um projeto de software ágil

Fonte: Adaptado de: Chow e Cao (2007)

A pesquisa feita para definir os fatores de sucesso em projetos de software teve como base trabalhos de outros pesquisadores. A tabela 3 apresenta elementos da pesquisa que também são encontrados em outros trabalhos.

Elemento de Sucesso	Reel (1999)	Cohn e Ford (2003)	Highsmith (2002)	Beck (2004)	Larman (2004)	Lindvall (2004)	Nerur et al. (2005)	Chow e Cao (2007)	Misra et al. (2009)
Comprometimento da Gerência		X					X	X	
Ambiente Organizacional	X					X	X	X	X
Ambiente do Time	X			X	X			X	X
Capacidade do Time	X	X	X	X	X		X	X	X
Envolvimento do Cliente	X	X	X	X	X	X	X	X	X
Processo de Gerência de Projeto	X	X	X		X		X	X	X
Processo de Definição de Projeto			X					X	
Técnicas Ágeis de Software			X	X	X		X	X	
Estratégia de Entrega			X	X	X		X	X	

Tabela 3: Autores que citam os elementos de sucesso de um projeto de software ágil

Fonte: Desenvolvido pelo autor

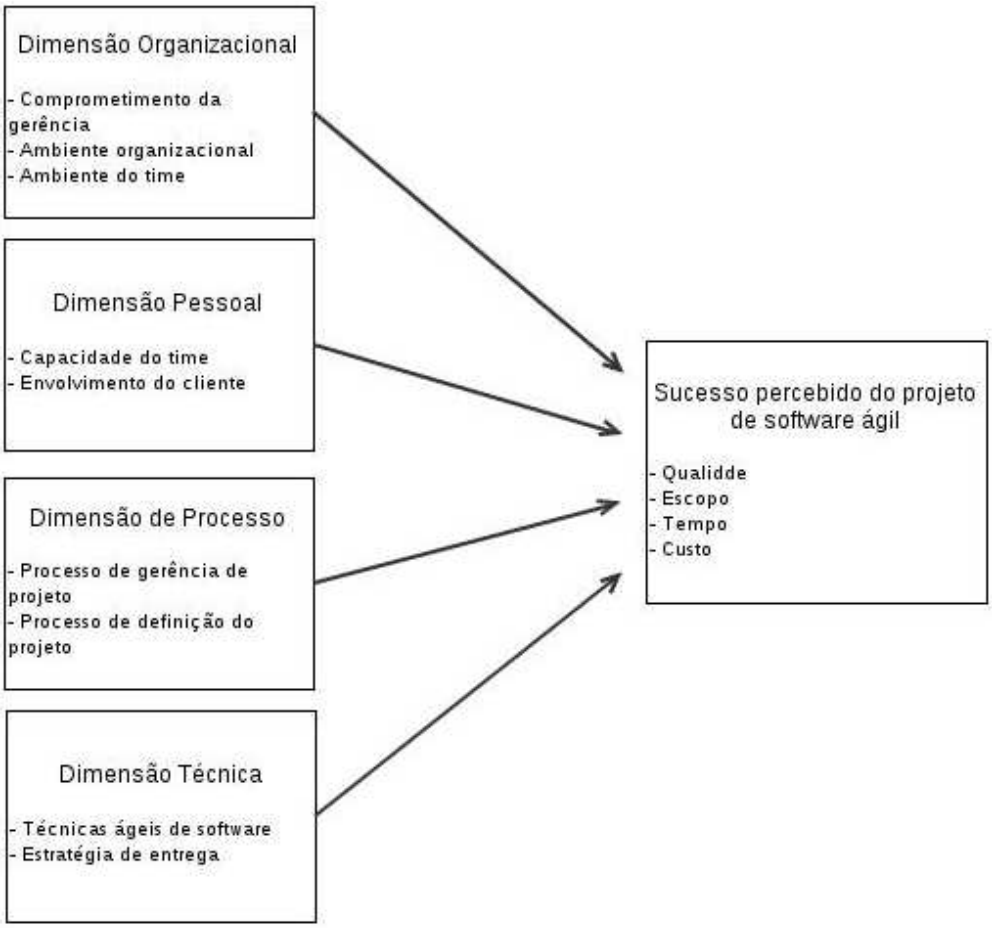


Figura 8: Modelo de pesquisa

Fonte: Adaptado de Chow e Cao (2007)

5. Método

5.1. Método de pesquisa

O método utilizado no presente trabalho é a pesquisa *survey*. De acordo com Collins e Hussey (2005, p. 70) “uma *survey* é uma metodologia positivista na qual uma amostra de sujeitos é retirada de uma população e estudada para se fazerem inferências sobre essa população”

Através da aplicação de uma survey, baseado no modelo de Chow e Cao (2007), a pesquisa procurou identificar a percepção sobre quais fatores de sucesso têm mais influência no sucesso de projetos de software desenvolvidos no mercado. A pesquisa possui um caráter descritivo, que busca medir os fatores que contribuem para o sucesso de um projeto de software ágil de acordo como a percepção de usuários desse pensamento. Essa avaliação será feita através de análises quantitativas em cima dos resultados so questionário.

5.2. Classificação da pesquisa

A presente pesquisa tem natureza quantitativa, do tipo *survey*. Para coletar os dados foi utilizado questionários (seção 5.5.2). Para a análise dos dados foram utilizadas técnicas estatísticas (seção 5.6).

5.3. Desenho da pesquisa

Para se chegar a conclusões concretas sobre o tema estudado, foram necessárias diferentes etapas, que, em conjunto, forneceram dados e informações relevantes para o aprendizado sobre o assunto escolhido.

A primeira etapa consistiu em uma busca bibliográfica, "a pesquisa bibliográfica é desenvolvida a partir de material já elaborado, constituído principalmente de livros e artigos científicos" (GIL, 2008, p. 50). Essa etapa da pesquisa pode ser classificada como exploratória que "é significativa em qualquer situação na qual o pesquisador não dispõe de entendimento suficiente para prosseguir com o projeto de pesquisa" (MALHOTRA, 2006, p. 100). “A pesquisa bibliográfica é desenvolvida a partir de

material já elaborado, constituído principalmente de livros e artigos científicos" (GIL, 2008; p. 50).

A segunda etapa, o instrumento de pesquisa foi adaptado para alinhar a pesquisa ao ambiente em que foi aplicada (seção 5.4).

A terceira etapa representa a escolha da população e a aplicação do questionário (seção 5.5).

A quarta etapa representa o tratamento estatístico dos dados finais, a análise estatística em cima dos dados e ao estudo e conclusões que podem ser tiradas desses dados.

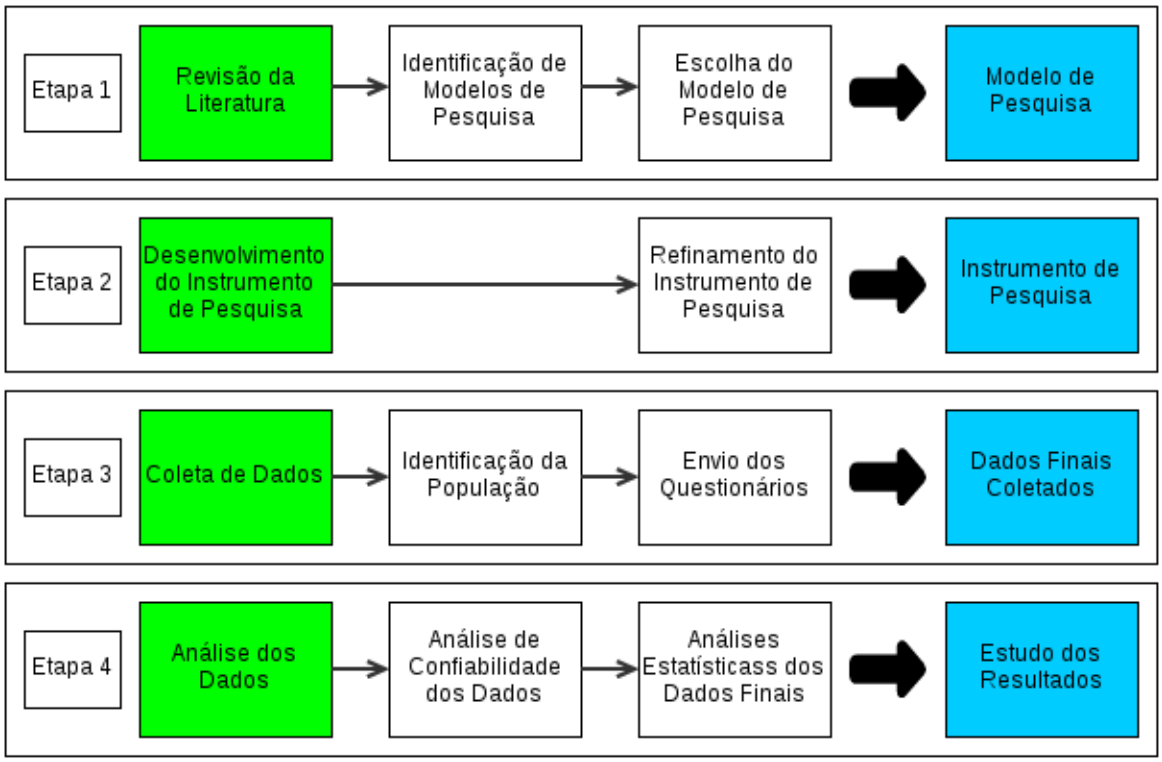


Figure 9: Desenho da Pesquisa

Fonte: Desenvolvido pelo autor

5.4. Justificativa da alteração do questionário

No questionário de Chow e Cao algumas perguntas que não eram aderentes ao público respondente da pesquisa e ao objetivo da pesquisa. Por esse motivo, as perguntas foram retiradas do questionário enviado, visto que ficariam fora de contexto na pesquisa com profissionais que aplicam a metodologia ágil diariamente.

A primeira hipótese retirada refere-se a importância de projetos ágeis de software sejam de pouca criticidade para a empresa que está desenvolvendo o software. A Thoughtworks não atua apenas como fornecedor de software para o seus clientes, mas sim como um parceiro para o desenvolvimento de software e, através da melhora da qualidade do software, o desenvolvimento da empresa no mercado. Sendo assim, não existe uma distinção entre projetos que sejam pouco importantes para a empresa ou projetos dos quais a empresa dependa para ficar no mercado.

As outras duas hipóteses que foram retiradas propõem que se limite a usar metodologias ágeis para projetos de escopo variável e para projetos dinâmicos e com uma agenda acelerada. O pensamento ágil suporta esses dois cenários, porém, diversos ensinamentos trazidos por esses novos valores também podem ser aplicados em projetos sem tantas adversidades, pois possui técnicas e metodologias que podem ser aplicadas independentemente da variação de escopo ou de agenda.

5.5. Coleta de dados

A coleta de dados é a aplicação do questionário. Nessa seção é descrita a amostra da pesquisa, como os questionários foram aplicados e o período dado para o público responder os questionários.

5.5.1. **Amostra**

Segundo Malhotra (2006, p. 320) amostra é "um subgrupo dos elementos da população selecionado para participação no estudo". No presente estudo, a amostra consiste dos funcionários da empresa que colaborou com a pesquisa, a Thoughtworks. Em conversa com colaboradores de dentro da empresa, foi decidido enviar o e questionário foi para quatro listas de e-mails a de desenvolvedores, a de testadores, a de analistas de sistemas e de gerentes de projetos.

5.5.2. **Questionário**

O questionário foi elaborado e disponibilizado utilizando o Google Docs, visto que este fornece uma ferramenta para montar os questionários facilmente e fornece uma página para os usuários responderem online.O processamento das respostas é

facilitado por essa ferramenta, quando o questionário é respondido os resultados foram colocados em uma planilha, o que automatizou o processamento de dados e facilitou a análise das respostas. A ferramenta também gera gráficos para facilitar a visualização das respostas.

O questionário online foi uma necessidade, pois a pesquisa engloba os funcionários de uma empresa que possui sedes em vários países do mundo, impossibilitando, assim, o pesquisador aplicar os questionários pessoalmente.

O questionário utilizou a escala Likert, de sete pontos, que está na figura 7.

Figura 10: Escala utilizada nos questionarios

Fonte: Dados do trabalho

Entre as vantagens dessa escala Malhotra (2006, p. 267) fala que "é fácil de construir e entender" e "os entrevistados entendem rapidamente como utilizar a escala, o que a torna adequada para entrevistas postais, telefônicas ou pessoais". No caso do presente trabalho, foi utilizada essa escala também pela facilidade de entendimento, visto que o formulário foi disponibilizado na internet para os respondentes.

5.5.3. **Aplicação da pesquisa**

O questionário foi enviado dia oito de setembro e sua aplicação consistiu de quatro dias, sendo os resultados recolhidos dia onze de setembro. O questionário foi enviado para quatro listas de e-mail internas de empresa, e obteve no total 30 respostas.

5.6. Análise dos Dados

A partir dos dados coletados, foram feitos análises de confiabilidade utilizando o Coeficiente Alfa Cronbach que é utilizado para medir a confiabilidade dos dados de uma pesquisa. Com dados mais confiáveis, foi utilizado a análise descritiva para chegar a

conclusões sobre os dados. A figura 10 apresenta os processos para validação do instrumento de pesquisa.

Para a realização dos testes estatísticos foi utilizado o software SPSS (do inglês Statistical Package for the Social Sciences).

Figura 11: Modelo para validação do instrumento de pesquisa

Fonte: Desenvolvido pelo autor

6. Análise dos Resultados

Para realizar uma análise mais detalhada dos dados, efetuou-se o processamento dos dados e as análises estatísticas separadamente para as perguntas referentes a cada fator de sucesso de projeto de software. O índice de Alpha Cronbach foi feito para descartar dados que não são confiáveis e posteriormente foi feita a análise descritiva.

Com relação ao fator "qualidade", para ter um índice de confiabilidade aceitável para os dados (maior que 0,7), foi necessário descartar dois elementos pesquisados, "capacidade do time" e "envolvimento do cliente". Assim, o índice ficou 0,723. A média dos elementos do fator ficou 5,37; e, dos sete constructos restantes, quatro ficaram acima da média, sendo a maior média a do elemento "ambiente do time", com média 6,56, seguido pelo constructo "ambiente organizacional", com uma média de 5,86. Os itens em destaque indicam que o ambiente que o time está inserido pode influenciar na qualidade do produto, um ambiente com espaço para o time se comunicar e trabalhar em equipe, como deve ser um ambiente ágil, é percebido como sendo um fator diferencial para a qualidade do software desenvolvido.

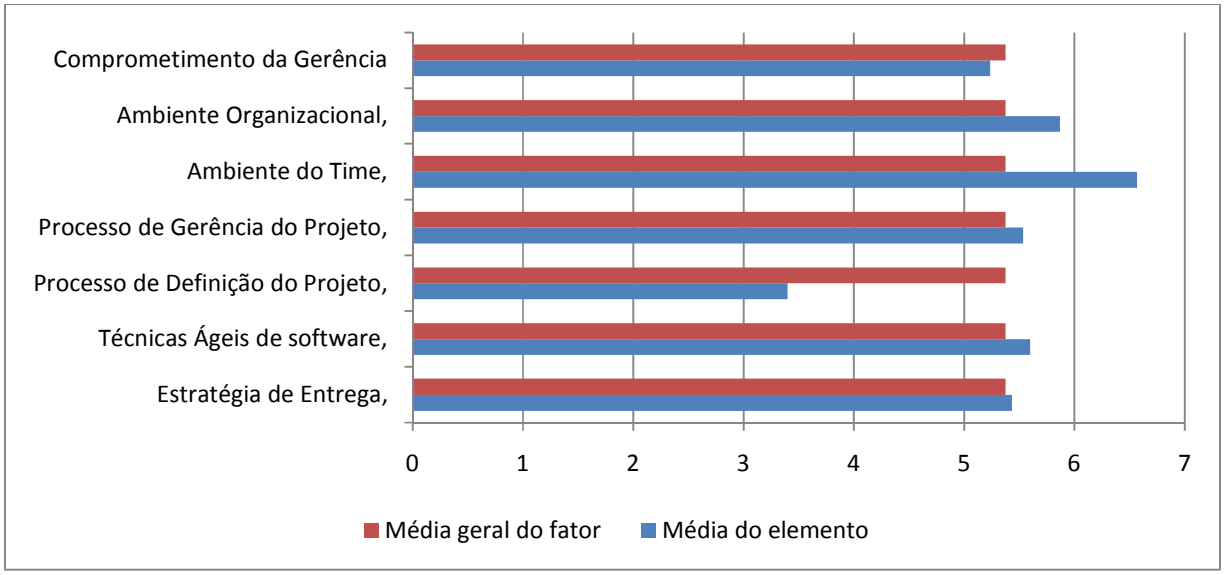


Figure 12: Média dos elementos do fator "qualidade"

Fonte: Desenvolvido pelo autor

O fator escopo obteve um índice de confiabilidade de 0,742 após a remoção de dois elementos, "comprometimento da gerência" e "processo de gerência do projeto". Os elementos que tiveram as maiores médias foram "envolvimento do cliente" e "ambiente do time", com 6,83 e 5,5 respectivamente. Dos sete elementos apenas três ficaram abaixo da média geral do fator, que foi de 5,21.

O destaque no envolvimento do cliente reflete o pensamento ágil, que prega que o cliente deve verificar cada pedaço de funcionalidade e averiguar se essa atende as necessidades do sistema. O escopo de projetos ágeis normalmente é flexível, ou seja, funcionalidades podem ser adicionadas ou retiradas durante o projeto de acordo com a necessidade do cliente, reforçando, assim, a idéia de que o envolvimento deste é um fator que influencia fortemente o escopo do projeto.

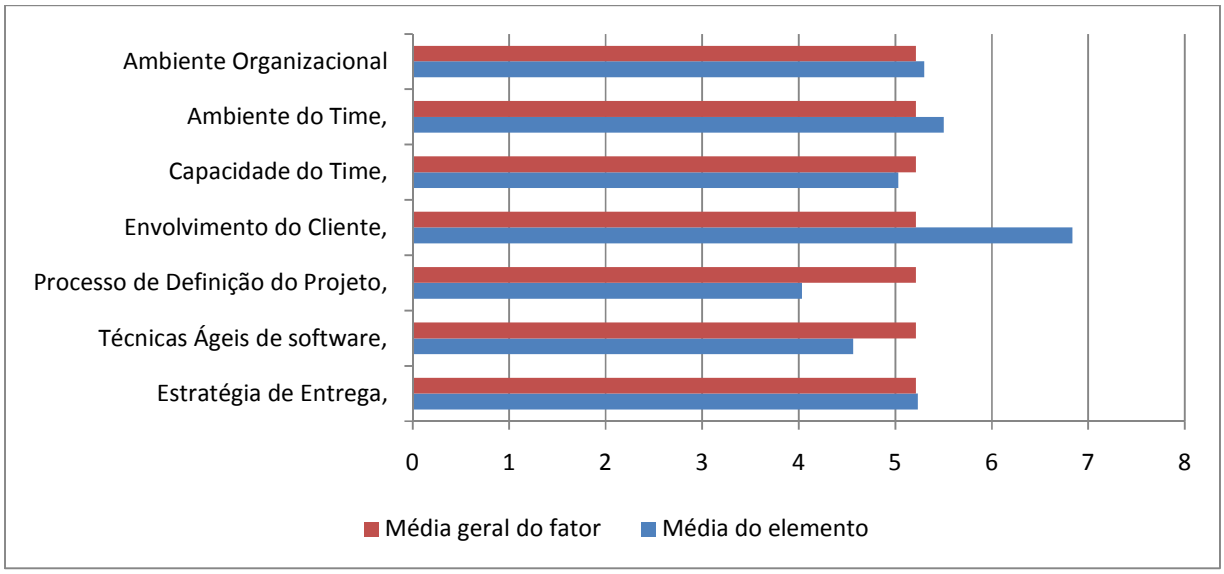


Figure 13: Média dos elementos do fator "escopo"

Fonte: Desenvolvido pelo autor

O fator tempo teve apenas um elemento removida para ter um índice de Alfa Cronbach aceitável para a presente pesquisa, o índice final ficou 0,726. Na análise descritiva do fator, três elementos se destacaram positivamente como fatores de sucesso: "envolvimento do cliente" com uma média de 6,4; "ambiente do time" com média de 6,06; e "capacidade do time" com média de 6,03. Dos oito elementos, apenas

três ficaram abaixo da média geral do fator (5,68), "ambiente organizacional" (5,66); "processo de gerência do projeto" (5,36); e "processo de definição do projeto" (4,33).

O elemento "envolvimento do cliente" também teve destaque positivo no fator tempo, visto que, como o cliente está disponível para solucionar dúvidas do time sobre como a aplicação deve ser, o tempo desperdiça criando funcionalidades que o cliente não precisa não existe no projeto. O elemento "capacidade do time" também merece destaque, pois teve destaque positivo apenas para o fator "tempo". Times capazes se alinham com o pensamento ágil visto que contribui ara uma elevada produtividade do time.

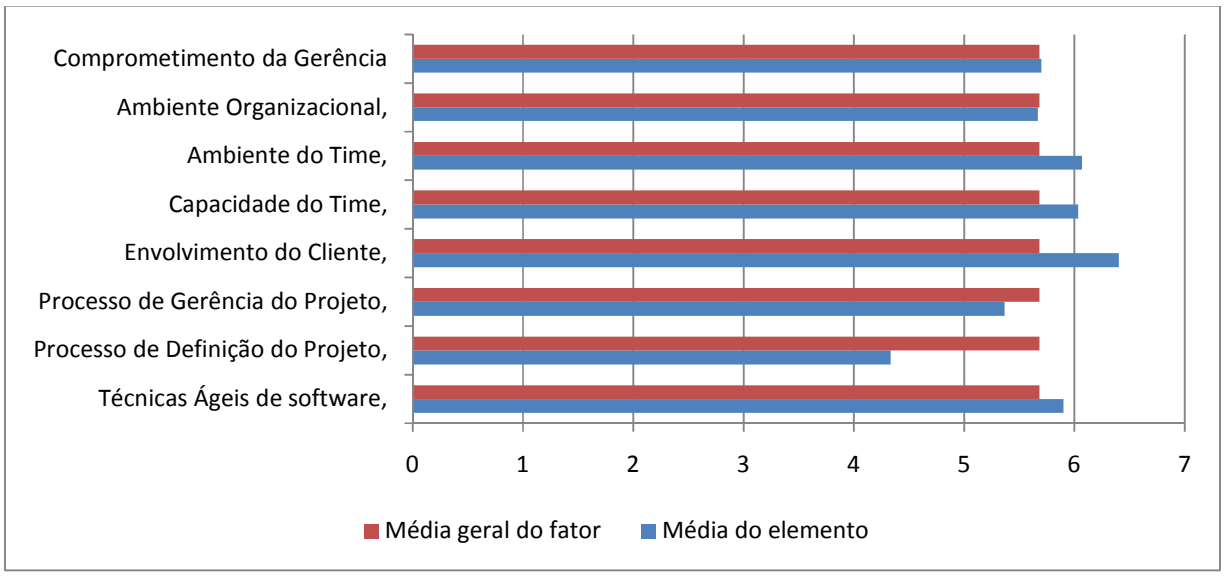


Figure 14: Média dos elementos do fator "tempo"

Fonte: Desenvolvido pelo autor

Assim como o fator tempo, apenas um elemento foi descartada para atingir a confiabilidade necessária para a pesquisa, que para esse fator ficou 0,712. Esse fator teve uma média geral de 5,32 com apenas três elementos acima da média, "envolvimento do cliente" (6,4) "estratégia de entrega" (5,7) e "capacidade do time" (5,4).

A estratégia de entrega e o envolvimento do cliente foram percebidos como elementos que contribuem para a diminuição do custo do projeto, visto que avaliações

frequentes do que está sendo desenvolvido podem ser feita, mantendo, assim, o time desenvolvendo apenas o que é necessário para o cliente.

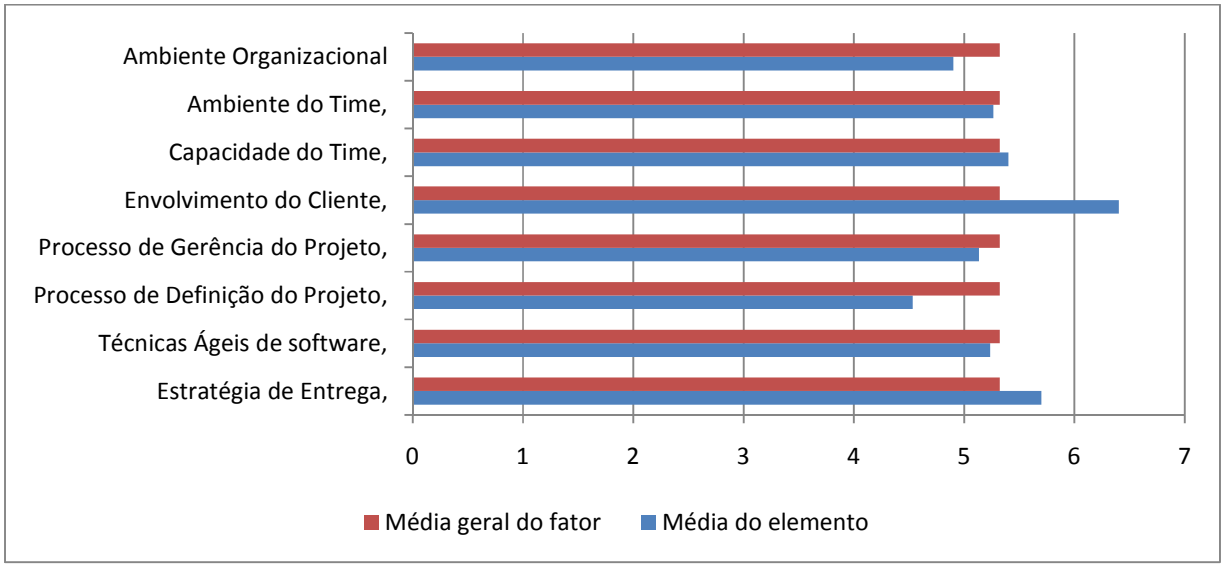


Figure 15: Média dos elementos do fator "custo"

Fonte: Desenvolvido pelo autor

Vale ressaltar que o elemento "definição do projeto" fica com uma média baixa em todos os fatores de sucesso o que enfatiza que o desenvolvimento de software ágil é um processo evolutivo e adaptativo, assim, a fase inicial não é considerado tão importante, pois a adaptação constante que é feita durante o desenvolvimento é o suficiente para o cliente ter o produto final desejado.

6.1. Conclusão

Na análise dos resultados, três elementos de sucesso se destacaram em relação aos fatores de sucesso de software mais de uma vez: "envolvimento do cliente" que ficou acima da média em "escopo", "tempo" e "custo"; "ambiente do time" que se destacou em "escopo" e "tempo"; e "capacidade do time" que ficou acima da média em "tempo e custo". Outros três elementos se destacaram apenas uma vez: "comprometimento da gerência", "processo de gerência do projeto" e "estratégia de entrega".

O elemento "envolvimento do cliente" condiz com o que é enfatizado no manifesto ágil "colaboração com o cliente mais que negociação de contratos" (Manifesto Ágil), pode-se notar que essa colaboração com o cliente é vista como um fator crítico de sucesso para projetos de software ágeis, visto que o desenvolvimento de software é um processo empírico e evolutivo.

Através dos resultados, também pode-se afirmar que o time de desenvolvimento é um elemento de sucesso, assim como o ambiente em que este trabalha, isso pode ser visto através do destaque dos elementos "ambiente do time" e "capacidade do time" em mais de um fator de sucesso de projetos de software. O pensamento ágil corrobora com os dados ao incentivar que o time de desenvolvimento seja composto de indivíduos motivados e competentes.

A gerência servil, como é chamado o gerente que está no projeto para ajudar a resolver problemas para que o time trabalhe com uma maior eficiência, também é considerada um importante elemento de sucesso para o projeto de software, porém, a gerência que o pensamento ágil prega é uma gerência que tem um papel diferente no projeto do que o time, e não apenas uma pessoa que está no projeto apenas para controlar os seus subordinados.

Por fim, o elemento "estratégia de entrega" é considerada um fator crítico de sucesso, esse fator também está presente no manifesto ágil que prega "software em funcionamento mais que documentação abrangente" (Manifesto Ágil), pois, software funcionando, é o produto final a ser entregue para o cliente.

6.2. Limitações da Pesquisa

Baseado nos dados coletados do questionário existe algumas limitações na pesquisa. Primeiramente, alguns elementos não puderam ser considerados na pesquisa devido à falta de confiabilidade dos dados, isso limita a pesquisa a uma análise menos precisa, por exemplo, o elemento "envolvimento do cliente" se destacou nos fatores "escopo", "tempo" e "custo", porém, devido a falta de confiabilidade dos dados não pode ser analisado referente a qualidade.

O baixo número de respondentes do questionário referente a população, visto que, com um número maior de respondentes se teria mais dados para analisar e para se chegar a conclusões sobre fatores de sucesso.

A terceira limitação é o alto índice de respondentes do Brasil (60%), visto que o questionário foi mandado para um público de nove países diferentes, assim, o resultado da pesquisa pode ser viesado ao pensamento do Brasil.

Bibliografia

ALTER, S. **Information systems:** a management perspective. 3. ed. San Francisco: Addison-Wesley, 1999.

BECK, K. **Extreme Programming Explained:** embrace change. 1. ed. Boston: Addison-Wesley, 2002.

BECK, K.; ANDRES, C. **Extreme Programming Explained:** Embrace Change, Second Edition. [S.l.]: Addison Wesley , 2004.

BOEHM, B. W.; PAPACCIO, P. N. Understanding and Controlling Software Costs. **IEEE Transactions on Software Engineering**, Washington, v. 14, n. 10, p. 1462-1477, Outubro 1988.

BOEHM, B.; TURNER, R. Management Challenges. **IEEE Software 22**, 2005. 30–39.

BYTHEWA, A. Successful software projects and how to achieve. **IEEE Software 16**, 1999. 15–17.

CHOW, T.; CAO, D.-B. C. A survey study of critical success factors in agile software projects. **The Journal of Systems and Software**, p. 961–971, 2007.

COHN, M.; FORD, D. Introducing an agile process to an organiza-. **Computer 36**, p. 297–334, 2003.

COLLINS, J.; HUSSEY, R. **Pesquisa em administração:** um guia prático para alunos de graduação e Pós-Graduação. 2. ed. Porto Alegre: Bookman, 2005.

CUNNINGHAM, W. **Manifesto para Desenvolvimento Agil de Software**, 2001. Disponivel em: <http://agilemanifesto.org/iso/ptbr/>. Acesso em: 8 Agosto 2010.

CUNNINGHAM, W. **Princípios por trás do Manifesto Ágil**, 2001. Disponivel em: <http://agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 9 Agosto 2010.

HIGHSMITH, J. **Agile Project Management:** Creating Innovative Products. [S.l.]: Addison Wesley, 2004.

LARMAN, C. **Agile and Iterative Development:** A Manager's Guide. Boston: Addison-Wesley Professional, 2004.

LINDSTRÖM, J. Reaktor Innovations Ltd. **Reaktor - Scrum**. Disponivel em: <http://www.reaktor.fi/web/en/technology-and-research/scrum>. Acesso em: 9 Agosto 2010.

LINDVALL, M. et al. Agile software development in large organizations. **Computer 37**, 2004. 26–34.

MARTIN, R. C.; MARTIN, M. **Agile Principles, Patterns, and Practices in C#**. [S.l.]: Prentice Hall, 2006.

MATTAR, F. N. **Pesquisa de marketing**. São Paulo: Atlas, 1993.

MISRA, S. C.; KUMAR, V.; KUMAR, U. Identifying some important success factors in adopting agile software. **The Journal of Systems and Software**, p. 1869–1890, 2009.

NERUR, S.; MAHAPATRA, R.; MANGALARAJ, G. Challenges of migrating. **The Journal of Systems and Software**, p. 961–971, 2008.

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

REEL, J. S. Critical success factors in software projects. **IEEE Software**, p. 18-23, 1999.

SCHWABER, K. **Agile Project Management with Scrum**. [S.l.]: Microsoft Press, 2004.

WELL, D. Give the Team a Dedicated Open Work Space. **Extreme Programming:** A gentle introduction. Disponivel em: <http://www.extremeprogramming.org/rules/space.html>. Acesso em: 14 Agosto 2010.

Anexo A – Questionário Aplicado

Success factors in Agile software projects

I am doing my paper about the success factors on Agile software projects and this form is part of my research. This questionnaire is an adaptation of the one used in a research done by Chow and Cao in 2008.

This first page collects some personal information, the following parts are questions about the success factors on software projects according to four different project attributes, quality, scope, time and cost (each of one containing 9 questions). The last page is a feedback about the survey.

Personal information

Primary Role

- ○ Developer
- ○ Quality Assurance
- ○ Business Analyst
- ○ Project Manager
- ○ Other:

Office location

- ○ United States
- ○ Brazil
- ○ Canada
- ○ United Kingdom
- ○ Australia
- ○ India
- ○ China
- ○ Germany
- ○ Sweden

Methods used in your current company

- ☐ XP
- ☐ Scrum
- ☐ Crystal
- ☐ Lean
- ☐ Other:

Continue »

Powered by Google Docs

Report Abuse - Terms of Service - Additional Terms

Success factors in Agile software projects

Quality research

This part of the questionaire is about the critical success factors of the project regarding the final quality of the software

1. The existence of a strong management commitment is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

2. The presence of agile-friendly organizational environment is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

3. The existence of agile-friendly project team environment is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

4. Having a team of high caliber is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

5. Having a strong customer involvement is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

6. The practice of agile project management process is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

7. The practice of a methodical project definition process is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

8. The practice of agile software engineering techniques is a critical success factor that contributes to the successful agile software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

9. The execution of a correct delivery strategy is a critical success factor that contributes to the successful agile process requirement management software development projects in terms of quality

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

« Back Continue »

Success factors in Agile software projects

Scope

This part of the questionaire is about the critical success factors of regarding the scope of the project

10. The existence of a strong management commitment is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

11. The presence of agile-friendly organizational environment is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

12. The existence of agile-friendly project team environment is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

13. Having a team of high caliber is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

14. Having a strong customer involvement is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

15. The practice of agile project management process is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

16. The practice of a methodical project definition process is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

17. The practice of agile software engineering techniques is a critical success factor that contributes to the successful agile software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

18. The execution of a correct delivery strategy is a critical success factor that contributes to the successful agile process requirement management software development projects in terms of scope

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

« Back Continue »

Success factors in Agile software projects

Time

This part of the questionaire is about the critical success factors to deliver the project on time

19. The existence of a strong management commitment is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

20. The presence of agile-friendly organizational environment is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

21. The existence of agile-friendly project team environment is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

22. Having a team of high caliber is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

23. Having a strong customer involvement is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

24. The practice of agile project management process is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

25. The practice of a methodical project definition process is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

26. The practice of agile software engineering techniques is a critical success factor that contributes to the successful agile software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

27. The execution of a correct delivery strategy is a critical success factor that contributes to the successful agile process requirement management software development projects in terms of time

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

« Back Continue »

Success factors in Agile software projects

Cost

This part of the questionaire is about the critical success factors to deliver the project on the estimated cost

28. The existence of a strong management commitment is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

29. The presence of agile-friendly organizational environment is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

30. The existence of agile-friendly project team environment is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

31. Having a team of high caliber is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

32. Having a strong customer involvement is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

33. The practice of agile project management process is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

34. The practice of a methodical project definition process is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

35. The practice of agile software engineering techniques is a critical success factor that contributes to the successful agile software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

36. The execution of a correct delivery strategy is a critical success factor that contributes to the successful agile process requirement management software development projects in terms of cost

	1	2	3	4	5	6	7	
Not Important	○	○	○	○	○	○	○	Important

« Back Continue »