GUILHERME AUGUSTO MACHADO DE ALMEIDA

Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências

Área de concentração: Engenharia de Produção

Orientador: Prof. Livre-Docente Roberto Marx

São Paulo

2017

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Almeida, Guilherme Augusto Machado de
Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis / G. A. M. Almeida -- versão corr. -- São Paulo, 2017.
105 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.Desenvolvimento de software 2.Metodologias ágeis 3.Metodologias tradicionais de software I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

Nome: ALMEIDA, Guilherme Augusto Machado de

Título: Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências

Aprovado em:

Banca Examinadora

Prof. Dr. _____Instituição:_____

Julgamento: _____Assinatura:_____

Prof. Dr. _____Instituição:_____

Julgamento: _____Assinatura:_____

Prof. Dr. _____Instituição:_____

Julgamento: _____Assinatura:_____

Aos meus pais, pessoas vitoriosas a quem admiro muito e tenho gratidão maior ainda.

AGRADECIMENTOS

Ao Prof. Dr. Roberto Marx, que desde o primeiro momento acreditou no meu potencial, correspondeu às minhas necessidades, compartilhou de seu conhecimento e me incentivou sempre.

Ao Prof. Dr. André Fleury e ao Prof. Dr. Mauro Spínola, pelos elogios, questionamentos, críticas e atenção que muito agregaram ao direcionamento final deste trabalho, no momento da qualificação.

Ao Prof. Dr. Mario Salerno, ao Prof. Dr. Fernando Laurindo, ao Prof. Dr. Davi Nakano, e novamente ao Prof. Dr. Mauro Spínola e ao Prof. Dr. Roberto Marx, pela dedicação às disciplinas, sendo que todas, de alguma fora, despertaram e mantiveram minha motivação a cada aula e / ou estudo sobre seus temas.

À Escola Politécnica, pela oportunidade de realização do curso de mestrado e disponibilização de diversos recursos que colaboraram durante o desenvolvimento do curso.

Aos participantes das entrevistas utilizadas nesse trabalho, que têm seus nomes preservados, mas que colaboraram com grande boa vontade.

Aos meus colegas de classe, por terem sempre sido participativos e pelo sentimento de identidade e conversas que diversas vezes foram fortalecedoras.

RESUMO

ALMEIDA, G. A. M. **Fatores de escolha entre metodologias de software tradicionais e ágeis**. 2017. 105 f. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2017.

A escolha entre o uso de metodologias ágeis ou metodologias tradicionais de desenvolvimento de software continua sendo amplamente discutida em vários aspectos, sendo um deles a presença ou ausência de certos fatores que precisam ser identificados para que as aplicações das metodologias sejam realizadas com sucesso. Neste estudo, tanto metodologias ágeis quanto tradicionais são discutidas através da literatura desde seu surgimento, histórico e evolução, até estudos comparativos entre ambas e outros com evidências empíricas, embora ainda haja a necessidade de estudos deste último tipo para o domínio. Com o intuito de avaliar as adequações dos tipos de metodologia para os diferentes cenários que uma organização ou projeto pode apresentar a partir dos fatores encontrados na literatura, foram realizadas entrevistas e questionários com pessoas envolvidas em desenvolvimento e definição de processos de desenvolvimento de software em um estudo de caso em empresa selecionada para a obtenção de mais evidências empíricas sobre o tema. Com os resultados obtidos, identifica-se então entre os fatores estudados quais são habilitadores e quais são inibidores para cada tipo de metodologia, propondo-se um modelo para a escolha de um ou de outro tipo a partir da presença ou ausência destes fatores nos cenários de aplicação das metodologias.

Palavras-chave: Desenvolvimento de software. Metodologias ágeis. Metodologias tradicionais de software.

ABSTRACT

ALMEIDA, G. A. M. **Factors of choice between traditional and agile software development methodologies**. 2017. 105 f. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2017.

The choice between agile or traditional software development methodologies continues to be widely discussed in several aspects, being one of these aspects that certain factors presence or absence must be identified for methodologies usage to be successful. In this study, both agile and traditional methodologies are discussed on the domain literature from their emergence, historical facts and evolution, to comparative studies and empirical evidences obtained, despite there is still need for studies on this last subject for the domain. In order to evaluate adequacy for both types of methodologies to the different scenarios in which an organization or project may occur considering the factors appointed by literature, interviews and surveys where done with key people involved in software development or software processes in a case study in a selected company for more empirical evidence achievement. Then it is possible to identify between the factors which are enablers and which are inhibitors for each type of methodology, then purposing a model for the choice between the two types from the presence or absence of these factors in the scenarios for their uses.

Keywords: Software development. Agile methods. Traditional software development methods.

LISTA DE TABELAS

LISTA DE FIGURAS

Sumário

1 Introdução

1.1 Contexto, problema e importância do trabalho

O uso de metodologias ágeis e metodologias tradicionais de desenvolvimento de software continua sendo amplamente discutido em vários aspectos, sendo que, entre estes grupos de metodologias, as ágeis são mais recentes historicamente, assim como o conhecimento e evidências que se têm sobre elas e suas aplicações.

Cada um desses tipos de metodologia possui premissas específicas para uso e buscam atingir algumas vezes objetivos em comum e, em outras vezes, objetivos distintos. Também apresentam diferentes resultados em suas aplicações, dependendo dos cenários em que são observados os seus usos.

Os tipos de metodologia são aqui tratados nestes dois grupos, conforme comumente descritos na literatura:

- Metodologias tradicionais de desenvolvimento de software - metodologias que apresentam, entre outras características, fases bem definidas de ciclo de vida de desenvolvimento de software, maior documentação sobre o produto gerado e suas fases de desenvolvimento, maior formalização da comunicação através de documentação, papéis bem definidos para os membros da equipe e busca de padronização das etapas de desenvolvimento;

- Metodologias ágeis de desenvolvimento de software - metodologias que apresentam, entre outras características, etapas iterativas (vários ciclos incrementais) de desenvolvimento de software, menor documentação e maior foco no desenvolvimento do software ou sistema, comunicação mais informal entre os membros da equipe, papéis menos definidos ou rotação de papéis entre os membros e maior aceitação a mudanças que possam ocorrer durante o processo de desenvolvimento.

Sobre uma das diferenças apresentadas e também sobre o termo utilizado para o segundo tipo de metodologias, Erickson, Lyytinen e Siau (2005) definem agilidade, em seu núcleo, como “retirar quanta carga (...) seja possível para promover resposta rápida em ambientes mutáveis a mudanças em requisitos dos usuários, a prazos acelerados dos projetos e fatores relacionados”. Onde carga se refere a documentação, artefatos gerados, tarefas e processos no trabalho.

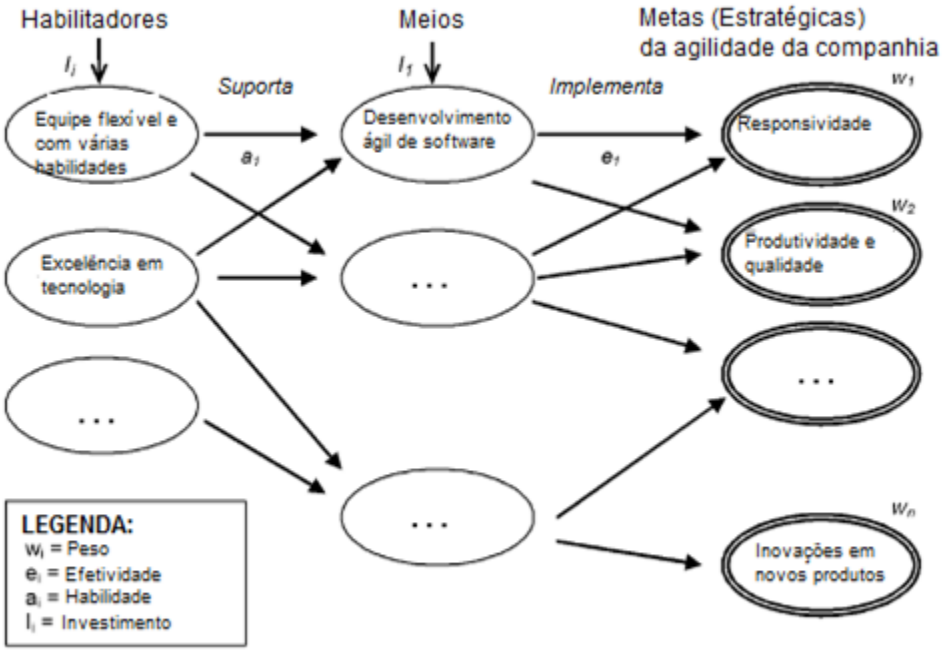
Laanti, Salo e Abrahamsson (2011) encontraram em seu trabalho que “metodologias ágeis vieram para ficar, não sendo apenas uma onda temporária”, sustentando a importância de estudos acerca do tema. Ainda assim, esse tipo de metodologia é assunto de muita discussão e apresenta pouca evidência empírica, como concluem Dybå e Dingsøyr (2008) em sua revisão sistemática da literatura.

Sobre o termo ágil, Kettunen e Laanti (2008) citam que, na área de manufatura, o conceito de agilidade é muito mais abrangente do que em desenvolvimento de software, pois cobre também produtos ágeis, ambientes ágeis e empresas ágeis, enquanto em desenvolvimento de software, a agilidade está principalmente concentrada nos processos de projeto de software, onde o termo foi adotado independentemente. Os autores também citam que é perceptível, na área de manufatura ágil, que a ideia de tornar apenas os processos ágeis, sem se pensar no negócio ou no produto gerado, faz pouco sentido e inverter este pensamento leva à conclusão de que pensar nos “habilitadores” da agilidade presentes na organização e no produto gerado seria muito benéfico para qualquer companhia de software que deseje utilizar metodologias ágeis. Habilitadores serão aqui tratados como os fatores que compõem os cenários nos quais as metodologias estão aptas a serem aplicadas.

Também de acordo com Kettunen e Laanti (2008), diferentes companhias, ou ainda a mesma companhia sob diferentes circunstâncias, podem dar “pesos” diferentes às diferentes “metas” da agilidade e, tipicamente, há muitos “meios” para que estas sejam atingidas, assim como estes meios são suportados por diferentes fatores habilitadores. Com isto estabelecido, os autores apresentam um esquema gráfico mostrando que, em uma organização de grande porte de desenvolvimento de novos

produtos, o desenvolvimento de software é apenas um elemento contribuindo para as metas da agilidade para a empresa em sua totalidade (figura 1).

Figura 1 - Alguns componentes para alcançar a agilidade em companhias de desenvolvimento de novos produtos.



Fonte: Kettunen e Laanti (2008)

Sircar, Nerur e Mahapatra (2001) apud Nerur, Mahapatra e Mangalaraj (2005) indicam que mudanças no processo de desenvolvimento de software, portanto nas metodologias, representam complexos fenômenos de mudança e que não podem ser realizadas somente pela substituição de ferramentas e tecnologias. Consequentemente, devem ser observados outros aspectos relacionados aos habilitadores, meios e metas das companhias, justificando a observação dos fatores que compõem os cenários onde serão aplicadas as metodologias de desenvolvimento de software. Neste trabalho, as metodologias são os meios para se atingirem as metas.

Silva et al (2011) apresentam um estudo sobre a integração entre metodologias ágeis e linhas de produção de software (SPL) - sendo este um modelo de desenvolvimento com mais características das metodologias tradicionais - onde reconhecem que ambas são soluções efetivas para lidar com a complexidade

crescente dos softwares e com as necessidades competitivas das organizações que os desenvolvem além de que, apesar das diferentes abordagens de cada tipo, ambas propiciam benefícios como melhorar a produtividade, reduzir o *time-to-market*, diminuir os custos de desenvolvimento e aumentar a satisfação do consumidor. Assim, as obtenções desses e de outros benefícios podem ser metas como as do modelo apresentado.

Zahran (1998) apud Abrahamsson (2000) argumenta que um dos maiores obstáculos à adoção de melhorias de processo de software é a relutância do gerenciamento do negócio em investimentos e que este obstáculo vem da falta de informação confiável sobre os benefícios dessas melhorias, porém cita que ao se alinharem as metas dessas melhorias às da organização pode-se trazer maior aceitação e, então, investimento, onde apresenta como exemplos os benefícios também citados por Silva et al (2011). Ambos os trabalhos justificam a oportunidade de reconhecer que o conhecimento e uso das duas abordagens, tradicional e ágil, pode trazer benefícios. O resultado deste trabalho pode ser utilizado para atender a necessidade de informação confiável sobre os benefícios do uso dos dois tipos de metodologias e das melhorias que possam ser necessárias para que estes sejam melhor aplicados, dados os fatores presentes em cada cenário observado.

Complementando sobre as melhorias, Kettunen e Laanti (2008) notam que para se estar apto a realmente implementar as metas do desenvolvimento ágil de software, muitos habilitadores e outras condições, algumas externas à área de desenvolvimento de software, precisam estar presentes, caso contrário é difícil (ou até impossível) atingir a agilidade de forma sustentável. Assim, a presença ou características de certos fatores precisam ser identificadas e, também, o quanto estes permitem melhor aplicabilidade, ou não, das metodologias ágeis de desenvolvimento de software. O mesmo também deve ser observado para as metodologias tradicionais e ambas serão abordadas neste estudo.

Sobre tais condições, Nerur, Mahapatra e Mangalaraj (2005) citam que, de fato, há características que diferenciam os dois tipos de metodologias, como o desenvolvimento ágil ser dependente de trabalho em equipe, de forma oposta à

atribuição de papéis individuais, que caracteriza o desenvolvimento tradicional. Kettunen e Laanti (2008) citam também que os modelos de processos de software ágeis são baseados em certas premissas fundamentais, mas frequentemente ocultas, sobre times de projetos, produtos a serem desenvolvidos e ambientes de negócios com a participação de clientes, assim as organizações precisam considerar a validade destas premissas em seus cenários.

Silva et al (2011) citam que há crescente interesse sobre se a integração entre ágil e SPL pode fornecer mais benefícios e resolver questões em torno de desenvolvimento de software, pois um balanço entre ambos pode trazer interessantes possibilidades para uma nova abordagem: de lidar com as mudanças solicitadas pelos clientes, de gerenciar a variabilidade de produtos e de alavancar as oportunidades de reuso de software, sendo todos esses benefícios já verificados para o desenvolvimento de software e também são resultados, nesta ordem, do uso de metodologias mais ágeis para mais tradicionais. Assim, tanto metodologias ágeis quanto tradicionais serão abordadas para avaliação das suas adequações de uso para os diferentes cenários e também para atender a esse interesse sobre o uso de ambas.

Em sua conclusão, Kettunen e Laanti (2008) recomendam que futuros estudos elaborem o modelo para os fatores que influenciam a agilidade (figura 1), examinem a correlação entre metas, meios e habilitadores e abordem a questão do “por que adotar metodologias ágeis em times de desenvolvimento de software em grandes organizações algumas vezes falha?”. Entre outros pontos, indicam que uma importante extensão é incluir os “orientadores” da agilidade (os fatores de turbulência ou quão ágil as organizações são ou precisam ser) em adição às metas, meios e habilitadores no modelo. Hilkka, Matti e Tuure (2005) também reforçam que a seleção de uma metodologia deveria focar nesses aspectos, ao invés de discutir simplesmente qual método é o melhor.

Esse estudo se apoia sobre estas sugestões para propor um modelo de escolha ao final, mas, primeiramente, elicita e trata os fatores que compõem os cenários de aplicação das metodologias como habilitadores ou orientadores destas, tanto para

as metodologias ágeis quanto para as metodologias tradicionais. A comparação entre ambos os tipos de metodologia dar-se-á sob os mesmos fatores, podendo então se ponderar os cenários de acordo com suas adequabilidades para cada tipo.

Boehm e Turner (2003) apresentam em seu livro cinco fatores críticos relacionados às áreas de origem e aplicação dos dois tipos de metodologia para situações de escolha entre algum deles para uso, sugerindo que o equilíbrio entre ambos os tipos seja importante para o sucesso dos projetos de desenvolvimento. Estes fatores também serão aqui abordados e espera-se encontrá-los entre aqueles com os maiores pesos entre os levantados. Contudo, mais fatores serão incluídos, dadas as suas recorrências e importâncias na literatura.

Por fim, Abrahamsson (2000) indica que é de maior importância estudar e entender os fatores subjacentes que afetam a satisfação dos *stakeholders* em prol da agilidade em organizações que adotam métodos ágeis, porque o nível de sucesso na melhoria do processo de software pode ser caracterizado em termos de satisfação pessoal e se o novo processo é realmente usado.

Tanto por isso quanto para complementar as evidências empíricas do domínio, a metodologia deste trabalho é baseada em estudo de caso e na experiência e no conhecimento de envolvidos com desenvolvimento partindo dos pontos indicados pela literatura como fatores de escolha entre os tipos de metodologia de desenvolvimento de software.

1.2 Objetivos e escopo

Este trabalho busca identificar e ponderar os fatores presentes nos cenários nos quais cada tipo de metodologia possa ser mais adequado do que o outro para ser utilizado, permitindo assim atingir as metas desejadas. A escolha entre os tipos de metodologias é uma questão ainda bastante discutida e com muito espaço para mais estudos que tragam evidências empíricas.

Embora nem todos os problemas em projetos de software possam ser resolvidos por processos de metodologias de software, nos casos durante a transformação ágil, uma organização pode ser vista como um campo de força constituído de forças a favor e contra a agilidade (KETTUNEN; LAANTI, 2008). “Transformação ágil” refere-se às mudanças para a implementação de metodologias ágeis numa organização. Como descrito anteriormente, Kettunen e Laanti (2008) descrevem esse campo de força como metas (o que se tenta atingir com a agilidade, como a resolução de alguns problemas), meios (o que precisa ser feito para se tornar ágil) e habilitadores (o que precisa estar posto antes que a agilidade possa ser colocada em prática). Este trabalho parte deste escopo, abordando os habilitadores como os fatores para escolha e os meios como sendo as próprias aplicações das metodologias. O resultado servirá de apoio para observação de se a metodologia escolhida é a melhor para o cenário atual, pois sua correta aplicação é que permite atingir as metas desejadas.

Kettunen e Laanti (2008) também citam que, tomando a companhia como um todo, pode-se observar que há fatores suportando e inibindo a agilidade de projeto de software. Observa-se que os fatores “inibidores” devem ser identificados e removidos, bem como alguns fatores habilitadores são mais significantes do que outros e que ainda alguns podem ser compensados, a algum custo, se necessário. Este trabalho buscará identificar estes fatores de acordo com a literatura e também avaliar quais as suas significâncias nos cenários. Também serão observados os mesmos fatores caso a metodologia a ser adotada seja do tipo tradicional.

Os fatores aqui abordados estão classificados como pertencentes a dois grupos, de acordo com suas abrangências e descrições na literatura:

- Fatores relacionados à organização e às pessoas - porte e estrutura organizacional, formatação das equipes de desenvolvimento, autonomia para os grupos de trabalho, alinhamento a normas e boas práticas;

- Fatores relacionados aos projetos de desenvolvimento de software - duração, complexidade do projeto e do resultado gerado, quantidade de pessoas,

viabilidade de comunicação e proximidade entre os membros da equipe, viabilidade de comunicação e proximidade com o cliente ou área solicitante, risco e incerteza, criticidade.

Este estudo tem como objetivo identificar os fatores habilitadores e também inibidores relacionados aos cenários formados por características da organização, das pessoas e dos projetos de desenvolvimento de software para a escolha da aplicação de metodologias ágeis ou de metodologias tradicionais de acordo com a que apresentar maior adequabilidade.

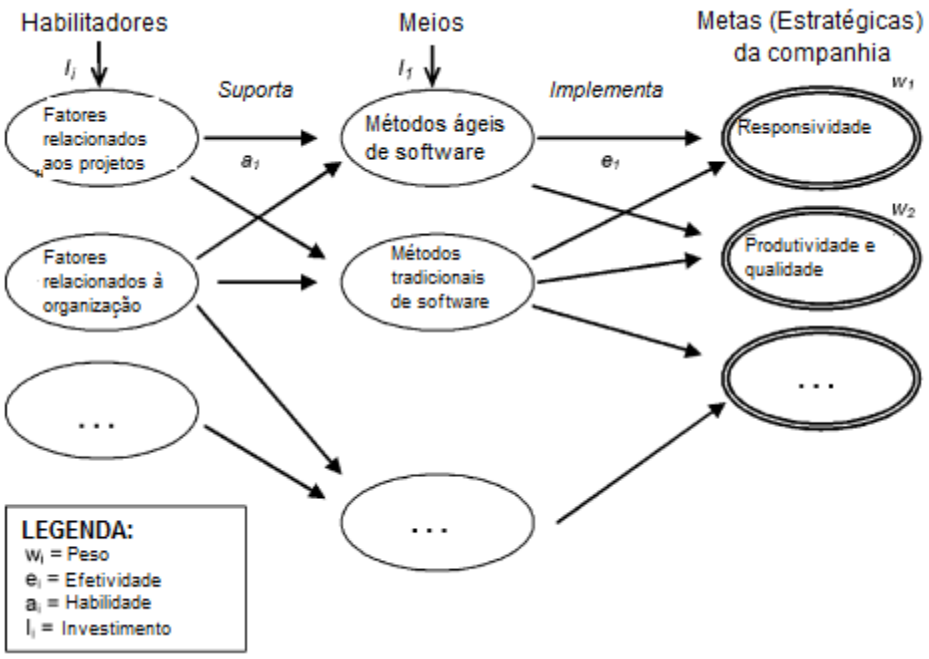
Os fatores aqui tratados foram levantados e identificados de acordo com o que é apresentado na literatura sobre evolução e histórico de uso, premissas de aplicação, objetivos da organização e das metodologias e outros estudos sobre as metodologias de desenvolvimento de software.

Abrahamsson (2000) faz uma adaptação da literatura de gerenciamento de projetos onde conclui que, em melhoria dos processos de software, são sugeridas cinco dimensões que caracterizam o nível de sucesso alcançando: eficiência do projeto, impacto no usuário do processo (sendo o usuário o objeto da mudança, como são os desenvolvedores de software neste contexto), sucesso do negócio, sucesso operacional direto e adequação da melhoria do processo e preparação para o futuro. Os resultados de seu trabalho indicam que os agentes de mudança (uma pessoa, um grupo de pessoas facilitadoras e outros responsáveis pela mudança desejada) valorizam mais, entre essas dimensões, a satisfação do usuário do processo, mas ainda assim todas foram avaliadas como sendo pelo menos moderadamente importantes para avaliação do sucesso como um todo. O nível de sucesso nesta dimensão é caracterizado em termos do nível de satisfação com o novo processo, realização das necessidades dos desenvolvedores de software, resolução dos problemas que experimentaram e se o processo melhorado é realmente utilizado. Este trabalho será baseado em estudo de caso e questionários respondidos por usuários do processo de desenvolvimento de software sobre a utilização e adequação dos tipos de metodologia aos fatores apresentados.

Abrahamsson (2000) também justifica esta sua adaptação ao dizer que a literatura de gerenciamento de projetos pode ser vista como um referencial adequado para tal, já que a maioria das atividades de melhoria de processos de software são desempenhadas em projetos com seus próprios orçamentos, escalas de tempo, recursos dedicados e metas. Isto também suporta a divisão dos fatores entre aqueles relacionados à organização e às pessoas e aqueles relacionados aos projetos.

Os questionários respondidos por desenvolvedores de uma empresa escolhida como base empírica serão utilizados para avaliar os pesos ou significâncias dos fatores e, então, utilizados para complementar o esquema gráfico como proposto por Kettunen e Laanti (2008) (figura 2).

Figura 2 – Inclusão de componentes para utilização de métodos ágeis e métodos tradicionais de produção de software.



Baseado em Kettunen e Laanti (2008)

Para a companhia toda trabalhar de forma ágil, não é suficiente focar apenas nas dimensões dos projetos e da equipe, como seria para uma aplicação típica das metodologias (KETTUNEN; LAANTI, 2008). Contudo, o escopo deste trabalho é restrito à escolha de um tipo de metodologia para a sua aplicação típica, dados os fatores que compõem os cenários nos quais algum tipo possa ser mais adequado do

que o outro e que permita alcançar as metas estabelecidas pela companhia, mas por projeto.

Abrahamsson (2000) também reforça em seu trabalho que é importante considerar outras dimensões de sucesso além da avaliação pelo usuário do processo, pois embora esta seja considerada a mais significativa para a melhoria do processo, as outras são também, pelo menos, relevantes. Este trabalho, dada a metodologia utilizada por ele, tem seu foco limitado à principal dimensão, o que representa um avanço para as evidências empíricas no domínio. Todavia, para medir a efetividade total dos processos e metodologias adotadas, o modelo ainda precisa ser mais incrementado, indo além do escopo aqui abordado em estudos futuros.

Algumas referências sobre processos compostos por metodologias mistas (mistura de metodologias tradicionais e ágeis) também são descritas, decompondo-se em suas partes ágeis e tradicionais para a discussão apresentada, porém este trabalho não se aprofunda em propor a integração entre os dois tipos de metodologia e estudar as características que envolvem tal atividade, mas sim em manter em foco a adequação individual dos dois tipos de metodologia a cada fator que compõe os cenários estudados. Ainda que citando estes casos nas discussões teóricas, deixa-se espaço para estudos similares sobre tal integração, como sugerido e realizado por Silva et al (2011).

Também recorrente no domínio é a discussão sobre a customização (*tailoring*) de processos de softwares. O termo é definido como “o ato de adaptar um processo padrão de software para atender às necessidades de uma organização ou projeto em específico” (PEDREIRA et al., 2007). Isso acontece devido à dificuldade de processos de software se adequarem às organizações na forma em que eles são apresentados.

Em sua revisão da literatura sobre customização, Martínez-Ruiz et al. (2012) indicam que, embora existam muitas iniciativas sobre o tema, não há uma abordagem única ou consenso sobre como realizar tal customização de forma controlada e consistente ou ainda que haja uma notação completa que a suporte. Em seu estudo,

levantam fatores que afetam as definições de processos, classificando-os entre fatores de projeto, da organização e de contextos sociais / legais / políticos da organização. Por fim, também citam que cada projeto e organização é diferente, portanto certas características de processos podem se diferenciar entre elas. Uma customização detalhada permite ao engenheiro de processos lidar com variações no modelo de processo, assim como configurar processos isolados para seu cenário.

Este estudo também não pretende se aprofundar sobre customização de processos, exceto pelo modelo para escolha entre o tipo tradicional ou ágil para cada cenário, seja uma metodologia customizada ou bem consolidada dentro de cada tipo. Contudo, pela divisão de fatores apresentada e pela discussão proposta entre os tipos de metodologia e suas características, pode servir de referência para estudos que se proponham a discutir mais profundamente o tema.

A metodologia deste trabalho consiste em estudo de caso e tratamento dos resultados dos questionários obtidos através dos respondentes, profissionais da área, e de seus conhecimentos e experiências ágeis e tradicionais, seguido de discussão sobre o que existe na literatura sobre o tema e confrontada com os resultados obtidos para a totalidade do trabalho e composição do modelo de escolha apresentado.

Assim, o objetivo atende à necessidade encontrada por Dybå e Dingsøyr (2008) de estudos que determinem as situações nas quais aquilo que se aconselha na literatura possa ser adequadamente aplicado. A pergunta que caracteriza o objetivo principal deste estudo e composição do modelo é: “quais fatores relacionados à organização, às pessoas e aos projetos devem influenciar na escolha de um tipo ou de outro de metodologia de desenvolvimento de software? ”.

Os resultados e metodologia aqui apresentados contribuem com evidências empíricas que ainda são necessárias e úteis para este domínio do conhecimento. Ainda contribui, ao final, com um modelo para a visualização destes fatores de acordo com os resultados obtidos, de forma a auxiliar na escolha entre um dos tipos

de metodologia, além de trazer a proposta de classificação dos fatores envolvidos nos cenários de acordo com suas abrangências (organização e pessoas ou projeto).

As próximas seções estão organizadas nos seguintes tópicos: aspectos teóricos, divididos em uma primeira parte que apresenta a literatura estudada através do histórico, características e outros estudos sobre cada tipo de metodologia e uma segunda parte que trata dos fatores recorrentes nos textos para a escolha de um tipo ou outro de metodologia; a metodologia deste trabalho; os resultados obtidos pela metodologia e discussões; e, por fim, análise e conclusões.

2 Aspectos teóricos

2.1 Apresentação das metodologias tradicionais e metodologias ágeis de desenvolvimento de software

Esta parte trata do levantamento da literatura sobre o tema para o desenvolvimento teórico do estudo. As seções a seguir estão divididas em: histórico e evolução das metodologias tradicionais de desenvolvimento de software; histórico e evolução das metodologias ágeis de desenvolvimento de software; e observações realizadas em estudos anteriores sobre ambos os tipos de metodologias. Todavia, as seções não possuem o escopo de explicar ou de se aprofundar muito em cada uma das metodologias citadas, apenas o suficiente para apresentar o histórico de evolução e as características que as unem sob um mesmo tipo ou diferenciam-nas entre si.

2.1.1 Histórico e evolução das metodologias tradicionais de desenvolvimento de software

O desenvolvimento de software é uma composição complexa de processos, tecnologias, ferramentas e pessoas, frequentemente presente em domínios com requisitos voláteis (SILVA et al., 2011). De acordo com Pressman e Maxim (2016) uma definição de software pode ser: “instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; estruturas de dados que possibilitam aos programas manipular informações

adequadamente; e informação descritiva, tanto na forma impressa quanto na virtual, descrevendo a operação e o uso dos programas".

Boehm (1988) cita que as principais funções de um modelo de processo de software são determinar a ordem ou evolução dos estágios envolvidos no desenvolvimento de software e estabelecer os critérios de transição para se progredir de um estágio para o outro. Portanto, incluem critérios de conclusão para o estágio atual, critérios condicionais de escolha e critérios de entrada para progressão ao próximo estágio. Um modelo de processo trataria então das seguintes questões envolvidas no projeto de software: "o que devemos fazer a seguir? ", "o quanto devemos continuar a fazer o que está sendo feito? ".

Boehm (1988) ainda complementa que um modelo de processo difere de um método de software, frequentemente chamado metodologia, onde este segundo foca em como navegar entre cada fase (determinar dados, ter controle ou usar hierarquias, particionar funções, alocar requisitos) e como representar as fases dos produtos (gráficos estruturais, linhas de estímulo e resposta, diagramas de transição de estado). Contudo, dadas as suas características e propósitos, tanto alguns modelos quanto metodologias podem ser categorizados entre os tipos tradicional ou ágil, embora haja diferentes ênfases e níveis de detalhamento entre eles.

Boehm (1988) também apresenta alguns modelos de processos e sua evolução. O modelo "codificar-e-corrigir" foi o modelo básico utilizado nos primórdios de desenvolvimento de software e baseava-se nas etapas:

1. Escrever algum código;
2. Corrigir os problemas no código.

Assim, a ordem dos passos era codificar algo primeiro e pensar sobre requisitos, design, teste e manutenção apenas depois, durante o desenvolvimento. Este modelo apresenta três dificuldades primárias:

A. Após certo número de correções, o código se torna tão fracamente estruturado que novas correções são muito caras. Isto destaca a necessidade de uma fase de design antes da codificação;

B. Frequentemente, mesmo softwares bem desenvolvidos apresentavam uma baixa aderência às necessidades dos usuários. Isto destacou a necessidade de uma fase de definição de requisitos antes do design;

C. O código se torna caro para ser corrigido devido à baixa preparação para testá-lo e modificá-lo. Isto trouxe o reconhecimento da necessidade da fase de testes e da possibilidade de correções ou modificações futuras, assim como a necessidade de planejamento para tal, com tarefas de preparação nas primeiras fases.

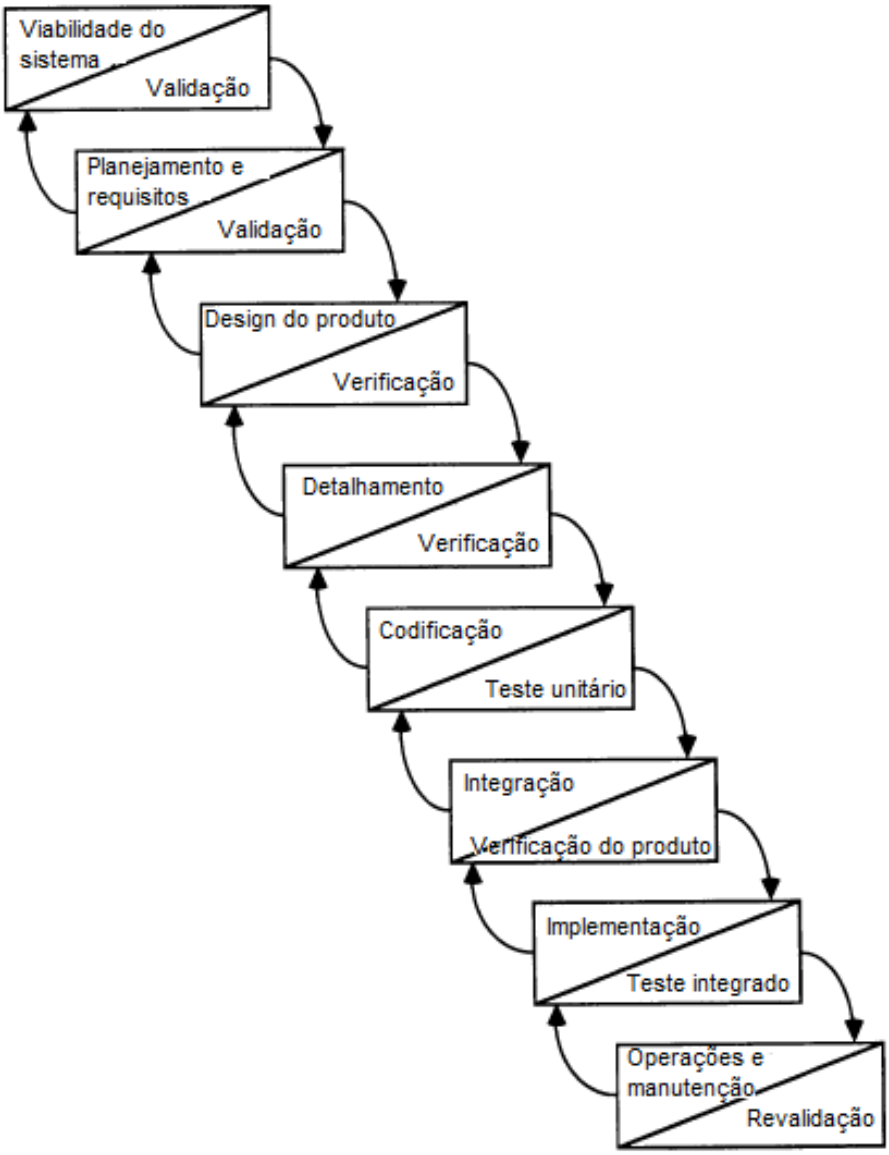
Consequentemente, Boehm (1988) apresenta os modelos *stagewise* e cascata (do inglês "*waterfall"*). Em 1956, os problemas decorrentes do modelo codificar-e-corrigir foram devidamente reconhecidos e levaram ao desenvolvimento de um modelo *stagewise* para tratá-los. Este modelo estipulava que o software deveria ser desenvolvido em estágios sucessivos: planejamento operacional, especificações operacionais, especificações do código, codificação, teste de parametrização, teste de montagem, *shakedown* e avaliação do sistema. O modelo cascata (figura 3) veio então como um refinamento deste modelo anterior e que fora muito influente na década de 70. Ele forneceu duas melhorias primárias para o modelo *stagewise*:

1. Reconhecimento de loops de feedback entre estágios, como uma linha guia para confinar os loops de retorno apenas aos estágios sucessivos entre si, o que minimizaria o caro retrabalho ao retornar muitos estágios;
2. Uma incorporação inicial de prototipação no ciclo de vida do software, em um passo de "construir duas vezes", em paralelo à análise de requisitos e design também no início.

Larman e Basili (2003) citam que alguns motivos para a adoção antecipada ou promoção continuada das ideias do modelo cascata incluem:

- Simplicidade de se explicar e de se lembrar da ordem das fases, enquanto desenvolvimento incremental (que já existia) é mais complexo de se compreender e descrever. Mesmo o modelo cascata original de Winston Royce (ROYCE, 1970), de duas iterações fora evoluído imediatamente para um modelo conhecido por fases únicas e sequenciais, de acordo com a maneira pela qual os desenvolvedores o adotaram e fora descrito por muitas vezes;

Figura 3 – Modelo cascata de ciclo de vida do software.



Fonte: Boehm (1988)

- Impressão de um processo ordenado, contável e mensurável, através da utilização de marcos orientados a documentação;
- Promoção do modelo na formulação de muitos requisitos de engenharia de software em textos de gestão, cursos e por consultorias, sendo sempre considerado apropriado ou ideal, como se não houvesse histórico e evidências conhecidas em paralelo a favor de desenvolvimento incremental.

O modelo cascata ajudou a eliminar muitas das dificuldades encontradas previamente em projetos de software, tornando-se a base para a maioria dos

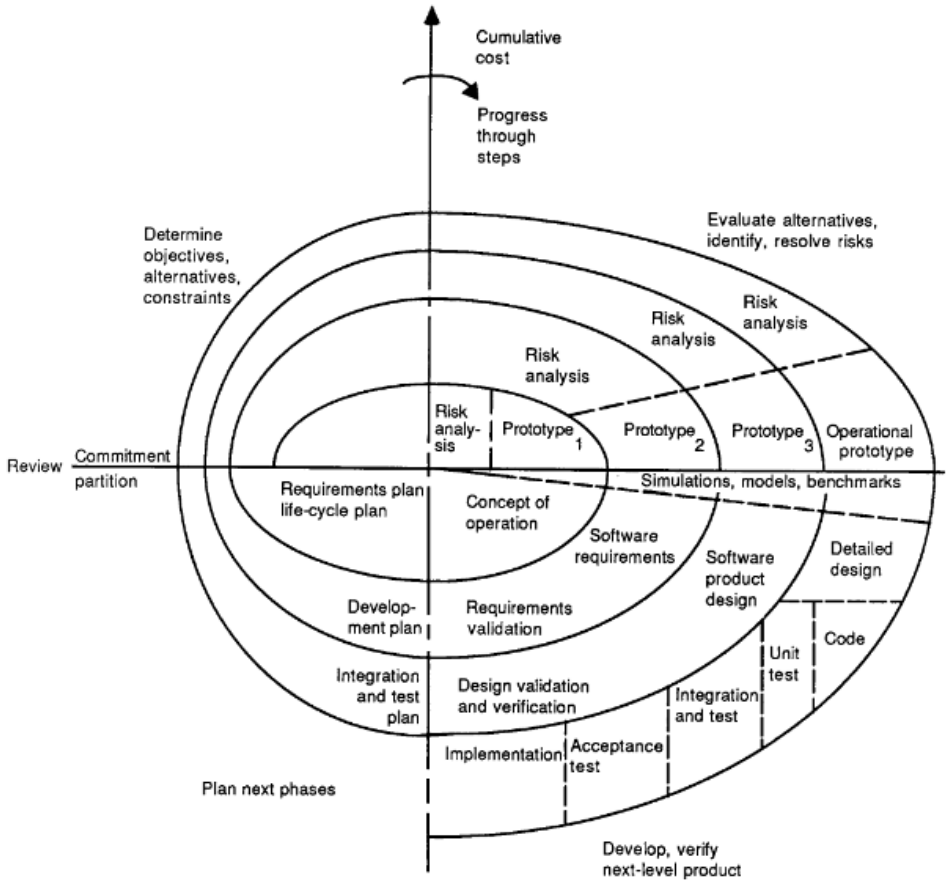
padrões de desenvolvimento e de aquisição de software no governo e na indústria (BOEHM, 1988). Algumas de suas dificuldades iniciais foram tratadas adicionando-se extensões para cobrir o desenvolvimento incremental, desenvolvimento em paralelo, famílias de software, acomodação de mudanças evolucionárias, desenvolvimento formal de software, verificação, validação e análise de riscos. Contudo, mesmo com revisões extensivas e refinamentos, o esquema básico do modelo cascata encontrou algumas dificuldades fundamentais e estas levaram à formulação de modelos de processo alternativos (BOEHM, 1988).

Boehm (1988) ainda cita que uma fonte primária de dificuldade com o modelo cascata foi sua ênfase em documentos muito elaborados como critérios de completude para fases primárias de requisitos e design. Para algumas classes de software como compiladores ou sistemas operacionais de segurança, este é o modo mais efetivo de se proceder, porém isso não funciona bem para muitas outras classes, particularmente para as aplicações interativas para o usuário final. Padrões orientados à documentação levaram muitos projetos a terem suas especificações de interfaces ao usuário muito elaboradas e funções de suporte à decisão fracamente compreendidas pelos envolvidos no início do desenvolvimento, seguidas pelo design e desenvolvimento de grandes quantidades de código não utilizado ao final. Tais projetos são exemplos de como o modelo cascata pode apresentar dificuldades devido à ordem fixa de seus estágios. Além disso, em áreas suportadas por linguagens de quarta geração (planilhas eletrônicas ou aplicações de pequenos negócios) é muitas vezes desnecessário escrever especificações muito elaboradas para uma aplicação antes de implementá-la.

Boehm (1988) apresenta então o modelo espiral (figura 4), evoluído e baseado na experiência obtida com vários refinamentos do modelo cascata. A maior diferença em relação ao modelo anterior é que este cria uma abordagem orientada a riscos para o processo de desenvolvimento de software ao invés de ser primariamente orientado a documentação ou ainda orientado a codificação. O autor complementa que este modelo incorpora muitas das características tradicionais do modelo cascata, mas enfrenta também alguns dos problemas citados através de sua proposta, reforçando a característica incremental que apresenta. O modelo espiral

formalizou e tornou proeminente o conceito de iterações orientadas a risco e também a necessidade de avaliação de risco a cada iteração (LARMAN; BASILI, 2003). Esta é também uma metodologia do grupo tradicional, por apresentar o ciclo de vida característico deste grupo e outros de seus aspectos, como será apresentado mais adiante.

Figura 4 – Modelo espiral de desenvolvimento de software.



Fonte: Boehm (1988)

O ciclo de vida de software clássico é o apresentado pelo modelo cascata: uma abordagem sequencial e sistemática para o desenvolvimento de software, passando por planejamento, modelagem, construção, disponibilização e suporte contínuo do software concluído (PRESSMAN E MAXIM, 2016). Cada iteração - comunicação, planejamento, modelagem, construção e entrega aplicados repetidamente - produz um incremento de software que disponibiliza uma parte dos recursos e das funcionalidades deste, tornando-o mais completo a cada execução (PRESSMAN; MAXIM, 2016).

Seguindo historicamente, uma das mais utilizadas e mais conhecidas abordagens para se avaliar a complexidade e sucesso de sistemas emergiu do trabalho realizado na Carnegie Mellon University no final dos anos 80, conhecida como Capability Maturity Model (CMM) (PAULK, 2001). Este modelo apresenta cinco níveis de capacidade e maturidade organizacional relacionados ao desenvolvimento de software. Ainda que resumido apenas ao essencial, o software CMM comporta 52 metas primárias e 18 áreas de processos. Embora tragam benefícios para cada uma de suas áreas, as recomendações para se tornar uma organização nível 5 consomem mais do que 500 páginas de requisitos. Embora muito utilizado e desejado, o modelo apresentou algumas limitações e foi sucedido pelo Capability Maturity Model Integration (CMMI).

Antes de continuar a linha do tempo para a apresentação das metodologias ágeis, atualmente o Product Line Practice Framework (PLPF) do Software Engineering Institute (SEI) é considerado um padrão industrial maduro e estabelecido para as práticas de SPL (CLEMENTS; NORTHROP (2001) apud SILVA et al. (2011)). Ele identifica os conceitos fundamentais do SPL e as atividades a serem consideradas na criação de uma linha de produto através de áreas práticas. Cada uma destas áreas é composta por um conjunto de tarefas que a organização deve cumprir para obter sucesso com o trabalho essencial de uma linha de produto (CLEMENTS; NORTHROP (2001) apud SILVA et al. (2011)). Este framework tem vasto uso em organizações com diferentes estruturas e áreas de negócios nos dias atuais, sendo que apresenta características das metodologias tradicionais.

Até este ponto, tem-se o histórico das metodologias tradicionais, com os problemas por elas solucionados, mas também os que apresentaram posteriormente. Nerur, Mahapatra e Mangalaraj (2005) resumem as características deste tipo de metodologia:

- O desenvolvimento de sistemas na abordagem tradicional é guiado por um modelo de ciclo de vida, como o modelo cascata, o modelo espiral ou algumas variações destes. Este modelo de ciclo de vida especifica as tarefas a serem

realizadas e os resultados desejados de cada fase, além de atribuir papéis aos indivíduos que executam essas tarefas;

- Em adição ao produto final de código funcional, estas metodologias também produzem grande quantidade de documentação, que codifica os processos e o conhecimento sobre o produto de software gerado;

- A comunicação entre os participantes do projeto também é formalizada através de documentos;

- Clientes tem um papel importante durante o desenvolvimento da especificação, mas a participação deles é mínima em outras atividades;

- Estas metodologias são apropriadas tanto para tecnologias orientadas a objetos (softwares com estruturas mais modularizadas e que seguem padrões de projeto, sendo este um paradigma mais atual de desenvolvimento) quanto às que não são (e. g., os padrões mais antigos, desenvolvidos em linguagens estruturadas e com estruturas mais monolíticas);

- Planejamento e controle realizados por um estilo de gerenciamento de comando-e-controle fornecem o ímpeto para o desenvolvimento do produto de software (HIGHSMITH, 2003);

- É uma abordagem racionalizada, baseada em engenharia, e tem dominado o desenvolvimento de software quase desde a sua concepção. Assim, apresenta planejamento extensivo no início e medição durante o desenvolvimento, mas também apresenta problemas de controle decorrentes de variações durante o ciclo de vida de desenvolvimento.

Highsmith e Cockburn (2001) ainda citam que estas metodologias são centradas em processos e guiadas pela crença de que mesmo as fontes de variações são identificáveis e podem ser eliminadas por medição contínua e refinamento de processos. O foco primário é na realização de processos altamente otimizados e

repetíveis. Baseiam-se no princípio de sistemas sólidos e seriam melhor aplicadas em ambientes onde fosse possível especificar bem os problemas, prever soluções e otimizá-las (CAVALERI; OBLOJ, 1993).

Assim sendo, este tipo de metodologia é defendido por aqueles advogam o uso de planejamento extensivo, processos codificados e rigoroso reuso para tornar o desenvolvimento uma atividade preditiva e eficiente, gradualmente amadurecendo em direção à perfeição (BOEHM, 2002). Contudo, Cockburn e Highsmith (2001) citam que, nas últimas décadas, as forças do mercado, os requisitos dos sistemas, a implementação de tecnologias e equipes dos projetos apresentaram mudanças a uma taxa crescente, então um estilo diferente de desenvolvimento emergiu para mostrar as suas vantagens sobre o tradicional: o estilo ágil, que aborda diretamente os problemas que surgem por conta de tais mudanças rápidas.

Assim, organizações que sentem essas influências precisam de sistemas de informação que evoluam constantemente para atingirem seus requisitos de mudança, mas as metodologias tradicionais de desenvolvimento de software, mais orientadas a planejamento, não possuem a flexibilidade para se ajustarem dinamicamente a este cenário de processo de desenvolvimento de software. As abordagens ágeis veem a mudança de uma perspectiva que espelha esse ambiente turbulento, como as vindas da necessidade incessável dos negócios por inovações e também da preparação de força de trabalho para o futuro (HIGHSMITH; COCKBURN, 2001).

2.1.2 Histórico e evolução das metodologias ágeis de desenvolvimento de software

Métodos de desenvolvimento ágil emergiram no final da década de 90 (LARMAN; BASILI, 2003; KETTUNEN; LAANTI, 2008). Hoda, Noble e Marshall (2011) e Wang, Conboy e Cawley (2012) citam em seus trabalhos que o termo “ágil” foi adotado para abranger métodos como Scrum (SCHWABER; BEEDLE, 2002), XP (eXtreme Programming) (BECK, 1999), Crystal (COCKBURN, 2001), FDD (Feature-Driven

Development) (PALMER; FELSING, 2001; COAD; PALMER, 2002), DSDM (Dynamic Software Development Method) (STAPLETON, 1997), Adaptive Software Development (HIGHSMITH, 2000; HIGHSMITH, 2002) e Lean (POPPENDIECK, M.; POPPENDIECK, T., 2003; CHARETTE, 2003).

As metodologias DSDM (STAPLETON, 1997), definida em 1994 por um grupo de desenvolvedores que praticavam “desenvolvimento rápido de aplicações”, e XP (BECK, 1999), originada da participação de seu autor em 1996 no projeto C3 da Chrysler, são algumas das primeiras metodologias ágeis (LARMAN; BASILI, 2003).

A adoção do termo se deu com a formação da Agile Alliance (2001), onde 17 pessoas, representantes dessas metodologias e outros simpatizantes que percebiam a necessidade de uma alternativa a processos de desenvolvimento de software orientados a documentação e “pesados”, reuniram-se para encontrar bases em comum, ainda que algumas vezes fossem competidores entre si, cada qual com sua metodologia. De sua concordância emergiu o “Manifesto Ágil” (AGILE ALLIANCE, 2001):

> “Manifesto para Desenvolvimento Ágil de Software
>
> Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
>
> Indivíduos e interações mais que processos e ferramentas;
> Software em funcionamento mais que documentação abrangente;
> Colaboração com o cliente mais que negociação de contratos;
> Responder a mudanças mais que seguir um plano.
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.”

Highsmith e Cockburn (2001) explicam em seu trabalho que quando há pressão entre esses itens, e normalmente há, algo precisa ceder. Sobre essas prioridades:

- Apoiar-se nas interações entre indivíduos facilita o compartilhamento de informações e a realização de mudanças rápidas de processo quando necessário, além de permitir minimizar a documentação total gerada;

- O uso de software em funcionamento permite medir o quão realmente rápido a equipe produz resultados e também permite rápido feedback;

- Colaboração com o cliente significa fazer com que todos os envolvidos (patrocinador, cliente, usuário e desenvolvedor) estejam no mesmo time, onde a combinação de suas diferentes experiências e conhecimentos, utilizados com boa-fé, permitam ao grupo mudar os rumos rapidamente, podendo-se produzir resultados mais apropriados e designs menos caros (Contudo, mesmo havendo contratos, sem colaboração, estes são insuficientes para alcançar tais resultados);

- Planejamento ajuda os membros do time a pensar em todo o projeto e em suas contingências, porém o plano normalmente fica desatualizado em poucos dias, o que faz ser mais importante lidar com as realidades que mudam do que focar em seguir estritamente estes planos.

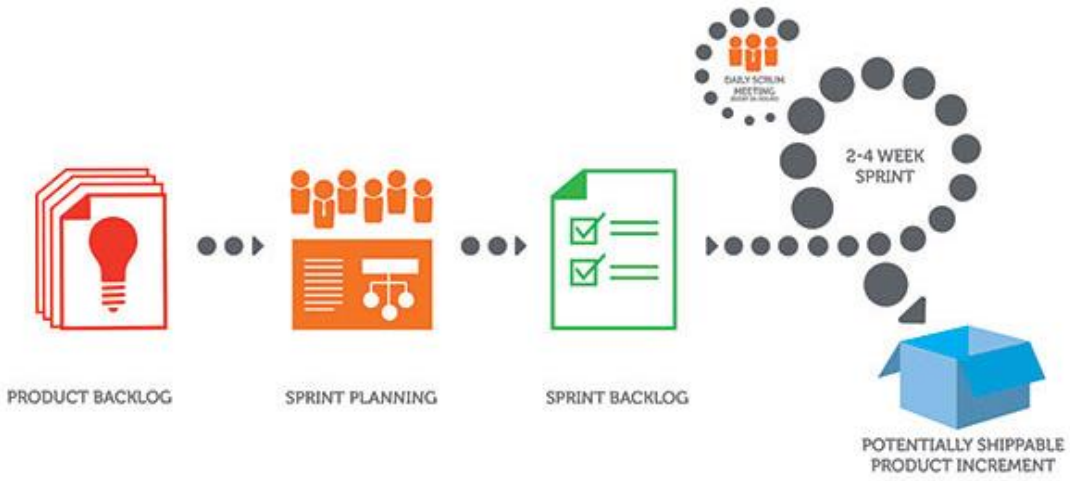
Kettunen e Laanti (2008) citam que não há definição universal de agilidade, porém, todas as definições geralmente incorporam os conceitos básicos de velocidade e flexibilidade para resposta a mudanças nos ambientes dinâmicos de mercado. Também lembra que, em desenvolvimento de software, o termo “agilidade” foi adotado depois e independentemente do termo agilidade em manufatura ágil.

Embora todas as metodologias ágeis compartilhem de certos princípios em comum, não há padrão de definição dos processos de desenvolvimento ágil de software entre elas, havendo espaço também para compará-las entre si (KETTUNEN; LAANTI, 2008). Nerur, Mahapatra e Mangalaraj (2005) posicionam-se da mesma maneira. Todavia, Silva et al (2011) citam que, em geral, essas metodologias aplicam desenvolvimento iterativo “*time-boxed*” (em intervalos regulares) e evolutivo, planejamento adaptativo, promovem entrega evolutiva e incluem outros valores e

práticas que encorajam o feedback e permitem resposta flexível a mudanças. Wang, Conboy e Cawley (2012) também citam essas características em comum, reforçando a velocidade e frequência com que os ciclos acontecem. Apesar dos diferentes processos propostos, algumas práticas são também compartilhadas entre diferentes metodologias ágeis.

Dybå e Dingsøyr (2008), Laanti, Salo e Abrahamsson (2011) e Silva et al (2011) lembram que diferentes métodos ágeis buscam atingir diferentes aspectos de desenvolvimento de software, como XP em implementação de software, TDD em testes unitários e Scrum em gerenciamento do planejamento do projeto e implantação. Scrum e XP são os métodos ágeis mais largamente adotados no mundo, seguidos por TDD (DYBA; DINGSØYR, 2008). Um híbrido entre XP e Scrum também tem sido uma das “metodologias” mais adotadas (VERSIONONE, 2010). A tabela 1 mostra uma breve descrição de algumas das principais metodologias ágeis. A figura 5 mostra as iterações do Scrum como exemplo.

Figura 5 – Scrum – metodologia ágil iterativa de desenvolvimento de software.



Fonte: Scrum Alliance (2015)

Dadas as suas características, as metodologias ágeis podem ser caracterizadas sob um mesmo tipo, assim também as diferenciando das metodologias tradicionais. Cockburn (2001), por exemplo, cita que métodos ágeis focam em código funcional enquanto reduzem o design no início e a sobrecarga de processos das abordagens

Tabela 1 - Descrição breve de algumas metodologias ágeis.

Método ágil	Descrição
Metodologias Crystal	Uma família de métodos para times coalocados de diferentes tamanhos e criticidades de projeto: Clear, Yellow, Orange, Red e Blue. O método mais ágil, Crystal Clear, foca em comunicação em times pequenos desenvolvendo software que não é muito crítico. Desenvolvimento Clear apresenta sete características: entrega frequente, melhoria reflexiva, comunicação próxima, segurança pessoal, foco, fácil acesso a usuários experts e definição de requisitos para o ambiente técnico
Dynamic software development method (DSDM)	Divide projetos em três fases: pré-projeto, ciclo-de-vida do projeto e pós-projeto. Nove princípios sustentam DSDM: envolvimento do usuário, autonomia do time do projeto, entrega frequente, abordagem das necessidades atuais do negócio, desenvolvimento iterativo e incremental, permissão para mudanças reversivas, escopo em alto-nível fixado antes do projeto iniciar, testes durante o ciclo-de-vida e comunicação eficiente e efetiva
Feature-driven development (FDD)	Combina modelagem e desenvolvimento ágil com ênfase no objeto inicial do modelo, divisão do trabalho em funcionalidades e design iterativo para cada funcionalidade. Afirma ser adequado para o desenvolvimento de sistemas críticos. Uma iteração para uma funcionalidade consiste de duas fases: design e desenvolvimento.
Desenvolvimento lean de software	Uma adaptação dos princípios da produção lean, em particular o *Toyota Production System* (TPS), ao desenvolvimento de software. Consiste em sete princípios: eliminar o desperdício, amplificar o aprendizado, decidir o quão tarde for possível, entregar o quão rápido for possível, dar autonomia ao time, construir com integridade e observar o todo.
Scrum	Foca em gestão de projetos em situações onde é difícil de se planejar à frente, com mecanismos para "controle empírico de processos", onde os loops de feedback constituem o elemento central. Software é desenvolvido por times autogeridos em incrementos (chamados "*sprints*"), começando com o planejamento e terminando com uma revisão. Funcionalidades a serem implementadas no sistema são registradas em um backlog. Então o dono do produto decide quais itens do backlog devem ser desenvolvidos no sprint seguinte. Membros do time coordenam seus trabalhos em uma reunião em pé ao final do dia. Um membro do time, o *scrum master*, tem a função de resolver problemas que impedem o time de trabalhar efetivamente.
Extreme Programming (XP, XP2)	Foca nas melhores práticas para o desenvolvimento. Consiste em doze práticas: o jogo do planejamento, pequenos lançamentos, metáforas, design simples, teste, refatoração, programação pareada, propriedade coletiva, integração contínua, 40h por semana, clientes *in loco* e padrões de codificação. O "XP2" revisado consiste nas seguintes "práticas primárias", além de outras "práticas corolárias": sentar juntos, visão de time como um todo, espaço de trabalho informativo, trabalho energizado, programação pareada, uso de estórias, ciclo semanal, ciclo trimestral, intervalos, construção em 10 minutos, integração contínua, programação do teste primeiro e design incremental.

Baseado em Dybå e Dingsøyr (2008)

tradicionais. Isso é uma característica das metodologias ágeis e é decorrente da adoção de valores fundamentais e princípios existem por trás do Manifesto Ágil (AGILE ALLIANCE, 2001):

> “Princípios por trás do Manifesto Ágil
>
> Nós seguimos estes princípios:
> 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.”

Nerur, Mahapatra e Mangalaraj (2005) resumem sobre metodologias ágeis:

- São caracterizadas por ciclos de desenvolvimento curtos e iterativos orientados pelos recursos alocados, períodos de reflexão e introspecção, tomada de decisão colaborativa, incorporação de rápidos feedbacks e mudanças no projeto e da integração contínua destas ao sistema em desenvolvimento (HIGHSMITH; COCKBURN, 2001; HIGHSMITH, 2002; HIGHSMITH, 2003);

- Um projeto é dividido em subprojetos, cada qual envolvendo tipicamente planejamento, desenvolvimento, integração, teste e entrega;

- Lidam com imprevisibilidade ao confiar em pessoas e em sua criatividade ao invés de processos, pois foca nos talentos e habilidades dos indivíduos, assim moldando os processos de acordo com as pessoas específicas e os times formados (COCKBURN; HIGHSMITH, 2001);

- Desenvolvedores trabalham em pequenos times, juntos aos clientes (ou representantes de usuários dos sistemas), com estes sendo membros ativos do time. Os recursos a serem implementados em cada ciclo de desenvolvimento são decididos em conjunto pelo cliente e o restante do time de desenvolvimento. Tomada de decisão colaborativa envolvendo *stakeholders* com diversos backgrounds e metas é uma característica de desenvolvimento ágil;

- Metodologias ágeis favorecem um estilo de gestão de liderança-e-colaboração onde o papel de gestor de projeto é de um facilitador ou coordenador (HIGHSMITH, 2003);

- A entrega de cada ciclo de desenvolvimento é código funcional que pode ser usado pelo cliente;

- Métodos ágeis não focam em documentação além do código. O conhecimento sobre o produto se torna mais tácito. Existe rotação dos membros do time para garantir que o conhecimento não seja monopolizado por alguns poucos indivíduos;

- Além disso, as estratégias iterativas que caracterizam metodologias ágeis são melhores suportadas por tecnologias orientadas a objetos.

Assim sendo, os autores também concluem que desenvolvimento ágil é caracterizado por investigação social, na qual colaboração extensiva e comunicação providenciam as bases para uma ação coletiva (COCKBURN; HIGHSMITH, 2001;

COCKBURN, 2002; HIGHSMITH, 2003). Diversos envolvidos, incluindo desenvolvedores e usuários finais, passam por ciclos de pensamento-ação-reflexão repetidamente, o que fomenta um ambiente de aprendizagem e adaptação. Membros do time, com maior discricionária e poderes de tomada de decisão, não são confinados a um papel especializado. Isto aumenta a diversidade ou variedade dos times e permite eles a se autogerirem e responderem com espontaneidade a situações emergentes.

Sobre XP, Hilkka, Matti e Tuure (2005) citam que se trata de um modo de trabalhar e que não trouxe algo completamente novo para a discussão de desenvolvimento incremental de software, mas que suas ideias foram mais fáceis de se aceitar ao virem da comunidade de desenvolvedores e também devido ao novo nome. Os autores concluíram que XP é "vinho velho em novas garrafas" e que "formaliza vários hábitos que aparecem naturalmente (...) como envolvimento mais próximo com o cliente, ciclos mais curtos de lançamentos, desenvolvimento cíclico e respostas rápidas a solicitações de mudanças". Todavia, a maioria dos estudos trata o desenvolvimento ágil como algo "novo" e que, consequentemente, ainda requer introdução e maior adoção (DYBA; DINGSØYR, 2008). Highsmith e Cockburn (2001) complementam que o que é novo sobre os métodos ágeis não são as práticas que utilizam, mas o reconhecimento de que pessoas são os orientadores primários para o sucesso do projeto, junto a um intenso foco em efetividade e flexibilidade. Isto leva a uma nova combinação de valores e princípios que definem uma visão de mundo "ágil".

Por fim, Wang, Conboy e Cawley (2012) afirmam que, em anos mais recentes, houve uma notável mudança de atenção daqueles que usam desenvolvimento ágil de software para "desenvolvimento *lean* de software", frequentemente nomeando esse fenômeno como mudança "do ágil para o *lean*". Todavia, não é tão simples quanto parece: a análise dos autores identificou seis tipos de aplicação *lean* utilizadas em processos ágeis de diferentes maneiras e para diferentes propósitos. Conceitos, princípios e práticas *lean* são mais utilizadas para melhoria contínua no processo ágil e introduzem a abordagem de continuidade temporal, ou seja, "*flow-based*", em substituição aos processos ágeis "*time-boxeds".*

Wang, Conboy e Cawley (2012) ainda complementam que, originalmente, desenvolvimento *lean* de software foi visto apenas como outro método ágil (DYBÅ; DINGSØYR, 2009), mas que há um foco crescente em vê-lo como um novo tipo de metodologia. Outros ainda citam que o desenvolvimento *lean* de software fornece a teoria por trás das práticas ágeis (POPPENDIECK, M.; POPPENDIECK, T., 2003; POPPENDIECK, M.; POPPENDIECK, T., 2006). Essa discussão será apresentada novamente na comparação entre os diferentes tipos de metodologias, mas dado o momento em que este estudo é realizado e que um novo tipo ainda não foi consolidado, a metodologia *lean* estará incluída no tipo ágil, assim como o modelo espiral, mais orientado a riscos ao invés de documentação, está entre os tradicionais, dadas as compatibilidades da maioria de suas características.

Estudos no domínio foram realizados buscando investigar os benefícios, dificuldades e fatores relacionados à implantação de metodologias ágeis de desenvolvimentos de software, assim como estudos comparativos ou que as relacionam de alguma forma às metodologias tradicionais de desenvolvimento de software. Alguns desses estudos serão discutidos a seguir para descrever o cenário que envolve o problema abordado.

Benefícios observados pelo uso de metodologias ágeis

Assim como o surgimento e a evolução das metodologias tradicionais trouxeram benefícios, as metodologias ágeis também apresentaram benefícios nos resultados estudados após suas aplicações, sendo que ambos os tipos são utilizados tentando atingir seus respectivos objetivos em relação a área de desenvolvimento de software ou também os objetivos da organização como um todo.

Baseada nos valores destacados pelo Manifesto Ágil, como de colaboração e comunicação, responsabilidade e foco no implementação de software funcional, uma organização pode estar interessada, por exemplo, em: aumentar o retorno sobre o investimento em seus esforços para a implantação de métodos ágeis, aumentar a habilidade de entregar software mais brevemente e continuamente, melhor resposta

a mudanças nos escopos dos produtos e solicitações do cliente, aumentar a motivação e a autonomia dos times de desenvolvimento, manter o ritmo de trabalho estável e sustentável, continuamente aumentar a produtividade no trabalho ou melhorar a qualidade dos produtos finais (LAANTI; SALO; ABRAHAMSSON, 2011).

Laanti, Salo e Abrahamsson (2011) encontram que as pessoas envolvidas com desenvolvimento de software, uma vez que tenham experiência com métodos ágeis, percebem que o uso destes adiciona valor agregado ao trabalho e que quanto maior a experiência prática com métodos ágeis, mais as opiniões sobre sua utilidade são positivamente afetadas. Os autores também concluem que, para estas pessoas, as seguintes características apresentam efeitos positivos: “aumenta a efetividade do desenvolvimento”, “aumenta a qualidade do produto”, “torna o trabalho mais divertido”, “torna o trabalho mais organizado/planejado”, “aumenta a autonomia dos times de desenvolvimento”, “permite encontrar bugs/erros/defeitos mais cedo”. Além desses, os efeitos do desenvolvimento ágil que mais foram percebidos como positivos entre as afirmações pesquisadas foram “aumento de transparência” e “aumento de colaboração”. Na mesma pesquisa, a afirmação de que “ágil torna o trabalho mais agitado” ficou abaixo de neutro na média da avaliação, assim não a confirmando.

Este estudo suporta o que foi explicitado por Dybå e Dingsøyr (2008) sobre os benefícios de colaboração com o cliente e qualidade aumentada e pelo estudo de Petersen e Wohlin (2009) sobre melhor transferência de conhecimento devido a melhor comunicação e ao frequente feedback, bem como feedback em geral acontecendo mais cedo através das discussões com o cliente, como os principais benefícios dos métodos ágeis. Abrahamsson, Salo e Laanti (2011) incluem estes benefícios sob “visibilidade e transparência” como um todo. Outros benefícios mais percebidos nos resultados de seu trabalho estão em “gestão de requisitos/planejamento iterativo” e depois em “produtividade / foco / eficiência / previsibilidade” e “entrega frequente / velocidade / responsividade a mudanças”.

Mannaro, Melis e Marchesi (2004) ainda citam que 95% dos trabalhadores que utilizam XP gostariam de continuar utilizando este processo de desenvolvimento,

porém entre os que não usam desenvolvimento ágil, o índice é de 40% com seus respectivos processos tradicionais. Os que usam XP apontaram ter maior satisfação com o trabalho, sentiram o ambiente de trabalho mais confortável e acreditam que sua produtividade tem sido maior.

Dybå e Dingsøyr (2008) encontram benefícios reportados nas seguintes áreas: colaboração com o cliente, processos para lidar com defeitos, aprendizagem com programação pareada, estimativas, pensamento à frente apenas para os gestores e foco no trabalho atual para os engenheiros. Petersen e Wohlin (2009) organizam as vantagens percebidas pelo estudo de Dybå and Dingsøyr (2008):

- Melhor transferência de conhecimento devido à melhor comunicação e feedbacks frequentes a cada iteração;

- Clientes são percebidos pelos programadores como muito valiosos, permitindo aos desenvolvedores terem discussões e conseguirem antecipar os feedbacks;

- Programação pareada facilita o aprendizado, se os parceiros são trocados regularmente;

- Controle dos processos, transparência e qualidade são aumentadas através de integração contínua e tarefas pequenas e gerenciáveis;

- XP, por exemplo, é muito orientado aos aspectos técnicos, aumentando os poderes dos engenheiros e também a sua motivação;

- Pequenos times e encontros presenciais melhoram a cooperação e ajudam a conseguir melhores inspirações no processo de desenvolvimento;

- O ambiente social do trabalho é percebido como pacífico, confiável, responsável e mantenedor da qualidade de vida no trabalho;

- Clientes apreciam a participação ativa nos projetos, pois isso os permite controlar o projeto e o processo de desenvolvimento, além de se manterem atualizados;

- Desenvolvedores percebem o ambiente de trabalho como confortável e sentem-se trabalhando mais produtivamente ao utilizarem programação pareada;

- Programadores estudantes percebem também a qualidade do código aumentar ao utilizar programação pareada.

Petersen e Wohlin (2009) ainda encontram em seu estudo de caso em metodologias ágeis que pacotes de desenvolvimento foram lançados mais rapidamente, reduzindo a volatilidade dos requisitos nos projetos, e que o desperdício em trabalho não utilizado foi reduzido.

Wheelwright e Clark (1992) citam que os principais objetivos da agilidade são: responsividade a mudanças (agilidade em seu sentido literal ou flexibilidade), alcançar alta produtividade no desenvolvimento e criar produtos com distinção e integridade.

Problemas e dificuldades observadas no uso de metodologias ágeis

Laanti, Salo e Abrahamsson (2011) buscam em seu trabalho identificar as principais áreas beneficiadas pelo uso de métodos ágeis, mas também os principais desafios de sua utilização, dividindo e comparando os resultados entre grupos da população mais receptivos (grupo positivo) e os mais resistentes (grupo negativo) a tais metodologias. Três áreas são especialmente identificadas como desafiadoras: “implantação dos métodos ágeis (por si só)”, “gestão de requisitos e planejamento de modo flexível e iterativo” e “visibilidade e transparência”. Implantação é a área mais desafiadora e as outras duas, apesar de aparecerem também entre as áreas mais beneficiadas pelas metodologias, estão também entre as mais desafiadoras. Assim, poderia ser afirmado que essas três áreas em particular deveriam ser cuidadosamente contempladas por qualquer organização planejando ou já em transição para a utilização de métodos ágeis.

Petersen e Wohlin (2009) também mencionam a falta de foco no design da arquitetura entre os maiores desafios do desenvolvimento ágil e incremental e também estes não terem boa escalabilidade para toda a organização, assim como escalabilidade para a comunicação entre diferentes times. Laanti, Salo e Abrahamsson (2011) perceberam isso vindo do grupo positivo e categorizaram estas questões sob “implantação dos métodos ágeis”. Comunicação entre os times foi percebida como o terceiro maior problema apontado pelo grupo positivo e foi incluído num contexto maior, que inclui desafios organizacionais, distribuição do trabalho, subcontratação e cooperação em geral, em sua classificação.

Ainda nos resultados de Laanti, Salo e Abrahamsson (2011), o grupo negativo vê como problemas as ideias imediatistas e a falta de planejamento e de design, já o grupo positivo vê ainda como desafiador ter o desenvolvimento planejado corretamente (priorizar, incorporar mudanças), sob a categoria “gerenciamento e planejamento de requisitos”. Contudo, este mesmo ponto foi considerado beneficiado pelas metodologias ágeis por ambos os grupos, onde o grupo negativo observa maior flexibilidade em adotar novos requisitos a qualquer tempo e o positivo vê melhora na previsibilidade (estimativa torna-se mais realista). Isso significa que mesmo os pontos desafiadores podem apresentar alguns benefícios se dominados ou superados.

Silva et al (2011) também citam que há muitos desafios quando da integração entre métodos ágeis e o modelo de SPL, devido às diferenças nas filosofias de ambas as abordagens, tais como as questões sobre design e as estratégias propostas para gestão de mudanças (CARBON et al., 2006; TIAN; COOPER, 2006 apud SILVA et al., 2011). Além disso, métodos ágeis não se propõem a desenvolver artefatos para reuso (CARBON et al., 2006, NAVARRETE; BOTELLA; FRANCH, 2006 apud SILVA et al., 2011) ou desenvolver e manter documentação extensiva, como requisitado por SPL (LINDEN; SCHMID; ROMMES, 2007 apud SILVA et al., 2011). Silva et al (2011) reconhecem que é necessária uma avaliação mais profunda dos riscos e desafios associados à integração de práticas ágeis e de SPL. Este estudo identificará os fatores que fazem com que um tipo ou outro de metodologia seja

avaliado como mais adequado aos diferentes cenários que possam ser formados por estes, o que não ataca diretamente os problemas relacionado à integração entre as metodologias, mas os resultados podem servir para futuros estudos que apresentem isto em seu escopo.

Abrahamsson (2000) também cita um problema sobre companhias que tiveram más experiências no passado em seu processo de melhoria de projetos de software e que são relutantes em iniciar tais atividades novamente, onde as razões da falha são múltiplas e muitas vezes não são analisadas tão profundamente para se descobrir o que houve. Isto impede tais companhias de se prepararem para o futuro, embora continue havendo oportunidades para tal. Apesar de todas as considerações, Dybå e Dingsøyr (2008) ainda assim apontam que a maioria dos estudos reportam que as práticas de desenvolvimento ágil são fáceis de serem adotadas e que funcionam bem.

Dados esses resultados, muitos estudos abordam as questões envolvendo a implantação de metodologias ágeis - o maior desafio identificado pelos envolvidos para o uso delas. Este estudo utilizará também destas referências para identificação dos fatores que compõem os cenários para a escolha do tipo de metodologia mais adequada, uma vez que são fatores também relacionados às características das organizações, pessoas ou projetos. Assim, os resultados deste trabalho poderão também apresentar informações úteis àqueles que desejarem implantar ou mudar o uso de alguma metodologia de desenvolvimento.

Principais conclusões realizadas em estudos anteriores na implantação de metodologias ágeis

Muitos estudos descrevem processos para implantar e as consequências da implantação de metodologias ágeis em equipes de desenvolvimento de software. Aqui, estes estudos são utilizados para identificação das características neles apresentadas e que possam ser úteis para tratar a questão desta pesquisa. O escopo deste trabalho não tem como motivação o estudo da implantação do uso de metodologias ágeis de software, mas sim a proposição de um modelo que possa

levar à escolha da adoção destas ou das tradicionais em cenários cujos fatores da organização, das pessoas e dos projetos permitam melhor adequação a um tipo ou de outro de metodologia.

A adoção de métodos ágeis pode exigir modificações radicais para se encaixar no contexto atual das organizações (GRENNING, 2001 apud LAANTI, SALO E ABRAHAMSSON, 2011). Também é comum a adoção de práticas ágeis individuais (uso de algum artefato, técnica ou processo proposto por alguma metodologia ágil) ou certos fundamentos de desenvolvimento ágil de software para complementar processos já existentes em organizações (MANHART; SCHNEIDER, 2004 apud LAANTI; SALO; ABRAHAMSSON, 2011).

Nerur, Mahapatra e Mangalaraj (2005) citam que as organizações não podem ignorar a onda ágil, mas que para organizações mais arraigadas às metodologias tradicionais de desenvolvimento de sistemas, a adoção de metodologias ágeis pode apresentar muitos desafios, pois as duas metodologias de desenvolvimento estão baseadas em conceitos opostos. Uma das maiores barreiras para a migração é a mudança no modelo do processo de ciclo de vida de software, com fases bem definidas, para outro que suporta desenvolvimento baseado em funcionalidades, evolucionário e iterativo, pois tal mudança implica em maiores alterações nos procedimentos de trabalho, ferramentas e técnicas, meios de comunicação, estratégias de resolução de problemas e os papéis das pessoas.

A transformação ágil é descrita, algumas vezes, como uma mudança de cultura no desenvolvimento e implica que a cultura organizacional também seja um motivo para as dificuldades encontradas nesta transformação (DINGSØYR; DYBÅ; MOE, 2010). Sobre isso, Highsmith e Cockburn (2001) citam que também, durante décadas, as organizações buscaram incansavelmente atingir a meta de criar processos otimizáveis e repetitíveis. Nerur, Mahapatra e Mangalaraj (2005) complementam que essa desejada estabilidade representa um dos maiores obstáculos para a adoção das metodologias de desenvolvimento ágil.

Segundo Nerur, Mahapatra e Mangalaraj (2005), o problema em mudar as atitudes e práticas de centradas em processos para centradas em pessoas é mais suscetível, por exemplo, em organizações que tentaram por anos alcançar os níveis mais altos de CMM. A ideia de mudar um processo para se adequar às capacidades e competências das pessoas e das características dos projetos ao invés de utilizar um processo rígido que segue atividades padronizadas pode até parecer interessante nesses casos (COCKBURN; HIGHSMITH, 2001), mas pode ser alcançada somente através de investimentos significativos de tempo, esforço e capital.

Nerur, Mahapatra e Mangalaraj (2005) ainda reforçam que as ferramentas adotadas têm um papel crítico no sucesso da implementação de uma metodologia de desenvolvimento de software, mas que estas, por si só, não são o suficiente para tal, pois as pessoas também precisam ser treinadas corretamente para usá-las. É nesse sentido que o paradigma de linguagem de programação tem alguma influência, pois linguagens orientadas a objetos ou estruturadas possuem ferramentas adequadas a elas e a algumas metodologias específicas. Existem ferramentas que suportam parte dos processos de metodologias, porém há alguns exemplos de ferramentas que trazem consigo algum tipo de metodologia ou princípios arraigados, como Rational Unified Process (RUP) e Microsoft Solution Framework (MSF), que são amplamente utilizadas. Este trabalho presume que a organização tenha as ferramentas adequadas e as pessoas tenham sido treinadas para permitir a escolha entre metodologias de desenvolvimento tradicionais ou ágeis baseando-se então nos fatores relacionados à organização e às pessoas e aos fatores relacionados aos projetos para escolherem um tipo ou outro de abordagem, uma vez que problemas com treinamentos e ferramentas tenham sido resolvidos.

Hoda, Noble e Marshall (2011) apresentam em seu trabalho o conceito de "*Agile Undercover*", que consiste em utilizar técnicas para contornar problemas que impeçam a correta aplicação de metodologias ágeis e que são normalmente utilizadas por equipes de desenvolvimento como medida temporária durante a transição para o ágil. Esta é uma medida bastante relacionada aos fatores que serão identificados nesse trabalho e que podem tornar a utilização de um tipo ou de outro

de metodologia mais adequado ao cenário em que o desenvolvimento está inserido, dado o que pode ser feito em tais situações adversas.

Nos resultados de Laanti, Salo e Abrahamsson (2011), o grupo positivo em relação a metodologias ágeis apontou como desafio o *roll-out* simultâneo destas para todos os times de desenvolvimento (utilizando corretamente a metodologia e a implantação em larga escala). Já o grupo negativo preocupa-se, nesse aspecto, em "tornar-se ágil demais" (perder controle da previsibilidade e do andamento do desenvolvimento dos times).

Em seu trabalho, Nerur, Mahapatra e Mangalaraj (2005) identificam os fatores-chave na adoção de metodologias ágeis, dividindo-os em quatro categorias:

- Gerenciamento e organizacional
    - Cultura organizacional;
    - Estilo de gestão;
    - Formato organizacional;
    - Gerenciamento do conhecimento em desenvolvimento de software;
    - Sistemas de recompensas.
- Pessoas
    - Trabalho em equipe efetivo;
    - Alto nível de competência;
    - Relações com os clientes – comprometimento, conhecimento, proximidade, confiança e respeito.
- Processos
    - Mudanças de centrado em processos para orientado a funcionalidades e centrado em pessoas;
    - Desenvolvimento mais curto, iterativo, orientado a testes, enfatizando adaptabilidade;
    - Gerenciamento de projetos grandes e escaláveis;
    - Seleção de método ágil apropriado.
- Tecnologia (ferramentas e técnicas)
    - Adequação das ferramentas e tecnologias existentes;

- Novos conjuntos de habilidades – refatoração, gestão de configuração.

Este trabalho busca identificar os fatores que podem ser habilitadores (adequados) para metodologias ágeis e também os que podem ser habilitadores para metodologias tradicionais. Alguns dos fatores-chave identificados por Nerur, Mahapatra e Mangalaraj (2005) serão tratados nas próximas seções, porém com adição de outros fatores obtidos na literatura. Outra classificação será utilizada ao dividí-los entre um grupo de fatores relacionados à organização e às pessoas e outro grupo de fatores relacionados aos projetos, de acordo com os escopos em que eles se apresentam.

Antes de apresentar os fatores e os resultados obtidos após a aplicação da metodologia deste trabalho, também é importante apresentar os estudos que envolvem a discussão sobre comparações, conflitos e o uso de ambos os tipos de metodologia. O modelo proposto ao final tem a função de direcionar para o uso de um ou de outro tipo, dependendo de cada cenário de desenvolvimento, baseado no estudo de caso e nos questionários realizados neste trabalho, mas também se baseia nos resultados destes estudos para desenvolver a discussão.

2.1.3 Principais conclusões obtidas em estudos anteriores sobre a comparação de ambos os tipos de metodologias

Nerur, Mahapatra e Mangalaraj (2005) afirmam que metodologias de desenvolvimento de software estão evoluindo constantemente devido a mudanças na tecnologia e novas demandas dos usuários. Os autores também citam que, embora metodologias tradicionais continuem a dominar a área de desenvolvimento de sistemas, inúmeras opiniões e questionários mostram a crescente popularidade das metodologias ágeis.

Nos resultados do trabalho de Laanti, Salo e Abrahamsson (2011) a atitude em prol de métodos ágeis pode ser considerada como positiva, já que 60% dos respondentes de seu estudo preferem continuar no modo de trabalho ágil (grupo positivo), enquanto apenas 9% preferem os métodos tradicionais de trabalho (grupo

negativo). Todavia, também há 31% de respondentes que não veem nenhuma diferença ou não têm uma opinião clara nesta análise (grupo neutro). 21% do total dos respondentes não têm experiência com métodos ágeis, sendo que 12% de intersecção com estes estão no grupo neutro. Outra observação obtida neste estudo é a de que as médias nas opiniões tendem a se tornar mais positivas conforme a experiência ágil dos respondentes cresce e as médias nas opiniões tendem a se tornar mais negativas conforme a experiência não-ágil (tradicional) cresce.

Boehm (2002) diz que defensores das metodologias tradicionais advogam o uso de planejamento extensivo, processos codificados e rigoroso reuso para tornar o desenvolvimento uma atividade preditiva e eficiente, que gradualmente amadurece em direção à perfeição. O autor também diz que não é de se surpreender que muitos desenvolvedores que são a favor de métodos orientados a planejamento reagiram ao manifesto ágil com severo criticismo, mas afirma que, se comparado ao sem planejamento e indisciplinado "*hacking*" (desenvolvimento sem metodologia alguma), os métodos ágeis enfatizam uma quantidade bastante razoável de planejamento. As metodologias ágeis colocam mais valor nos processos de planejamento do que na documentação resultante (Highsmith; Cockburn, 2001), o que as faz parecer menos orientadas a planejamento do que realmente são. Em geral, métodos ágeis têm critérios para serem assim definidos e aplicados, não deixando *"hackers*" enganarem que os estão praticando quando não estiverem.

Boehm (2002) cita que uma nova geração de desenvolvedores afirma que o peso esmagador da burocracia corporativa, o rápido ritmo de mudanças na tecnologia da informação e os efeitos desumanizadores de desenvolvimento orientado a planejamento são motivos para uma revolução ágil. O autor também cita que metodologias ágeis atraíram alguns desenvolvedores que não haviam aderido anteriormente a outras metodologias.

Nerur, Mahapatra e Mangalaraj (2005) citam que o advento dessas metodologias dividiu a comunidade de desenvolvimento de software em campos opostos de tradicionalistas e agilistas, com cada grupo proclamando a superioridade de sua própria metodologia. Uma visão mais balanceada das duas metodologias

competidoras é oferecida por poucos que sugerem que cada método tem suas forças e limitações e são apropriados para tipos específicos de projetos (BOEHM, 2002; BOEHM E TURNER, 2004; COCKBURN 2002). Este trabalho apresenta este tipo de visão.

Com a crescente demanda por maior qualidade de software, custos reduzidos de desenvolvimento e menor *time-to-market*, a ideia de alavancar os pontos fortes das abordagens existentes tem aumentado como uma possível solução para se atingirem esses objetivos (SILVA et al., 2011). Esta afirmação não descarta nenhuma metodologia e é parte de uma proposta de integração entre elas.

Sintetizar os dois tipos de metodologias de desenvolvimento pode fornecer aos desenvolvedores um espectro compreensivo de ferramentas e opções (BOEHM, 2002). Este estudo também se baseia nisso para propor o modelo ao final, sendo principalmente útil para os casos em que a organização tenha dominado os dois tipos de metodologia, podendo utilizar cada um deles de acordo com a adequação a cada cenário.

Cohen, Lindvall e Costa (2004) apud Dyba e Dingsøyr (2008) acreditam que, cada vez mais, os métodos ágeis sejam consolidados, porém não acreditam que estes vão dominar totalmente sobre os tradicionais, mas sim que irão juntos apresentar um relacionamento simbiótico, nos quais fatores como o número de pessoas trabalhando no projeto, o domínio da aplicação, a criticidade e a inovatividade do produto vão determinar qual processo escolher.

Boehm (2002) acredita que ambos os tipos de abordagem, ágil e tradicional, tem um centro responsável e lateralidades de interpretação. O autor cita também que cada abordagem tem um terreno de características de projeto na qual executa muito bem e muito melhor que o outro, mas que fora de cada terreno, uma abordagem combinada é factível e preferível. Nerur, Mahapatra e Mangalaraj (2005) apresentam um quadro (tabela 2) que compara os dois tipos de metodologias dadas essas características.

Tabela 2 - Quadro comparativo entre características das metodologias tradicionais e ágeis.

	Tradicional	Ágil
Premissas fundamentais	Sistemas são bem especificáveis e podem ser construídos através de planejamento meticuloso e extensivo.	Software de alta qualidade e adaptativo pode ser desenvolvido por times pequenos que usam princípios de melhoria contínua em design e testes, sendo baseados em feedback rápidos e possíveis mudanças.
Controle	Centrado em processos	Centrado em pessoas
Estilo da gestão	Comando-e-controle	Liderança-e-colaboração
Gestão do conhecimento	Explícito	Tácito
Atribuição de papéis	Individuais - favorizam especialização	Times autogeridos - encorajam troca de papéis
Comunicação	Formal	Informal
Papel do cliente	Importante	Crítico
Ciclo do projeto	Guiado por tarefas ou atividades	Guiado por funcionalidades do produto
Modelo de desenvolvimento	Modelo de ciclo-de-vida (Cascata, espiral ou alguma variação destes)	Modelo de entrega evolutiva
Estrutura ou forma organizacional desejada	Mecânica (burocrática, com alta formalização)	Orgânica (flexível e participativa, encorajando ação social cooperativa)
Tecnologia	Sem restrições	Favorece tecnologia orientada a objetos

Fonte: Nerur, Mahapatra e Mangalaraj (2005)

Na área de engenharia de produção, tem-se uma discussão semelhante entre os modelos tradicionais (e. g. taylorismo-fordismo) e os modelos *lean* de produção industrial (como o TPS) para diferenciação e classificação destes de acordo com os seus aspectos, onde são observadas características como autonomia dos trabalhadores, controle do trabalho, padronização e repetição das atividades, trabalho em grupo e qualidade do trabalho (KRAFCIK, 1988). Contextualizando ao

cenário exposto no âmbito de desenvolvimento de software, há também uma discussão emergente sobre a relação entre metodologias ágeis e modelo *lean* de produção industrial (POPPENDIECK, M.; POPPENDIECK, T., 2003; WANG; CONBOY; CAWLEY, 2012; 5CQUALIBR, 2010).

Wheelwright e Clark (1992) também contribuem com a discussão sobre modelos organizacionais, porém focados em tipos de times de desenvolvimento de produto. Em seu trabalho, são discutidos os tipos: estrutura de time funcional, estrutura de time peso leve, estrutura de time peso pesada e estrutura de time autônoma. Na estrutura de time funcional, o trabalho é realizado e completado pelo trabalho seguindo funções organizacionais e é coordenado pelos gestores funcionais de cada área. Na estrutura de time peso leve, os gestores funcionais têm pouca influência sobre o trabalho, apenas através de ligações com os times, e também há influência do gestor de projetos sobre o trabalho na área funcional envolvida com o trabalho em questão. Na estrutura de time peso pesado há um líder, no caso o gestor de projetos, que exerce influência sobre todas as funções envolvidas, integrando-as. Na estrutura de time autônoma, o time peso pesado tem seus integrantes removidos de suas estruturas funcionais, dedicados a apenas um projeto e coalocados.

Existe também no domínio um outro conceito chamado *startup* enxuta que busca eliminar desperdícios ao evitar que sejam feitos produtos que ninguém irá querer e tira seu nome e ideias a partir do TPS, como valor sendo definido como aquilo que proporciona valor ao cliente e o restante sendo desperdício (RIES, 2011). A definição de *startup* apresentada é “uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza”. Decorrente disto, é característica deste modelo usar de iterações rápidas e da percepção dos consumidores sobre seus resultados, buscando avaliar as reações deles, além de inicialmente buscar um entendimento básico de seus problemas (RIES, 2011).

Por fim, também é usado como referência o modelo *stage-gates*, que se trata tanto de um modelo conceitual quanto operacional para movimentar um novo produto da ideia até o lançamento, buscando melhorar efetividade e eficiência no processo (COOPER, 1990). Assim como a *startup* enxuta, este modelo foi pensado para ser

usado em inovação, portanto para lidar com alto risco. Neste modelo: subdivide-se o processo em um número de estágios; entre cada estágio há um *checkpoint* de controle de qualidade chamado *gate*, onde existe um conjunto de entregáveis especificados e que são critérios de qualidade que o produto precisa atender antes de se mover para o próximo estágio. Por sua vez, os estágios são onde o trabalho é realizado e as *gates* garantem então que este tenha sido executado por completo, não havendo erros críticos de omissão ou lacunas no processo.

No artigo original sobre *stage-gates*, Cooper (1990) afirma que o sucesso do produto vem já dos primeiros passos do processo, no pré-desenvolvimento, onde há qualificação e definição de todo o projeto que irá se seguir, principalmente em relação ao atendimento de requisitos e detalhamento do escopo para o desenvolvimento, à identificação de mudanças mais cedo no projeto para evitar que estas ocorram nas fases seguintes de desenvolvimento, onde o custo para tomá-las é maior, ainda que isso tudo torne o projeto um pouco mais longo, devido a esta fase inicial.

Ries (2011) diferencia: a atividade fundamental de uma *startup* enxuta é transformar ideias em produtos, medir como os clientes reagem e aprender se é o caso de pivotar, mudar de rumo caso esteja se afastando do ideal do seu modelo de negócios, ou perseverar, diferentemente das equipes envolvidas com métodos tradicionais de desenvolvimento como cascata ou em estágios apresentada.

Contudo, Cooper (2014) revisou o modelo anterior, onde indica que as práticas e recomendações para a criação de sistemas de inovação ainda parecem com o processo tradicional, havendo estágios onde o trabalho é realizado e *gates* onde as decisões são tomadas, porém agora o apresenta de uma forma mais ágil, rápida, flexível e baseada em riscos: o modelo tornou-se adaptativo, incorporando o modelo espiral ou iterativo para levar o produto aos clientes antecipadamente. Neste novo modelo, o produto pode estar menos do que 50% definido quando entra em desenvolvimento, mas ainda evolui e adapta-se a novas informações conforme se movimenta entre desenvolvimento e testes. Também apresenta versões mais rápidas do processo para projetos de risco mais baixo.

Parte dos fatores identificados nas próximas seções são consequências ou partem das premissas das características apontadas especificamente para um tipo ou outro de metodologia. Apesar disto, este trabalho irá buscar identificar também o quanto a presença de determinado fator pode ser adequada a cada tipo de metodologia, levando em consideração que existem técnicas de "*agile undercover*" e que alguns fatores podem ser favoráveis ou desfavoráveis a ambos os tipos de metodologia. Cada um dos fatores é discutido dentro das referências da literatura em relação à sua adequação, pontos positivos e outras considerações quando da utilização de metodologias tradicionais ou metodologias ágeis.

2.2 – Levantamento e discussão sobre fatores de escolha

Nesta seção, são destacados os fatores mais recorrentes e discutidos na literatura e que possuem relação à adequação de cada tipo de metodologia aos cenários de desenvolvimento. Primeiramente, são apresentadas as características relacionadas à organização. Em seguida, as características relacionadas aos projetos.

2.2.1 - Fatores que compõem o cenário para escolha do tipo de metodologia relacionados à organização e às pessoas

Porte e estrutura organizacional

Aqui são apresentadas as implicações que o porte das organizações ou as diferentes estruturas organizacionais têm sobre o uso dos tipos de metodologia de desenvolvimento.

A transformação ágil é descrita, algumas vezes, como uma mudança de cultura no desenvolvimento e, por isso, a cultura organizacional tem sido mencionada como um fator relacionado às dificuldades encontradas nesta transformação (DINGSØYR; DYBÅ; MOE, 2010). Diferentes modelos de processos ágeis de software têm diferentes premissas e pré-requisitos, assim sendo afetados por características presentes em seus ambientes e projetos (KETTUNEN; LAANTI, 2008). Por causa

disso, muitas companhias precisam primeiramente repensar seus princípios organizacionais e de gestão para então se mostrarem aptas a adotar qualquer uma das metodologias ágeis de software (NERUR; MAHAPATRA; MANGALARAJ, 2005).

Metodologias ágeis requerem uma mudança no estilo de gestão, de comando-e-controle para liderança-e-colaboração, e que a forma da organização para facilitar esta mudança apresente autonomia e cooperação enquanto também forneça flexibilidade e responsividade (CAVALERI; OBLOJ, 2003). Kettunen e Laanti (2008) afirmam que, notavelmente, há uma tensão entre grandes organizações e a proposta do modelo de processo ágil, que envolve times mais autônomos, e também proximidade com o cliente (ou representante deste), devido à tomada de decisão neste caso (em conjunto apenas entre o time e o cliente). Também afirmam que é tipicamente mais fácil de aderir a esta proposta em pequenas organizações do que nas grandes, porque o gerenciamento do negócio já é mais próximo aos desenvolvedores nesses casos.

Segundo Cockburn e Highsmith (2001), o desenvolvimento ágil não é para todos. Impor um processo que abrace mudanças em equipes de projeto mais acomodadas pode não ser razoável. Tentar conseguir colaboração mais próxima com clientes em organizações que têm pouco tempo para os desenvolvedores também não funcionaria. Impor princípios ágeis para organizações centradas em processos, não-colaborativas ou ainda em tentativa de otimização dos processos atuais é provável de falha. Contudo, embora não seja adequado para todos, o desenvolvimento ágil é adequado para muitos.

Cockburn e Highsmith (2001) ainda afirmam que não é que organizações que empregam processos rigorosos valorizam menos as pessoas do que as mais ágeis, mas que as visões que têm das pessoas e como otimizar seu desempenho são diferentes. Processos rigorosos são desenvolvidos para padronizar as pessoas à organização, enquanto processos ágeis são desenvolvidos para tirar proveito de cada indivíduo e das forças únicas presentes em cada equipe: “uma abordagem única para cada caso”, onde todos os processos precisam ser selecionados, customizados e adaptados aos indivíduos de cada equipe de projeto em particular.

Cockburn e Highsmith (2001) também citam casos em que as equipes responderam às requisições de mudanças dos clientes devido a mudanças de plataforma de tecnologias para entregar algo de mais valor ao usuário, mas que o gerenciamento os repreendeu por falta de conformidade ao plano original. Organizações e gestores ágeis entendem que demandar certeza frente à incerteza é disfuncional. Porém, em outro trabalho, Highsmith e Cockburn (2001) citam que seguir o plano não era a meta primária de muitos projetos observados, mas sim a satisfação do cliente.

Os pontos até aqui apresentados que têm alguma relação com a organização e a sua estrutura, como autonomia e comunicação/proximidade entre a equipe e com o cliente, serão abordados em tópicos específicos, dado o enfoque que há sobre esses temas na literatura. Ainda que possuam relacionamento com o fator porte e estrutura organizacional, apresentam outras características que vão além deste escopo. A discussão sobre momentos atuais ou de transição das organizações também não é abordada, porém a escolha de uma metodologia deve estar de acordo com o objetivo da organização.

Highsmith e Cockburn (2001) também afirmam que um aspecto do desenvolvimento ágil é frequentemente esquecido ou encoberto nas discussões: uma visão de mundo de que organizações são sistemas complexos adaptativos. Um sistema complexo adaptativo é um sistema no qual indivíduos descentralizados e independentes interagem de maneira autogerida, guiados por um conjunto de regras generativas - um conjunto mínimo de coisas a serem feitas em todas as situações para gerar práticas apropriadas em situações especiais - utilizado para criar resultados emergentes e inovadores. A diferença para as metodologias tradicionais é que estas fornecem regras inclusivas: todas as coisas que podem ser feitas em todas as situações.

Kettunen e Laanti (2008) investigaram em seu artigo sobre o uso de metodologias ágeis e melhorias de processo de software em grandes organizações de desenvolvimento de novos produtos, onde encontraram que tal tipo de organização, ao atingir todas as dimensões da agilidade:

- Valorizaria pessoas com várias habilidades e seus ativos em capital intelectual;

- Consistiria em muitas organizações virtuais, reconfiguradas dinamicamente, buscando combinar as vantagens de ambas;

- Lançaria frequentemente novas funcionalidades em seus produtos ou novos produtos, baseados nas mudanças de mercado;

- Seus produtos possuiriam designs flexíveis e customizáveis;

- Processos também seriam flexíveis, onde as equipes de desenvolvimento pudessem ser formatadas de acordo com as suas necessidades, utilizando ferramentas que suportassem isso.

Os autores concluem que, em tal organização, equipes de desenvolvimento ágeis seriam naturalmente incorporadas. Como os próprios observaram, mesmo uma organização de grande porte pode estar preparada para o uso de métodos ágeis se, entre outras coisas, permitir que as equipes de desenvolvimento sejam formatadas de maneira adequada para tal. Por isso, o porte da empresa não é um fator decisivo no cenário para a escolha do tipo de metodologia de desenvolvimento de software, mas sim a caracterização da formatação das equipes, que é o fator discutido individualmente no próximo item. Ries (2011) percebe o mesmo quando afirma que o conceito de *startup* enxuta pode funcionar em empresas de qualquer tamanho, de qualquer setor. Para o uso de metodologias tradicionais também não foram encontradas restrições em relação ao porte da empresa.

Com base na literatura, pôde-se decompor o fator aqui discutido em outros que serão abordados em seus tópicos específicos, sendo que alguns deles ainda apresentam características que vão além do porte e da estrutura organizacional e que serão também discutidos de acordo com os estudos existentes.

Formatação das equipes de desenvolvimento

Dadas as diferentes formatações que equipes de desenvolvimento possam apresentar, há diferentes implicações sobre a adequação do uso de metodologias tradicionais e metodologias ágeis de desenvolvimento.

A partir da discussão sobre os diferentes modelos de organização, pode-se então buscar base nos modelos focados em times de desenvolvimento apresentados por Wheelwright e Clark (1992) como referência para a escolha da metodologia mais adequada a ser adotada.

Entre os modelos propostos por Wheelwright e Clark (1992), o modelo funcional é o que apresenta menor autonomia para o grupo de trabalho, com o gestor funcional concentrando as responsabilidades e a tomada de decisão, além do grupo seguir mais o que é definido para a organização como regras gerais. O modelo autônomo é o que apresenta o maior grau de autonomia para o grupo e, inclusive, propõe alocar os seus membros juntos durante o processo de desenvolvimento, com o gestor de projetos tendo o poder de tomada de decisão em lugar do funcional e havendo também a possibilidade de definição de regras internas, próprias do grupo de trabalho. Os outros modelos (times pesos-leves e times pesos-pesados) representam graus intermediários nestes quesitos de acordo com a autonomia, a alocação de recursos e os papéis na tomada de decisão.

Ainda sobre este assunto, Cockburn e Highsmith (2001) citam que uma ideia dominante no desenvolvimento ágil é a de que a equipe pode ser mais efetiva em responder a mudanças se puder reduzir o custo da movimentação de informação entre as pessoas e se puder reduzir o tempo gasto entre a tomada de decisão e ver as consequências das decisões.

Assim como modelos ágeis, o modelo *stage gates* requer que a organização tenha times por projetos, não podendo haver manipulação de departamento para departamento: um time e um líder precisam tocar o projeto em todos os estágios, senão o tempo torna-se demasiado longo. Também são exigidos seniores como

gatekeepers para a aprovação dos entregáveis, pois facilita o sucesso no modelo, requerendo recursos significantes e demandando a aceitação da alta gestão (COOPER, 1990).

Cockburn e Highsmith (2001) também citam que gestores ágeis de projetos focam em estabelecer um relacionamento colaborativo com os clientes, enquanto os tradicionais focam em seguir os contratos, que são detalhados sobre todo o sistema, além dos custos e prazos.

Nerur, Mahapatra e Mangalaraj (2005) complementam que o tradicional papel de planejamento e controle do gestor de projeto precisa ser alterado para o de um facilitador que direciona e coordena os esforços colaborativos dos envolvidos no desenvolvimento, também garantindo que as ideias criativas de todos eles sejam refletidas na decisão final (HIGHSMITH, 2003). O maior desafio neste aspecto é conseguir fazer os gestores de projetos renunciarem à autoridade que possuíam anteriormente e também, se necessária, a mudança de cultura para haver confiança e respeito dentro da equipe de forma que facilite a tomada de decisões, o que pode precisar de grande esforço, tempo e paciência.

Algumas das questões aqui apresentadas serão melhor detalhadas nos tópicos facilidade de comunicação e proximidade entre os membros da equipe e facilidade de comunicação e proximidade com o cliente. Também há bastante envolvimento com o fator autonomia, que é o próximo tópico abordado. Contudo, a formatação das equipes de desenvolvimento aparece como um fator importante nesse estudo.

Autonomia para os grupos de trabalho

Aqui são discutidas as implicações do grau de autonomia permitido para a equipe de desenvolvimento para a aplicação dos diferentes tipos de metodologia.

Hoda, Noble e Marshall (2011) afirmam que times ágeis são times autogeridos (COCKBURN; HIGHSMITH, 2001) compostos de indivíduos que gerenciam sua própria carga de trabalho, mudam de posições de trabalho entre eles baseados nas

necessidades e melhor adequação e participam de tomadas de decisão em equipe (HIGHSMITH, 2004). Cockburn e Highsmith (2001) ainda complementam sobre haver intensa colaboração dentro e além dos limites da organização.

Takeuchi e Nonaka (1986) descrevem que times autogeridos apresentam autonomia, fertilização cruzada e autotranscedência. Times autogeridos precisam manter um foco em comum, confiança mútua, respeito e a habilidade de se organizar repetidamente para enfrentar novos desafios (COCKBURN; HIGHSMITH, 2001). Times autogeridos não são times sem controle ou sem líderes (COCKBURN; HIGHSMITH, 2001; TAKEUCHI; NONAKA, 1986), mas sim com menos rigor e adaptáveis, fornecendo feedback e direcionamento mais sutil (TAKEUCHI; NONAKA, 1986).

Boehm (2002) cita que há pouca evidência de que a agilidade funcionaria na ausência de pessoas mais competentes (com conhecimento ou experientes). Nerur, Mahapatra e Mangalaraj (2005) complementam que isto pode causar sérios problemas relacionados à montagem da equipe e à moral desta devido à dificuldade de encontrar pessoas competentes o suficiente para compor o time e devido à cultura de elitismo que pode surgir neste grupo de desenvolvimento, afetando a moral dos desenvolvedores “não-ágeis”. Já Cockburn e Highsmith (2001) afirmam que se trata mais de uma atitude cujo objetivo é trabalhar em conjunto justamente para melhorar o conhecimento e as habilidades dos indivíduos. Dybå e Dingsøyr (2008) citam que estudantes universitários percebem o uso de tais metodologias como um treinamento relevante para seus futuros trabalhos, embora encontrem algumas dificuldades justamente quando há uma grande diferença de habilidades entre os membros do time.

Boehm (2002) também cita que isso não significa que métodos ágeis requerem somente pessoas com alta capacidade envolvidas, citando que muitos projetos obtiveram sucesso com misturas de pessoal experiente e juniores, dada que a maior diferença em relação aos projetos tradicionais é a de que os métodos ágeis devem muito de sua agilidade ao fato do conhecimento tácito estar presente na equipe, ao invés de escrito em planos (HIGHSMITH, 2002). Todavia, um time de

desenvolvedores inexperientes pode produzir uma especificação com um padrão diferente em níveis de detalhe: uma elaboração rica em detalhes para os elementos bem conhecidos e de baixo risco e uma elaboração pobre dos elementos pouco conhecidos e de alto risco (BOEHM, 1988). Boehm (1988) complementa ao dizer que a menos que haja uma revisão perspicaz de tal especificação por alguém experiente, este tipo de projeto pode levar a uma ilusão de progresso por algum período sendo que, na verdade, está indo em direção ao desastre. O modelo *stage gates* trata isso ao alocar um sênior para as *gates* (COOPER, 1990). Ries (2001) associa a autonomia com o tratamento de incertezas, onde afirma que *startups* precisam de autonomia completa para tal.

Segundo Boehm (2002), em relação ao modelo ágil, quando o conhecimento tácito do time é suficiente para as necessidades do desenvolvimento da aplicação, as coisas funcionam bem, mas também há o risco de o time cometer erros de arquitetura irrecuperáveis devido à tomada de rumos não conhecidos pelo seu conhecimento tácito. Métodos mais tradicionais reduzem tal risco ao investir em arquitetura e planejamentos por ciclo-de-vida, o que facilita também revisões por especialistas externos para cada uma das etapas, porém aceitam o risco de que mudanças rápidas irão tornar os planos obsoletos ou muito caros de se manterem atualizados. Embora tenha relação com este tópico, o fator que aborda riscos e incertezas será tratado em outra seção.

Ainda sobre isso, Boehm (2002) apresenta uma abordagem que algumas metodologias ágeis utilizam denominada de “você não vai precisar disso”, ou seja, não fazer mais trabalho do que o necessário e evitar desperdício. Ao falar sobre riscos, o autor cita que o replanejamento é simples quando ocorre num time com ótimos desenvolvedores desenvolvendo pequenos sistemas, assim essa abordagem além de válida é de baixo risco. Contudo, evidências empíricas encontradas pelo autor apontam que, com desenvolvedores não tão ótimos assim, os esforços para replanejamento aumentam com o número de requisitos ou estórias envolvidas no projeto. Desta forma, assim como os riscos, o fator complexidade também está relacionado a este fator, mas será tratado num tópico à parte no que se relaciona ao software ou sistema gerado ou alterado em um projeto.

Cockburn e Highsmith (2001) citam que o desenvolvimento ágil foca nos talentos e habilidades dos indivíduos, sendo estes até fatores críticos para o sucesso dos projetos, moldando os processos de acordo com as pessoas e times específicos. Citam também que se as pessoas são boas o suficiente, então elas podem usar quase qualquer processo e completar suas tarefas com sucesso, mas que se não são boas o bastante, nenhum processo irá reparar a inadequação, pois “pessoas atropelam processos”. Assim, mesmo nos processos de metodologias tradicionais, as pessoas também precisam ser boas o suficiente para executar suas tarefas corretamente (COCKBURN; HIGHSMITH, 2001).

Cockburn e Highsmith (2001) ainda complementam que pessoas trabalhando juntas e com boa comunicação e interação podem operar em níveis notavelmente maiores do que quando usam seus talentos individuais. Isto seria uma vantagem para as metodologias ágeis que propõem um time trabalhando fisicamente próximos e cooperativamente.

Cockburn e Highsmith (2001) enfatizam vários fatores críticos relacionados a pessoas para os métodos ágeis: amigabilidade, talento, habilidade e comunicação. Neste estudo, o fator autonomia não será decomposto e, dado o que se conhece, espera-se que este fator reflita o que se pensa em conjunto sobre todos estes aspectos relacionados às pessoas, seus conhecimentos e o grau de autonomia que é permitido à equipe pela organização. Apenas a questão da comunicação entre os membros da equipe será tratada em tópico específico, pois também tem forte influência de questões relacionadas aos projetos.

Alinhamento a normas e boas práticas

O alinhamento dos processos, atividades e artefatos gerados pelo uso dos tipos de metodologias a possíveis normas e modelos adotados ou buscados pela organização (como ISOs, CMMI, acordos internacionais e normas legislativas) também pode ter implicações na escolha de um outro tipo de metodologia.

Silva et al (2011) investigam o uso combinado de métodos ágeis, que possuem processos mais leves para tratar os requisitos de software, e SPL, que apresenta características de metodologias tradicionais. A proposta encontrada é manter o foco no processo de desenvolvimento, mas utilizar os artefatos importantes durante o andamento do projeto. SPL, assim como CMMI e outros frameworks, propõem modelos com áreas de processos a serem desenvolvidas de acordo com as melhores práticas de mercado e que servem também para obtenção de certificações e de avaliação externa para as companhias que desenvolvem software, o que faz com que estes sejam seguidos e desejados pelas companhias.

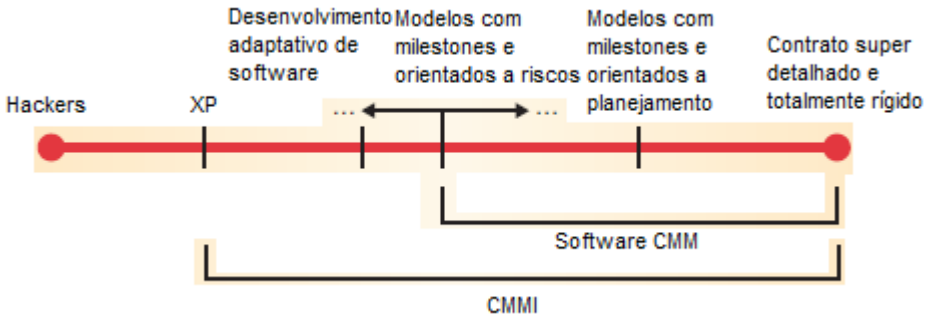
Silva et al (2011) apontam que a definição da arquitetura, desenvolvimento de componentes, engenharia de requisitos, testes, escopo e ferramentas de suporte são as áreas de SPL focadas em processos de desenvolvimento de software nas quais os métodos ágeis têm mais aplicabilidade. Práticas ágeis como programação pareada, desenvolvimento orientado a testes, refatoração e códigos coletivos são as mais frequentemente aplicadas. Já áreas como a mineração de ativos, compreensão de domínios relevantes, utilização de softwares externos disponíveis, análise do que fazer / comprar / minerar / comissionar e disciplina dos processos são áreas práticas mais organizacionais, onde métodos ágeis com práticas organizacionais mais fortes podem contribuir, como *Lean*, DSDM e ASD, embora não haja estudos empíricos o suficiente sobre o resultado da aplicação destas metodologias como nas áreas anteriormente citadas. Apesar do uso de SPL como referência, outros frameworks, como CMMI, apresentam as mesmas áreas ou similares e que mantém a validade dessas comparações. Os autores também citam que os métodos mais populares no contexto de SPL são XP, Scrum e TDD, pois estes métodos possuem práticas que consideram tanto aspectos de desenvolvimento quanto de gestão, assim abrangendo um escopo maior de contribuição às práticas de SPL. Uma observação importante dos autores é que não foram encontradas evidências de que a aplicação de um princípio ágil force o uso de outro, podendo ser usado livremente sem os outros processos da metodologia ao qual faz parte.

Nerur, Mahapatra e Mangalaraj (2005) citam que a gestão do conhecimento é de importância vital para as organizações. Nesse ponto, o desenvolvimento tradicional

é conhecido por criar muita documentação, mas principalmente artefatos utilizados para comunicação e histórico de desenvolvimento. Por outro lado, metodologias ágeis encorajam o pensamento mais enxuto, que corta sobrecarga, principalmente em documentação. Desta forma, muito do conhecimento no desenvolvimento ágil reside nas cabeças dos membros das equipes (BOEHM, 2002; HIGHSMITH, 2003), o que pode tornar a organização muito dependente dos membros dos times de desenvolvimento e ainda transferir o poder do gerenciamento para os times, o que pode não ser aceitável para muitas organizações. Contudo, isso pode ser resolvido ao se determinar que tipos de conhecimento devem ser codificados e quais permanecerem tácitos. A questão da gestão de conhecimento é parte de muitas das boas práticas e normas seguidas pelas organizações e por isso é levada em consideração neste tópico.

Boehm (1988) destacou o ponto de que os modelos tradicionais de processos de software (existentes até então) desencorajavam abordagens mais efetivas para o desenvolvimento de software, como prototipação e reuso. Sobre a metodologia espiral, cita que esta prepara, durante seu ciclo-de-vida, o produto de software para evolução futura, expansão e mudanças, pois já o prepara para acomodar melhor os riscos conhecidos que o permeiam. Contudo, já identifica como desafio a dependência em conhecimento e experiência para a gestão riscos nesta abordagem.

Figura 6 - Espectro de planejamento.



Fonte: Boehm (2002)

Boehm (2002) mostra a localização do CMM e seu sucessor, CMMI, no espectro de planejamento, onde o *"hacking"*, com falta de planejamento e indisciplinado, ocupa a extrema esquerda, enquanto o planejamento com marcos microgerenciados ocupa a

extrema direita (figura 6). Os proponentes do uso de Software CMM normalmente o utilizam de uma forma restritiva e burocrática, mas este pode ser interpretado de forma a incluir algumas formas de modelos com marcos orientados a riscos (como o modelo espiral). Alguns líderes no assunto também buscam mostrar como interpretações liberais, não tão literais, ainda podem incluir grandes porções de XP, método ágil mais focado em técnicas para codificação e assuntos mais próximos a estas, como sendo adequadas também ao Software CMM (PAULK, 2001). Porém, por fim, o autor cita que é possível interpretar o CMMI, que inclui áreas de processos para gestão de risco, integração dos times e um ambiente organizacional mais favorável a tal integração, de forma a incluir métodos ágeis.

Boehm (2002) ainda cita que outros objetivos dos métodos mais tradicionais de desenvolvimento e do Software CMM são previsibilidade, repetibilidade e otimização, mas complementa que num mundo de mudanças frequentes e radicais, ter processos repetitíveis e otimizados pode não ser um bom objetivo. Já o CMMI e os métodos mais orientados a riscos, ainda que trazendo características tradicionais, fornecem meios para se migrar ao objetivo de flexibilidade.

Segundo Nerur, Mahapatra e Mangalaraj (2005), o problema em mudar as atitudes e práticas de “centradas em processos” para “centradas em pessoas” é mais suscetível, por exemplo, em organizações que tentaram por anos alcançar os níveis mais altos do CMM. A ideia de mudar um processo para se adequar às capacidades e competências das pessoas e das características dos projetos ao invés de utilizar um processo rígido que segue atividades padronizadas pode parecer interessante (COCKBURN; HIGHSMITH, 2001) mesmo nesses casos, mas pode ser alcançada somente através de investimentos significativos de tempo, esforço e capital.

Apresentando este conjunto de aspectos, avaliar a adequação a melhores práticas e normas de ambos os tipos de metodologia pode apresentar um resultado decisivo na escolha de qual tipo utilizar. Este fator inclui diversos aspectos importantes aqui apresentados, porém riscos e incertezas serão tratados à parte, dadas as suas importâncias na literatura.

2.2.2 Fatores que compõem o cenário para escolha do tipo de metodologia relacionados aos projetos de desenvolvimento de software

Nesta seção, serão destacados os fatores mais recorrentes e discutidos na literatura que possuem relação à adequação de cada tipo de metodologia aos projetos de desenvolvimento de software.

Duração do projeto

Projetos avaliados de acordo com seus prazos podem ser mais ou menos adequados em relação ao que se conhece sobre o uso dos diferentes tipos de metodologias de desenvolvimento de software.

Kettunen e Laanti (2008) reconhecem em seu trabalho que é difícil escalar a agilidade para projetos grandes de software, mas que há exemplos de como se fazer isto (ECKSTEIN 2004 apud KETTUNEN; LAANTI, 2008; MCMAHON 2005 apud KETTUNEN; LAANTI, 2008). Os autores complementam que, em tais cenários, qualquer mudança tem impactos maiores, assim o alcance da agilidade também precisa ser amplo e os focos das mudanças deveriam passar da implementação para a antecipação, não apenas reagir às mudanças quando elas aparecerem.

O foco de métodos ágeis em antecipar resultados é bastante compatível com o desenvolvimento de sistemas pequenos, mas para sistemas grandes pode levar a um grande retrabalho caso a arquitetura desenvolvida não seja escalável, pois nesses casos um bom planejamento se faz necessário (BOEHM, 2002).

Boehm (2002) cita que métodos tradicionais escalam melhor para projetos grandes, mas que não teriam eficiência em projetos pequenos, principalmente se estes acontecerem em uma organização burocrática, que requer muito esforço para conseguir autorizar um projeto e iniciá-lo. Esta visão envolve autonomia, a duração do projeto e a quantidade de pessoas envolvidas, que será tratada em outro tópico.

Ainda assim, foi quando os projetos começaram a se tornar maiores e mais complexos que problemas como requisitos estagnados e programação má estruturada começaram a aparecer na utilização do modelo cascata (HILKKA, MATTI; TUURE, 2005)

O tamanho do produto final gerado não apresenta alguma correlação com o método de desenvolvimento utilizado (diferentes estudos empíricos reportaram tamanhos maiores, menores ou iguais para os resultados comparativos de produtos de software gerados utilizando-se métodos tradicionais e ágeis), o efeito das práticas e a satisfação no trabalho utilizando métodos ágeis ou tradicionais não foram conclusivamente estabelecidos ainda (DYBÅ; DINGSØYR, 2008). Ainda assim, este é um fator recorrente na literatura e será estudado.

Complexidade do projeto

O grau de complexidade e o tipo de software ou sistema desenvolvido a cada projeto podem apresentar variações e melhores adequações a um tipo ou outro de metodologia de desenvolvimento.

Boehm (1988) cita que uma fonte primária de dificuldade com o modelo cascata foi sua ênfase em documentos muito elaborados como critérios de completude para fases primárias de requisitos e design. Para algumas classes de software como compiladores ou sistemas operacionais de segurança, este é o modo mais efetivo de se proceder, porém isso não funciona bem para muitas classes de software, particularmente aplicações interativas para o usuário final. Padrões orientados à documentação levaram muitos projetos a terem suas especificações de interfaces ao usuário muito elaboradas e funções de suporte à decisão fracamente compreendidas no início do desenvolvimento, seguidas pelo design e desenvolvimento de grandes quantidades de código não utilizado ao final. Tais projetos são exemplos de como o modelo cascata pode apresentar dificuldades devido à ordem de seus estágios. Além disso, em áreas suportadas por linguagens de quarta geração (planilhas ou aplicações de pequenos negócios) é claramente

desnecessário escrever especificações muito elaboradas para uma aplicação antes de implementá-la.

Todas essas questões são relacionadas aos possíveis tipos de software que um projeto de desenvolvimento pode ter como fim. Não há a proposta neste trabalho de definir e classificar todos os tipos existentes de softwares a serem gerados, porém de avaliar a adequação a um tipo ou outro de metodologia dadas as complexidades e riscos que apresentam durante o desenvolvimento.

Por outro lado, sobre agilidade, apenas ter iterações mais curtas, ainda que resultando em melhor qualidade, pode não ser o suficiente para grandes companhias produzindo softwares complexos, onde uma visão mais holística pode ser necessária (KETTUNEN; LAANTI, 2008). Aqui pode ser observada novamente a questão da escalabilidade das metodologias ágeis, enquanto para os métodos tradicionais foram apontados o desperdício gerado e a inadequação ao que os clientes esperavam.

Nerur, Mahapatra e Mangalaraj (2005) apontam que existe uma forte questão sobre complexidade dos sistemas relacionada ao conhecimento dos clientes sobre estes. Ainda que importante para a discussão como um todo, o fator relacionado à contribuição dos clientes será abordado em outro tópico. A complexidade aqui tratada é focada na definição em algum momento sobre o desenvolvimento do projeto ou produto final.

Kettunen e Laanti (2008) citam que muitos produtos de softwares comerciais e outros novos produtos que contém softwares embarcados são inovadores por natureza, mas que há muitos tipos de produtos de software que não são, como aqueles com propósito final específico já conhecidos ou regulados por legislação. Complementam que, todavia, mesmo em áreas aparentemente estáveis, mudanças disruptivas podem surgir, requisitando respostas rápidas e muitas vezes radicais durante o desenvolvimento. Assim sendo, um estudo sobre os diferentes tipos de software gerados pode considerar quão inovadores estes podem ou precisam ser e a relação que podem apresentar com os tipos de metodologia de desenvolvimento.

Contudo, aqui restringem-se novamente a serem considerados como mais ou menos complexos ou com maiores ou menores riscos, onde os produtos inovadores frequentemente estão entre os que apresentam mais riscos. A questão sobre se adequar a alguma legislação é semelhante à adequação a melhores práticas, porém com mais rigor.

Ainda sobre inovação, Nerur, Mahapatra e Mangalaraj (2005) citam que formas e culturas organizacionais que conduzem à inovação podem adotar métodos ágeis mais facilmente do que aquelas baseadas em burocracia e formalização. Essas características podem ser um reflexo direto dos produtos de software gerados por tais organizações.

Cockburn e Highsmith (2001) citam em seu trabalho que Doug DeCarlo, pesquisador e consultor, afirma que a gestão tradicional de projetos sofre do que chama de uma “neurose newtoniana”, ou seja, tenta atacar problemas complexos e não-lineares com processos simples e lineares. Os autores ainda complementam que o desenvolvimento ágil tem excelência em problemas exploratórios, que são os casos de projetos em situações extremas, complexas ou com alto índice de mudanças e que estes são melhor atacados quando em uma cultura organizacional colaborativa e centrada em pessoas.

Segundo Silva et al (2011), suporte automatizado é uma necessidade para gerenciar a complexidade e variedade inerente a linhas de produção de software, principalmente no uso dos métodos ágeis. Atualmente, existem ferramentas preparadas tanto para os processos, atividades e artefatos de metodologias ágeis quanto de metodologias tradicionais. Ferramentas, assim como pessoas bem treinadas para o uso das metodologias, são fatores que influenciam na melhor aplicação de qualquer uma das metodologias, dependendo de para qual são preparadas. Assim como a empresa precisa ter domínio de ambos os tipos de metodologia para poder fazer a escolha entre um tipo ou outro, as pessoas e ferramentas também devem ser adequadas aos dois.

Quantidade de pessoas no projeto

Nesta seção é discutido como projetos são avaliados de acordo com a quantidade de pessoas na equipe em relação ao que se conhece sobre o uso dos diferentes tipos de metodologia.

Kettunen e Laanti (2008) citam que as metodologias ágeis evoluíram para atender às necessidades de times de software pequenos e flexíveis, mas que, atualmente, há uma tentativa de ampliar essas metodologias para melhor se adequarem aos projetos e organizações maiores interessadas em aplicá-los. Porém, em projetos de desenvolvimento em larga escala de produtos, as premissas originais das metodologias ágeis não se adequam necessariamente: pré-requisitos adicionais e condições organizacionais precisam ser antes satisfeitas para alcançar todos os benefícios do ágil. É necessário conhecimento sobre organizações ágeis e os fatores que permitem e previnem de alcançar agilidade nas grandes organizações.

Por outro lado, métodos tradicionais escalam melhor para projetos grandes, mas em uma organização burocrática, que requer muito para conseguir autorizar um projeto e iniciá-lo, não teriam a mesma eficiência que se aplicados a projetos pequenos (BOEHM, 2002). Isto envolve tanto a duração do projeto quanto a quantidade de pessoas envolvidas nele, além de autonomia.

Cockburn e Highsmith (2001) citam que, de fato, o desenvolvimento ágil é mais difícil para times maiores, porém citam também projetos de sucesso com times com essa característica, de até 250 pessoas. Na revisão de Dybå e Dingsøyr (2008), apenas três artigos investigaram projetos com mais de 50 envolvidos no total. Petersen e Wohlin (2009) citam que os resultados são difíceis de generalizar e que pouco é conhecido sobre as vantagens e considerações sobre o uso de práticas ágeis e incrementais no desenvolvimento de software em larga escala.

Facilidade de comunicação e proximidade entre os membros da equipe

Nesta seção os projetos são avaliados de acordo com a facilidade na comunicação ou proximidade entre as pessoas da equipe em relação ao que se conhece sobre o uso das diferentes metodologias.

Nerur, Mahapatra e Mangalaraj (2005) apontam como crítico o processo social e cooperativo caracterizado pela comunicação e colaboração entre uma comunidade de membros que se valorizam e confiam uns nos outros (COCKBURN; HIGHSMITH, 2001; HIGHSMITH 2003) para o uso de metodologias ágeis. Para programadores acostumados a atividades solitárias ou a trabalhar em grupos relativamente homogêneos de analistas e designers, as ideias de aprendizagem compartilhada, reflexão, workshops, programação pareada e tomada de decisão colaborativa das metodologias ágeis podem ser difíceis de suportar.

Cockburn e Highsmith (2001) citam que os times ágeis são caracterizados por auto-organização e colaboração intensa, dentro e além dos limites da organização. Os autores diferenciam colaboração de comunicação: colaboração é trabalharem ativamente unidos para entregar um produto ou tomar uma decisão; comunicação é o envio e recebimento de informação. Todavia, dada a abordagem deste estudo, o fator aqui discutido inclui os dois aspectos na discussão, pois tal diferenciação não será repassada no momento da aplicação da metodologia.

Sobre comunicação, Cockburn e Highsmith (2001) citam que uma ideia dominante no desenvolvimento ágil é a de que a equipe pode ser mais efetiva em responder a mudanças se puder reduzir o custo da movimentação de informação entre as pessoas e se puder reduzir o tempo gasto entre a tomada de decisão e ver as consequências das decisões. Os autores complementam que, para reduzir o custo de movimentação de informação entre pessoas, o time ágil busca colocar as pessoas fisicamente próximas, substituir documentos por conversas entre pessoas ou uso de quadros e aumentar a amigabilidade do time (senso de comunidade e moral) para poderem compartilhar informações de valor mais rapidamente.

Em outro trabalho, Highsmith e Cockburn (2001) afirmam que pessoas podem transferir ideias mais rapidamente se conversarem pessoalmente do que através da leitura e escrita de documentos, além de afirmarem que alguns desenvolvedores sentados juntos podem produzir um design melhor do que cada um produziria individualmente. Assim, posicionar as pessoas corretamente pode permitir efetivamente atingir agilidade, rapidez e redução de custos.

Viabilidade de comunicação e proximidade com o cliente ou área solicitante

Nesta seção os projetos são avaliados de acordo com a facilidade na comunicação e proximidade entre as pessoas da equipe e o cliente em relação ao que se conhece sobre o uso das diferentes metodologias.

Cooper (1990) cita que a falta de avaliação de mercado foi bastante apontada por anos como o motivo número um para falha de novos produtos. O autor também cita que empresas que tiveram mais atividade relacionadas ao mercado / cliente em seus desenvolvimentos obtiveram mais benefícios com o produto.

Highsmith e Cockburn (2001) enfatizam que a proximidade entre os membros da equipe e uma intensa interação entre eles são marcas dos métodos ágeis. Segundo Nerur, Mahapatra e Mangalaraj (2005), o sucesso de desenvolvimento ágil se sustenta em encontrar clientes que irão participar ativamente no processo de desenvolvimento. Espera-se que os consumidores sejam “colaborativos, representativos, autorizados, comprometidos e tenham conhecimento” (BOEHM, 2002; BOEHM; TURNER, 2004). Como citado anteriormente, principalmente para sistemas complexos, esta não é uma tarefa fácil.

Cockburn e Highsmith (2001) afirmam que, para reduzir o tempo entre uma decisão e seu feedback, o time ágil mantém clientes com conhecimento disponíveis para o time ou ainda fazem parte deste. Estes clientes, ao verem o software em desenvolvimento antecipadamente em seus estágios, percebem tanto aquilo que os desenvolvedores não entenderam muito bem quanto quais de suas requisições não funcionam bem na prática. Boehm (2002) também reafirma que métodos ágeis

funcionam melhor com os clientes dedicados ao time de desenvolvimento, desde que possuam conhecimento tácito suficiente para completar a aplicação.

Novamente cita-se que, segundo Highsmith e Cockburn (2001), posicionar as pessoas corretamente pode permitir efetivamente atingir agilidade, rapidez e redução de custos. Em relação aos clientes e patrocinadores, as conversas entre a equipe e estes podem resolver dificuldades, ajustar prioridades e examinar caminhos alternativos de forma que não seria possível se não estivessem unidos. Para tal colaboração, faz-se necessário que todos os envolvidos (patrocinador, cliente, usuário e desenvolvedor) estejam no mesmo time e que a combinação de suas diferentes experiências e conhecimentos permitam ao grupo mudar os rumos rapidamente, podendo produzir resultados mais apropriados. Boa-fé também é necessária, pois mesmo havendo contratos, sem tal colaboração, estes seriam insuficientes para a obtenção de bons resultados.

Highsmith e Cockburn (2001) complementam que um time não é ágil se o loop de feedback com os clientes levar muito tempo entre suas repetições. As abordagens ágeis recomendam iterações curtas, entre duas a seis semanas, durante as quais a equipe faz constantes decisões e ajustam novas informações com os clientes, como o planejamento das funcionalidades e priorização dinâmica (o cliente pode repriorizar as funcionalidades originalmente planejadas, bem como descartar ou adicionar novas). Um ponto-chave nas abordagens ágeis é que estas planejam funcionalidades, que é o que o cliente entende bem, e não tarefas.

Hoda, Noble e Marshall (2011) também citam que, apesar das práticas e princípios ágeis assumirem os cenários expostos para correta aplicação, os níveis de colaboração do cliente podem variar amplamente em diferentes contextos de projetos reais. Os métodos ágeis não estariam preparados para os contextos onde os níveis de colaboração não sejam os ideais, como problemas com clientes que demandam contratos com custo fixo e também aqueles difíceis de se manter proximidade durante todo o desenvolvimento. Contudo, os problemas encontrados levaram as equipes a desenvolver e utilizar estratégias de "*agile undercover*" para contornar tais situações.

Highsmith e Cockburn (2001) reconhecem que se os clientes ou áreas solicitantes não possuem um bom senso de direção para o que querem, os desenvolvedores ágeis acabarão por seguí-los mesmo assim. Segundo Boehm (2002), se os clientes não apresentarem os requisitos citados, os produtos gerados normalmente não obtêm o sucesso esperado, ainda que acabem por satisfazê-los. O autor ainda complementa que, também neste caso, estes métodos também se arriscam por caminhos orientados apenas ao conhecimento tácito, onde os métodos tradicionais reduzem os riscos através da elaboração de documentação, revisão de arquitetura e revisões de experts independentes. Clientes falhos resultam em sistemas falhos (HIGHSMITH; COCKBURN, 2001).

Contudo, Hoda, Noble e Marshall (2011) acharam em seu trabalho que a falta de envolvimento do cliente é um dos grandes desafios encontrados pelos times ágeis. Essa falta de envolvimento causa problemas adversos às propostas dos métodos ágeis. Porém, as estratégias de "*agile undercover*" encontradas pelos autores ajudam equipes que apresentam esta situação a contornarem-na.

Hoda, Noble e Marshall (2011) identificaram em seu trabalho as causas e consequências dessa falta de envolvimento dos clientes. Entre as causas, encontraram: ceticismo ou visão de moda passageira, a distância física, falta de comprometimento, lidar com clientes muito grandes, contratos com orçamento fixo e representantes não eficazes dos clientes. E entre as consequências: pressão para realizar as tarefas, problemas em coletar e esclarecer requisitos, problemas em priorizar os requisitos, problemas em garantir o feedback, perda de produtividade e, nos piores casos, quebra do acordo para o desenvolvimento.

Times ágeis desenvolveram diferentes estratégias para assegurar a boa aplicação das metodologias de acordo com os níveis de variação do envolvimento do cliente. São estratégias, do maior para o menor nível de envolvimento (figura 7) (HODA; NOBLE; MARSHALL, 2011):

- Donos das estórias - onde os membros da organização compartilham a responsabilidade do papel de cliente e estão disponíveis quando requisitados;

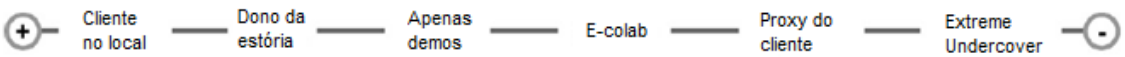
- Apenas demonstrações - onde o nível de envolvimento do cliente é limitado a participar nas demonstrações no final de iterações;

- *E*-colaboração - onde o time interage com o representante do cliente através de meios eletrônicos, como videoconferência;

- *"Proxy"* de cliente – onde há alguém do time representando o papel do cliente na ausência de um cliente real;

- *"Extreme Undercover"* - onde o cliente não sabe da natureza ágil do projeto, ou seja, o uso da metodologia é encoberto e aplicado somente entre desenvolvedores da equipe.

Figura 7 - Contínuo do envolvimento dos clientes em projetos ágeis.



Fonte: Hoda, Noble e Marshall (2011)

Nerur, Mahapatra e Mangalaraj (2005) citam que, em um ambiente ágil, o time de desenvolvimento, composto por desenvolvedores de software e o cliente, toma a maioria das decisões e isto cria um ambiente pluralista de tomada de decisões devido aos diversos backgrounds, atitudes, metas e disposições cognitivas destes membros (CAVALERI; OBLOJ, 1993). Hoda, Noble e Marshall (2011) ainda complementam que os clientes precisam perceber suas responsabilidades para garantir o sucesso de projetos ágeis autogeridos e, por sua vez, o sucesso de seus projetos.

Boehm (1988), sobre métodos tradicionais, diz que o modelo espiral tem a correspondência apropriada ao contrato de software como um dos desafios primários. A dificuldade está em apresentar flexibilidade e liberdade no projeto sem perder contabilidade e controle e também em definir contratos nos quais as entregas

não estejam bem especificadas. Certo progresso surgiu posteriormente no estabelecimento de mecanismos mais flexíveis nos contratos, como o uso de contratos *"front-end",* mais competitivos, para a definição de conceitos ou lançamentos de protótipos, o uso de contratos a nível-de-esforço ou "*award-fee*" para desenvolvimento evolutivo e o uso de contratos *"design-to-cost"*. O autor complementa que, embora geralmente obtenham sucesso, os procedimentos para uso destes tipos de contrato ainda precisam ser trabalhados de forma que os gestores de aquisição sintam-se completamente confortáveis em utilizá-los.

Hoda, Noble e Marshall (2011) sugerem a condução de estudos em áreas mais tradicionais de desenvolvimento de software para a exploração do nível de envolvimento dos clientes, suas consequências para os times de desenvolvimento e as estratégias que os times devem propor para superar os problemas decorrentes disto, bem como sugerem um estudo comparativo sobre o impacto da colaboração dos clientes em métodos ágeis versus a mesma colaboração em métodos tradicionais.

Risco e incerteza do projeto

Nesta seção, os projetos são avaliados de acordo com os riscos e incertezas em relação ao que se conhece sobre o uso dos diferentes tipos de metodologias.

Nerur, Mahapatra e Mangalaraj (2005) citam que o desenvolvimento de software é uma atividade complexa, caracterizada por tarefas e requisitos que apresentam um alto grau de variabilidade (BOEHM; TURNER 2004; HIGHSMITH 2003). Incertezas são agravadas pela diversidade e imprevisibilidade das pessoas que se engajam em tais tarefas (COCKBURN; HIGHSMITH, 2001; COCKBURN 2002). A natureza de mudança e sofisticação de ferramentas também pode aumentar os problemas causados por tais fenômenos. Kettunen e Laanti (2008) também citam os turbulentos ambientes do mercado global, onde estão inseridas as companhias desenvolvedoras de software, como fontes de muitas das incertezas (MULLINS; SUTHERLAND, 1998 apud KETTUNEN; LAANTI, 2008). Neste cenário, segundo os autores, a agilidade fornece uma vantagem competitiva para as organizações.

Tabela 3 - Lista priorizada dos dez maiores itens de risco para software e técnicas para enfrentá-los.

Item de risco	Técnicas de gerenciamento de riscos
1. Perdas de pessoas	Elencar talentos que combinem com o trabalho; treinamento cruzado; pré-seleção de pessoas-chave
2. Prazos e orçamentos não-realísticos	Estimativa detalhada dos prazos e custos por várias fontes; desenvolvimento incremental; reuso de software; enxugamento de requisitos
3. Desenvolvimento das funções erradas de software	Análise organizacional; análise da missão; formulação de conceitos de operação; questionário com usuários; prototipação; antecipação de manuais do usuário
4. Desenvolvimento da interface de usuário errada	Análise das tarefas; prototipação; cenários; caracterização do usuário (funcionalidade, estilo, carga de trabalho)
5. "Gold plating" (ir além do esforço útil)	Enxugamento de requisitos; prototipação; análise do custo-benefício; "*design to cost*" (desenhar a custo ótimo)
6. Onda contínua de requisições de mudanças	Limiar para muita quantidade em mudanças; ocultação de informação; no desenvolvimento incremental, deferir mudanças nos últimos incrementos
7. Perdas devido a componentes externos	Benchmarking; inspeções; checagem de referências; análise de compatibilidades
8. Perdas devido a tarefas externas	Checagem de referências; auditorias *"pre-award*"; contratos "*award-fee*"; design competitivo ou prototipação; construção em equipe
9. Perdas devido ao desempenho em tempo real	Simulação; benchmarking; modelagem; prototipação; instrumentação; configuração de cenários
10. Problemas com capacidades computacionais	Análise técnica; análise do custo-benefício; prototipação; checagem de referências

Fonte: Boehm (1988)

Boehm (1988) apresenta uma lista priorizada dos dez maiores itens de risco para software, mas também técnicas para enfrentá-los (tabela 3).

Boehm (1988) ao apresentar o modelo espiral de desenvolvimento de software, cita que a natureza orientada a riscos do modelo é mais adaptável à grande gama de situações em projetos de software do que as outras abordagens tradicionais e mais orientadas a documentação, como o modelo cascata, portanto sendo mais adequado a sistemas muito grandes, complexos e ambiciosos. Também complementa ao dizer que implementações parciais do modelo espiral, como o plano de gerenciamento de riscos, são compatíveis com a maioria dos modelos atuais (como o CMMI) e são muito úteis para se superar maiores fontes de risco para o projeto.

Highsmith e Cockburn (2001), partindo da afirmação de Boehm sobre o custo da mudança crescer a cada fase do ciclo de vida de desenvolvimento de software, dizem que a questão atual não é como parar as mudanças ainda cedo no projeto, mas sim em como melhor lidar com mudanças inevitáveis durante todo o ciclo de vida.

De acordo com Highsmith e Cockburn (2001), abordagens tradicionais almejam que, apenas tentando com esforço o suficiente, pode-se antecipar o conjunto completo de requisitos, assim reduzindo os custos por eliminar mudanças nos projetos. Através de medições contínuas, identificação de erros e refinamentos dos processos, estas metodologias buscam remover variações dos processos, assumindo que variações são resultados de erros. Todavia, ainda que problemas nos processos causem alguns erros, mudanças ambientais externas também causam variações críticas. Eliminar mudanças antecipadamente significa ser não-responsivo a condições do negócio, levando a falhas deste.

Highsmith e Cockburn (2001) enfatizam que abordagens ágeis são mais aplicáveis a ambientes turbulentos e voláteis, tentando diminuir o custo de resposta a mudanças que não podem ser evitadas. Essas abordagens recomendam uma variedade de práticas para feedbacks constantes nas decisões técnicas, requisitos dos clientes e considerações dos gestores. Os autores veem organizações como sistemas complexos adaptativos, nos quais os requisitos são mais emergentes do que pré-especificáveis. Todavia, Boehm (2002) aponta que se os desenvolvedores utilizarem

mal a habilidade do ágil em aderir a requisitos mutáveis, pode-se obter resultados desastrosos.

Sobre métodos tradicionais, Boehm (2002) afirma que estes funcionam melhor quando desenvolvedores podem determinar os requisitos em andamento, inclusive via prototipagem, e quando os requisitos permanecem relativamente estáveis, com baixa taxa de mudanças. Já em situações onde os requisitos mudam numa taxa alta, a ênfase tradicional em possuir requisitos completos, consistentes, precisos, testáveis e rastreáveis encontra problemas nas atualizações de requisitos que não podem ser trespassados, ainda que esta ênfase seja vital para softwares críticos ou de segurança. Por outro lado, se a arquitetura se antecipar e acomodar futuras mudanças de requisitos, estes métodos podem manter mesmo o desenvolvimento em aplicações grandes dentro do custo e do prazo estipulados.

Boehm (2002), sobre arquitetura no caso de métodos ágeis, cita que a abordagem “você não vai precisar disso” da metodologia XP, baseada no princípio de simplicidade, funciona bem quando os requisitos futuros são realmente imprevisíveis, mas que em situações onde são previsíveis, ainda assim a preocupação com a arquitetura é deixada de lado e cria problemas com clientes que querem que os desenvolvedores se importem com a acomodação de futuras evoluções. Algumas outras metodologias ágeis se preocupam mais com planejamento para arquitetura.

Boehm (2002) cita que um conceito central na gestão de riscos envolve determinar a exposição ao risco (*RE*) dado um curso de ação. Para determinar a *RE*, deve-se avaliar a probabilidade de perda (*P(L)*) envolvida no curso de ação e o tamanho da perda correspondente (*S(L)*) e então computar a exposição ao risco como a perda esperada: $RE = P(L) \times S(L)$. Perda pode incluir lucros, reputação, qualidade de vida ou outros atributos relacionados a valor.

Ambas as abordagens das metodologias ágeis e tradicionais possuem um cenário (terreno) (tabela 4) de características de projeto nos quais claramente funcionam melhor e que a outra terá dificuldades (BOEHM, 2002). Abordagens híbridas que

combinam ambos os tipos são factíveis e necessárias para projetos que combinam um misto das características dos terrenos das ágeis e das tradicionais. A análise do risco das características do projeto *versus* as características do terreno de uma metodologia podem ajudar a determinar o melhor balanço entre disciplinas ágeis e tradicionais.

Tabela 4 - Terreno para os métodos ágeis e tradicionais.

Terreno	Métodos ágeis	Métodos tradicionais
Desenvolvedores	Ágeis, com conhecimentos, coalocados e colaborativos	Orientados a planejamento, habilidades adequadas, acesso a conhecimento externo
Clientes	Dedicados, com conhecimentos, coalocados, colaborativos, representativos e com poderes	Acesso a clientes com conhecimentos, colaborativos, representativos e com poderes
Requisitos	Largamente emergentes, mudanças rápidas	Conhecidos antecipadamente, largamente estáveis
Arquitetura	Desenhada para os requisitos atuais	Desenhada para requisitos atuais e previstos para o futuro
Refatoração	Barata	Cara
Tamanho	Pequenos times e produtos	Grandes times e produtos
Objetivo primário	Valoração rápida	Alta confiança

Fonte: Boehm (2002)

Na prática, quantificar estimativas para *P(L)* e *S(L)* é difícil, mas caso se deseje combinar métodos ágeis e orientados a planejamento, pode-se usar os atributos dos terrenos de ambos para determinar os riscos relativos. Boehm (2002) sugere algumas adaptações para ambos os métodos ágeis e tradicionais quando um dos atributos no qual o projeto está sendo aplicado não está de acordo com o terreno da metodologia, sempre com foco em reduzir os riscos presentes nos projetos.

Hoda, Noble e Marshall (2011) apresentam em seu trabalho o conceito de "*Agile Undercover*", que consiste em utilizar técnicas para contornar problemas que impeçam a correta aplicação de metodologias ágeis e que são normalmente utilizadas por equipes de desenvolvimento como medida temporária durante a transição para o ágil. Esta é uma medida bastante relacionada aos fatores que serão identificados nesse trabalho e que podem tornar a utilização de um tipo ou de outro

de metodologia mais adequado ao cenário em que o desenvolvimento está inserido, dado o que pode ser feito em tais situações.

Ries (2011) cita que a startup enxuta é adequada para condições de muita incerteza, onde utiliza de aprendizagem validada: processo de demonstrar empiricamente que uma equipe descobriu verdades valiosas acerca das perspectivas de negócio presentes e futuras. É mais concreta, exata e rápida do que prognósticos de mercado ou o clássico planejamento empresarial, buscando evitar “executar com sucesso um plano que não levará a bons resultados”. O contrário seria abrir uma nova empresa que seja clone exato de um negócio existente, onde o sucesso depende somente da execução, podendo ser modelado com grande exatidão: bom plano, entrega sólida e pesquisa de mercado completa, onde planejamento e previsão precisos são possíveis quando baseados num histórico operacional longo e estável e num ambiente relativamente estático.

Tanto o modelo *stage gates* (COOPER, 1990) quanto a *startup* enxuta (RIES, 2011) mostram que seus usos já são melhores para lidar com incertezas do que o não seguimento de nenhum modelo formal para o desenvolvimento de novos produtos, que pode levar à ilusão de que estão indo pelo caminho certo sem uma estratégia definida.

Criticidade

Silva et al (2011) lembram em seu estudo que a metodologia da família *Crystal* de métodos ágeis pode ser adaptada para desenvolver desde de sistemas simples e triviais a sistemas grandes e complexos. A escolha da adaptação depende do número de pessoas no projeto, criticidade do sistema/domínio e prioridade do projeto. Desta forma, criticidade é um fator apontado como relevante para a escolha entre metodologias de desenvolvimento software.

Nerur, Mahapatra e Mangalaraj (2005) citam que processos tradicionais são orientados a *compliance* e medidas para fornecer segurança, enquanto métodos ágeis baseiam-se em especular ou planejar com o entendimento de que tudo é incerto, direcionando ao rápido desenvolvimento de sistemas flexíveis e adaptativos

de alto valor e relevando a importância de avaliar ao invés de medir, também sendo altamente tolerantes a mudanças.

De acordo com Boehm (2002), métodos tradicionais são mais necessários para o desenvolvimento de softwares de alta segurança. Contudo, complementa que, particularmente no setor de *e-services*, companhias com uma base grande de clientes, não há necessidade apenas de valor rápido ou de alta segurança – precisam de ambos. Pura agilidade ou pura orientação a planejamento, sozinhas, não podem alcançar essas necessidades: uma mistura de ambas é necessária. O autor complementa que a gestão de riscos oferece uma abordagem que pode ajudar a equilibrar agilidade e disciplina.

Boehm e Turner (2004), ao identificar cinco fatores críticos relacionados aos tipos de metodologias e aos seus terrenos de origem, apontam a criticidade como um deles e citam que a agilidade não foi testada em produtos de segurança crítica, mas pode apresentar dificuldades em potencial devido ao design simples e à menor documentação. Já os métodos tradicionais evoluíram para lidar com produtos altamente críticos, porém isso os torna difíceis de customizar quando se trata do desenvolvimento de produtos com baixa criticidade.

2.2.3 Conclusão da literatura

Os fatores mais recorrentes na literatura sobre escolha e uso dos dois tipos de metodologia aqui levantados e discutidos servirão como objetos de estudo para buscar evidências empíricas sobre a adequação ou não de cada um deles às metodologias tradicionais e ágeis.

O fator sobre porte e estrutura organizacional não estará presente diretamente na parte do estudo que buscará mais evidências empíricas sobre os cenários mais adequados a cada tipo de metodologia, pois, como apontados na literatura, podem

ser favoráveis ou não dependendo muito mais de outros fatores decompostos e que são abordados a seguir.

Por exemplo, a adequação de cada uma das metodologias às organizações / times mais divididos por função ou mais divididos por projetos é um ponto bastante forte na literatura e neste estudo busca-se identificar a adequação de cada uma delas a cada uma das configurações.

Para o fator autonomia, envolvendo o grau desta permitido pela organização e as características relacionadas aos envolvidos nos times, busca-se identificar entre as metodologias ágeis e tradicionais de desenvolvimento de software, a adequação de cada uma delas aos cenários onde há maior autonomia e cenários onde há menor autonomia para a equipe de desenvolvimento.

Para o fator alinhamento a melhores práticas e normas, que envolve certificações para a companhia, gestão do conhecimento, definição de arquitetura, otimização, adaptabilidade e outros aspectos aqui apresentados, busca-se identificar a adequação de cada uma das metodologias aos cenários onde tal alinhamento é desejável.

Sobre os fatores mais relacionados aos projetos, em relação à duração do projeto, busca-se identificar entre os tipos de metodologias a adequação de cada uma delas aos cenários onde o projeto seja mais longo e onde o projeto seja mais curto.

Em relação à complexidade do projeto, de acordo com as características do produto de software gerado, busca-se identificar a adequação de cada uma delas aos cenários onde o projeto é mais complexo e onde o projeto é menos complexo.

Em relação à quantidade de pessoas envolvidas no projeto, busca-se identificar entre as metodologias ágeis e tradicionais de desenvolvimento de software, a adequação de cada uma delas aos cenários onde há times grandes ou pequenos de desenvolvimento de software.

Para os cenários onde há maior ou menor facilidade de comunicação ou proximidade entre os membros da equipe de desenvolvimento, principalmente abordado quando se trata de metodologias ágeis, busca-se identificar, entre este tipo e o das tradicionais, a adequação de cada um deles. O mesmo será feito para a adequação de cada uma delas aos cenários onde há maior e menor facilidade de comunicação ou proximidade com o cliente ou área solicitante.

Busca-se identificar também a adequação de cada um dos tipos de metodologia aos cenários onde há maiores e menores incertezas e riscos no projeto. E, por fim, a adequação de cada um deles aos cenários onde há maior e menor criticidade no projeto.

3 Metodologia

Levantados os fatores mais discutidos na literatura sobre adequação de metodologias tradicionais e ágeis aos cenários de aplicação destas, têm-se os seguintes pontos para avaliação:

- Adequação das metodologias para projetos longos;
- Adequação das metodologias para projetos curtos;
- Adequação das metodologias para projetos muito complexos;
- Adequação das metodologias para projetos pouco complexos;
- Adequação das metodologias para projetos com muitos envolvidos no desenvolvimento;

- Adequação das metodologias para projetos com poucos envolvidos no desenvolvimento;
- Adequação das metodologias para projetos com membros da equipe alocados próximos ou com comunicação facilitada;
- Adequação das metodologias para projetos com membros da equipe alocados distantes ou com comunicação restrita;
- Adequação das metodologias para projetos com equipes e clientes ou área solicitante alocados próximos ou com comunicação facilitada;
- Adequação das metodologias para projetos com equipes e clientes ou área solicitante alocados distantes ou com comunicação restrita;
- Adequação das metodologias para projetos com incertezas e riscos altos;
- Adequação das metodologias para projetos com incertezas e riscos baixos;
- Adequação das metodologias para projetos com alta criticidade;
- Adequação das metodologias para projetos com baixa criticidade;
- Adequação das metodologias em estruturas organizacionais mais divididas por função;
- Adequação das metodologias em estruturas organizacionais mais divididas por projetos;
- Adequação das metodologias em organizações que permitam maior grau de autonomia para as equipes de desenvolvimento;
- Adequação das metodologias em organizações que permitam menor grau de autonomia para as equipes de desenvolvimento;
- Adequação das metodologias em organizações que buscam seguir melhores práticas (como CMMI e ITIL).

Parte destes pontos já possui direcionamento pela literatura sobre suas adequações, mas nem sempre considerando ambos os tipos de metodologia para comparação e alguns ainda apresentam necessidade de evidências empíricas para serem confirmados. Estes pontos serão avaliados através da aplicação da metodologia deste trabalho, que busca identificar quais fatores devem ser considerados para a escolha do tipo de metodologia a ser utilizado.

Para complementar o modelo de escolha entre um dos dois tipos de metodologias de desenvolvimento de software com evidências empíricas, foi realizado um estudo de caso baseado em questionários na área de tecnologia de uma empresa de grande porte da área financeira, que possui projetos que utilizam metodologias tradicionais e também metodologias ágeis.

A empresa do estudo possui ao todo 85 000 funcionários nas mais variadas funções, onde logo abaixo da presidência existem diversas diretorias e, entre elas, a de tecnologia. A área de tecnologia possui aproximadamente 6 500 funcionários, sendo dividida ainda em várias subdivisões de acordo com tecnologias utilizadas, sistemas específicos e outras funções específicas desta área. Em uma dessas divisões, existe uma equipe que define os processos de metodologias de desenvolvimento, composta por aproximadamente 15 pessoas. De forma geral, as subdivisões da empresa costumam sofrer alterações frequentemente, principalmente hierárquicas, e esse era o cenário durante o estudo.

A empresa escolhida possui um modelo interno para a escolha do tipo de metodologia a ser aplicado que veio evoluindo ao longo do tempo e do uso das mesmas. Inicialmente, o modelo era baseado principalmente na escolha a partir de alguns fatores a serem observados em cada projeto: quantidade de pessoas na equipe (até 10 pessoas era direcionado o uso de ágil), tamanho do projeto (ágil era recomendado para projetos pequenos e curtos) e outros fatores internos relacionados à conformidade com processos de governança da empresa. Com o passar do tempo, maior adoção e maior maturidade com o uso das metodologias ágeis, além treinamentos para tal, estes fatores iniciais de escolha foram ficando em segundo plano e as equipes que desejavam trabalhar desta forma buscavam se adequar mais ao uso do ágil, montando equipes dedicadas e coalocadas aos projetos ágeis e trazendo os clientes para o ambiente de desenvolvimento, além de dar autonomia até onde era possível para estes grupos de trabalho. Em paralelo, os processos para os projetos que utilizam metodologias tradicionais passaram por mudanças de adequação a ferramentas que permitem maior controle, gestão e integração de informações, buscando assim atingir maior maturidade em modelos

como o CMMI, embora essas mudanças também tenham afetado como é feito o uso do ágil para a organização como um todo. Por fim, atualmente, existe uma proposta de algumas áreas trabalharem completamente de forma ágil, não apenas por projeto, e se integrarem às outras áreas que trabalham ou não desta forma através de processos sendo definidos de acordo com toda a experiência adquirida ao longo dos anos, melhores práticas de mercado e estudos.

Além disso, apresenta dos mais diversos cenários de desenvolvimento internamente, possuindo projetos com times de diferentes tamanhos alocados, alguns divididos em estruturais funcionais e outros em times montados para projetos específicos, desenvolvimento de softwares e sistemas nas mais variadas tecnologias, linguagens e plataformas, sendo que estes também apresentam diferentes níveis de criticidade, complexidade, riscos e prazos. Por se tratar de empresa da área financeira, os projetos da área buscam se adequar a diversas normas, leis e padrões. Assim, todos os fatores levantados na literatura a serem analisados estão contemplados neste ambiente.

Por fim, esta empresa foi escolhida devido à possibilidade de acesso para que se realizassem entrevistas e questionários com o intuito de obtenção de evidências que atendessem às necessidades deste estudo.

A equipe citada da área de tecnologia da empresa é responsável pela definição, divulgação, suporte e treinamento para uso das metodologias, sendo que o direcionamento é feito para o uso da metodologia tradicional ou ágil customizadas que possuem: a primeira baseada principalmente no modelo cascata e *stage gates* e a segunda principalmente no *Scrum* e TDD; ambas influenciadas pelo CMMI. Esta área recebe *feedback* constante das equipes e pessoas que utilizam as metodologias, além de acompanhar de perto alguns projetos. Assim, conseguem propor melhorias de processos frequentemente, bem como direcionar melhor para usar cada tipo de metodologia e integrar os diversos projetos que acontecem concomitantemente. Para este estudo, as pessoas dessa área foram contatadas e o

questionário na tabela 5 foi aplicado num primeiro momento para validação sobre os fatores tratados neste trabalho. Esta equipe foi dividida em duas partes nesse estudo, tendo participado uma parte dos membros nesse momento e outra parte mais adiante. Além da validação, também foi respondido o questionário nesse momento, como mostra a tabela 6.

No questionário, para medir a importância dos fatores, uma escala Likert de cinco pontos foi utilizada. Para cada um dos tópicos destacados há a escolha entre "Concordo totalmente" e "Discordo totalmente", relacionando-os tanto às metodologias tradicionais quanto às metodologias ágeis, como descrito na tabela 5. Uma possível resposta "não sei opinar" pode ser dada separadamente da escala de cinco pontos para cada item, excluindo apenas este tópico da contabilização total, caso o respondente não pudesse avaliar o assunto em questão.

Tabela 5 - Tópicos abordados no questionário.

Tópicos	Respostas
Adequação das metodologias de desenvolvimento de software... ... Para projetos longos. ... Para projetos curtos. ... Para projetos muito complexos. ... Para projetos pouco complexos. ... Para projetos com muitos envolvidos no desenvolvimento. ... Para projetos com poucos envolvidos no desenvolvimento. ... Para projetos com membros das equipes alocados próximos ou com comunicação facilitada. ... Para projetos com equipes e clientes ou área solicitante distantes ou com comunicação restrita. ... Para projetos com membros das equipes alocados próximos ou com comunicação facilitada.	Concordo totalmente - valor 5. Concordo parcialmente - valor 4. Não concordo e nem discordo - valor 3. Discordo parcialmente - valor 2. Discordo totalmente - valor 1. Não sei opinar (valor nulo).

... Para projetos com equipes e clientes ou área solicitante distantes ou com comunicação restrita. ... Para projetos com incertezas e riscos altos. ... Para projetos com incertezas e riscos baixos. ... Para projetos com alta criticidade. ... Para projetos com baixa criticidade. ... Em estruturas organizacionais mais divididas por função. ... Em estruturas organizacionais mais divididas por projetos. ... Em organizações que permitam maior grau de autonomia para as equipes de desenvolvimento. ... Em organizações que permitam menor grau de autonomia para as equipes de desenvolvimento. ... Em organizações que buscam seguir melhores práticas. Ex: CMMI, ITIL. Utilizando... ... Metodologias tradicionais ... Metodologias ágeis	

Fonte: autor

Além disso, buscando abrangência maior, para se obter o ponto de vista de usuários dos processos de metodologias de desenvolvimento de software, o mesmo questionário foi aplicado para membros das equipes de desenvolvimento da empresa, como unidades de análise além da equipe de processos. Um *web survey* com estas questões foi direcionado para 102 pessoas de diferentes áreas, sendo que 25 responderam e, destes, 19 questionários estavam totalmente completos. Além daqueles que não chegaram a atender o pedido, uma parte das pessoas não se sentia preparada para avaliar os fatores, assim foram coletadas apenas as respostas dos questionários completos, mas que, por si só, não traziam resultados

robustos, todavia foram levados para discussão na entrevista de retorno com a equipe de processos das metodologias (tabela 5).

Por fim, num terceiro momento, as respostas dos membros das equipes foram apresentadas em forma de gráficos para a segunda parte da equipe de processos, que buscou analisar e discutir o comportamento dos dados, reafirmando alguns dos fatores destacados no primeiro momento e interpretando outros de acordo com a forma como as metodologias são utilizadas na empresa e o histórico recente de uso pelos times de desenvolvimento. Foi possível comparar também as percepções de pessoas envolvidas com desenvolvimento, que utilizam as metodologias em suas atividades de desenvolvimento, e as percepções dos especialistas no assunto sobre o uso destas. Tanto no primeiro momento quando neste, cada fator foi trazido para discussão, o que será apresentado na próxima seção.

A abordagem aqui utilizada busca ter explorado cada fator levantado e, também, identificar a influência que cada um deles deve ter na escolha de um tipo ou de outro de metodologia de desenvolvimento de software. Tal avaliação permitirá descobrir quais fatores são críticos, devendo existir ou não para cada tipo de metodologia (habilitadores e inibidores), quais são indiferentes e quais podem ser contornados em caso de sua presença ou ausência.

4 Resultados e discussão

A tabela 6 apresenta, de forma resumida: valores obtidos na primeira entrevista com a equipe de metodologias; médias das respostas obtidas pelos questionários respondidos por desenvolvedores (apenas para visualização nesta tabela); e valores obtidos na última entrevista com a equipe de metodologias, que tinha acesso aos outros dois. Será possível obter direcionamentos a partir dos resultados de fatores que sejam mais adequados ou menos adequados para cada tipo de metodologia.

Durante a última entrevista, foi possível notar que alguns fatores claramente foram percebidos pelos membros das equipes da mesma forma que foram para a equipe de metodologias. Contudo, alguns pontos não foram percebidos da mesma forma.

Tabela 6 - Resultados dos questionários.

	1ª entrevista		Questionários		2ª entrevista	
Fatores	**Tradicional**	**Ágil**	**Tradicional**	**Ágil**	**Tradicional**	**Ágil**
Projetos longos	2	5	3,57	3,69	3	5
Projetos curtos	4	5	3,17	4,22	4	5
Projetos muito complexos	1	5	3,30	3,63	2	5
Projetos pouco complexos	5	4	3,43	3,69	5	2
Projetos com muitos envolvidos no desenvolvimento	5	5	3,39	3,63	2	5
Projetos com poucos envolvidos no desenvolvimento	1	5	3,48	3,91	5	5
Projetos com membros da equipe alocados próximos ou com comunicação facilitada	1	5	3,52	4,56	4	5
Projetos com membros da equipe alocados distantes ou com comunicação restrita	5	1	3,74	2,69	3	1
Projetos com equipes e clientes alocados próximos ou com comunicação facilitada	1	5	3,61	4,60	4	5
Projetos com equipes e clientes alocados distantes ou com comunicação restrita	5	1	3,60	3,69	3	1
Projetos com incertezas e riscos altos	1	5	3,60	3,56	1	5
Projetos com incertezas e riscos baixos	5	4	3,47	3,95	5	3
Projetos com alta criticidade	3	3	4,09	3,81	3	3
Projetos com baixa criticidade	3	3	3,50	4,50	3	3
Estruturas organizacionais mais divididas por função	5	1	4,11	3,57	4	1
Estruturas organizacionais mais divididas por projeto	4	4	3,72	4,26	4	5
Organizações que permitam maior grau de autonomia para as equipes	1	5	2,94	4,68	4	5
Organizações que permitam menor grau de autonomia para as equipes	5	1	4,15	2,66	4	1
Organizações que buscam seguir melhores práticas. Ex: CMMI, ITIL	3	4	3,88	4,00	5	5

Fonte: autor

Pelas experiências acumuladas dos membros da equipe de metodologias, os outros respondentes apenas precisavam de maior tempo de uso e de terem passado por outros cenários para terem a mesma percepção, então voltaram a reafirmar os pontos na segunda aplicação do questionário. Também foi esclarecido que não necessariamente por um fator ser muito adequado para um tipo de metodologia seria inadequado para o outro ou vice-versa, o que fez com que alguns valores mudassem da primeira entrevista para a última (“projetos com membros da equipe alocados próximos ou com comunicação facilitada” e “projetos com equipes e clientes alocados próximos ou com comunicação facilitada”).

Comparando os resultados da primeira entrevista e da segunda, assumindo valores 4 ou maior para fatores habilitadores, 2 ou menor para fatores inibidores, e destacando aqueles que se sobressaem para decisão e comparação entre os tipos de metodologia (onde um fator seja habilitador ou inibidor para um tipo de metodologia e não para o outro), além de haver consenso entre a primeira parte do grupo de especialistas e a segunda, tem-se:

Metodologias ágeis são adequadas para projetos longos

Acredita-se que metodologias ágeis sejam mais adequadas para projetos longos, pois quanto maior a duração é maior a chance de que haja alguma mudança de escopo no projeto. As metodologias ágeis são mais preparadas para lidar com mudanças do que as tradicionais. Todavia, metodologias tradicionais não apareceram como inadequadas para projetos longos, pois a duração do projeto em si não é o fator determinante pelas discussões, mas sim se ocorrerá alguma mudança no decorrer do desenvolvimento. No fator sobre riscos e incertezas essa questão fica mais evidente.

Metodologias ágeis são adequadas para projetos muito complexos

Metodologias tradicionais não são adequadas para projetos muito complexos

Sobre complexidade, metodologias tradicionais não aparecem como adequadas para projetos muito complexos e, além disso, metodologias ágeis aparecem como adequadas. Se um projeto é muito complexo, é necessário o alinhamento entre diversas pessoas e uma boa comunicação entre estas durante o desenvolvimento para que as partes estejam de acordo umas às outras em vários momentos. Além disso o levantamento de todos os requisitos já no início é, de fato, complexo para a boa utilização de metodologias tradicionais. Já para projetos pouco complexos, talvez a necessidade de seguir todas as cerimônias das metodologias ágeis seja desnecessária, mas ainda assim não houve consenso se chega a ser, de fato, inadequado.

Metodologias ágeis não são adequadas para projetos com membros da equipe distantes ou com comunicação restrita

Metodologias ágeis não são adequadas para projetos com equipes e clientes ou área solicitante distantes ou com comunicação restrita

Metodologias ágeis dependem muito da comunicação entre todos os envolvidos no projeto tanto em tempo de planejamento, quanto de execução e durante todo o ciclo de desenvolvimento. Isso ocorre porque a comunicação deixa de ser documentada e passa a ser em tempo real: interrupções ou falta de comunicação entre os membros da equipe e entre estes e os clientes podem prejudicar o fluxo, a qualidade e o prazo do projeto.

Metodologias ágeis são adequadas para projetos com incertezas e riscos altos

Metodologias tradicionais não são adequadas para projetos com incertezas e riscos altos

Uma das maiores propostas das metodologias ágeis é claramente atingida: saber lidar com mudanças ao decorrer do projeto. Este é também um dos maiores defeitos das metodologias tradicionais, que buscam seguir especificamente o que foi definido inicialmente no planejamento do projeto. Assim sendo, quanto maiores forem as incertezas e riscos dos projetos, mais adequado é seguir um modelo mais iterativo e menos linear ou mais ágil, de fato, e menos preso a processos e documentos pesados. Para isso é necessário que os processos sejam leves e preparados para qualquer mudança que possa ocorrer.

Metodologias ágeis não são adequadas em estruturas organizacionais mais divididas por função

Metodologias tradicionais são adequadas em estruturas organizacionais mais divididas por função

Metodologias ágeis dependem da coalocação dos membros da equipe de desenvolvimento ou da comunicação direta e facilitada entre eles para poderem utilizar de modelos iterativos de desenvolvimento. Estruturas mais divididas por função dificilmente permitem esse tipo de interação e os processos de trabalho mais

lineares das metodologias tradicionais e a comunicação documentada são mais adequados nesses casos.

Metodologias ágeis não são adequadas em organizações que permitam menor grau de autonomia para as equipes de desenvolvimento

Metodologias tradicionais são adequadas em organizações que permitam menor grau de autonomia para as equipes de desenvolvimento

Por fim, metodologias ágeis dependem de autonomia para os grupos de trabalho para que estes possam rapidamente tomar decisões e seguir com o desenvolvimento nas interações, onde são feitas definições, bem como quando se deparam com mudanças que precisam acontecer sobre aquilo que já havia sido definido. Todavia, para se obter autonomia é necessária boa qualificação dos membros da equipe. Metodologias tradicionais estariam melhor preparadas para quando não há ou há pouca autonomia para a equipe de desenvolvimento.

Sobre os fatores que foram apontados como adequados para ambos os tipos de metodologia, é relevante para saber que nenhum dos tipos vai contra o fator, como adequação a melhores práticas e processos. Há também outros fatores que não são nem muito adequados e nem inadequados para ambos os tipos de metodologia, como os relacionados a criticidade. Assim sendo, não precisam fazer parte de um modelo para escolha entre um ou outro tipo de metodologia.

5 Análise e conclusões

Com os resultados obtidos através das entrevistas e questionários deste trabalho, é possível ver o alinhamento do que foi encontrado com o que foi antes levantado da literatura sobre o tema e concluir quais fatores relacionados à organização e às

pessoas e aos projetos são importantes para a escolha entre metodologias ágeis e metodologias tradicionais.

Sobre as estruturas organizacionais mais divididas por função, na revisão da literatura, quando se fala sobre os modelos propostos por Wheelwright e Clark (1992), o modelo funcional é o que apresenta menor autonomia para o grupo de trabalho, com o gestor funcional concentrando as responsabilidades e a tomada de decisão, além de o grupo seguir mais o que é definido para a organização como regras gerais. O modelo autônomo é o que apresenta o maior grau de autonomia para o grupo e, inclusive, propõe alocar os seus membros juntos durante o processo de desenvolvimento. Esses dois modelos estão muito de acordo com o que foi encontrado no resultado deste trabalho, onde metodologias ágeis não são adequadas para estruturas mais divididas por função (fator inibidor) e as tradicionais são (fator habilitador), além de apresentarem indicações sobre outros fatores também discutidos a seguir.

Sobre a proximidade entre os membros da equipe, Cockburn e Highsmith (2001) citam sobre comunicação que uma ideia dominante no desenvolvimento ágil é a de que a equipe pode ser mais efetiva em responder a mudanças se puder reduzir o custo da movimentação de informação entre as pessoas e se puder reduzir o tempo gasto entre a tomada de decisão e ver as consequências das decisões. Os autores complementam que, para reduzir o custo de movimentação de informação entre pessoas, o time ágil busca colocar as pessoas fisicamente próximas, substituir documentos por conversas entre pessoas ou uso de quadros e aumentar a amigabilidade do time para poderem compartilhar informações de valor mais rapidamente. Está de acordo com o resultado que aponta que metodologias ágeis não são adequadas quando não há facilidade de comunicação entre os membros da equipe.

Sobre a proximidade entre os membros da equipe e os clientes, Highsmith e Cockburn (2001) complementam que um time não é ágil se o loop de feedback dos clientes apresentar muito tempo entre suas repetições. As abordagens ágeis recomendam iterações curtas, entre duas a seis semanas, durante as quais a equipe

faz constantes decisões e ajustam novas informações com os clientes, como o planejamento das funcionalidades e priorização dinâmica (o cliente pode repriorizar as funcionalidades originalmente planejadas, bem como descartar ou adicionar novas). Um ponto-chave nas abordagens ágeis é que estas planejam funcionalidades, que é o que o cliente entende bem, e não tarefas. Também está de acordo com o resultado que aponta que metodologias ágeis não são adequadas quando não há facilidade de comunicação entre os membros da equipe. Hoda, Noble e Marshall (2011) ainda complementam que os clientes precisam perceber suas responsabilidades para garantir o sucesso de projetos ágeis autogeridos e, por sua vez, o sucesso de seus próprios projetos.

Um ponto forte sobre complexidade apontado na literatura é de Cockburn e Highsmith (2001), que citam que a gestão tradicional de projetos sofre do que é chamado de uma “neurose newtoniana”, ou seja, tenta atacar problemas complexos e não-lineares com processos simples e lineares. Os autores ainda complementam que o desenvolvimento ágil tem excelência em problemas exploratórios, que são os casos de projetos em situações extremas, complexas ou com alto índice de mudanças, portanto melhor atacados quando em uma cultura organizacional colaborativa e centrada em pessoas. Isso vai de acordo com os resultados que indicam que metodologias ágeis são adequadas a projetos mais complexos e metodologias tradicionais não são. Contudo, para projetos de baixa complexidade, não há consenso se metodologias ágeis chegam a ser inadequadas, devido aos procedimentos que apresentam e que podem ser desnecessários nesses casos.

Em alguns trechos aqui apontados e na seção específica que trata sobre riscos e incertezas, o direcionamento da literatura era da preferência por metodologias ágeis onde estes são grandes, devido à proposta de adotar bem mudanças de escopo, e da inadequação das metodologias tradicionais para tal, devido a proposta de seguir mais firmemente planos já definidos. Isso novamente foi apontado pelos resultados desta pesquisa e como um dos fatores mais decisivos para a escolha.

Sobre autonomia, os resultados apontam que metodologias ágeis não são adequadas onde há pouca autonomia para as equipes e que metodologias

tradicionais são. Isso complementa também o que a literatura aponta sobre autonomia e times autogeridos, que precisam manter um foco em comum, confiança mútua, respeito e a habilidade de se organizar repetidamente para enfrentar novos desafios (COCKBURN; HIGHSMITH, 2001). Lembrando que times autogeridos não são times sem controle ou sem líderes (COCKBURN; HIGHSMITH, 2001; TAKEUCHI; NONAKA, 1986), mas sim com menos rigor e adaptáveis, fornecendo feedback e direcionamento mais sutil (TAKEUCHI; NONAKA, 1986). Ainda sobre o tema, Boehm (2002) cita que o replanejamento é simples quando ocorre num time com ótimos desenvolvedores desenvolvendo pequenos sistemas. Contudo, evidências empíricas encontradas pelo autor apontam que, com desenvolvedores não tão ótimos assim, os esforços para replanejamento aumentam com o número de requisitos ou estórias envolvidas no projeto. Assim, quando se trata de autonomia é importante também avaliar as capacidades das pessoas envolvidas.

Sobre duração do projeto, havia pontos a favor e contra cada tipo de metodologia, mas não havia evidências empíricas conclusivas no levantamento. Neste trabalho, o direcionamento foi que metodologias ágeis são adequadas a projetos longos, embora não haja apontamento de que metodologias tradicionais sejam inadequadas.

Os fatores até agora apontados são aqueles que, nos resultados, apareceram como fatores de escolha para um tipo ou outro de metodologia, porém outros fatores apontados pela literatura não apareceram como decisivos:

- Alinhamento a normas e boas práticas, pois para ambos os tipos de metodologia existem boas práticas, frameworks e certificações, o que faz com que o uso de qualquer tipo possa ser adequado a este fator;
- Tamanho da equipe também não aparece como um fator decisivo, porém não houve consenso, em conversa com os especialistas, foi entendido que pode influenciar em outros tidos como decisivos, como a proximidade ou facilidade de comunicação entre os membros da equipe;
- Criticidade era um fator sobre o qual faltava evidência empírica e se acreditava que metodologias tradicionais eram mais adequadas por terem evoluído para se adaptar a ela, porém, os resultados apontam que metodologias ágeis também servem para lidar com criticidade. Além disso, nas entrevistas, entende-se

criticidade como algo que pode ser interpretado de diferentes maneiras, além de que a metodologia em si não seja algo que tratará ou não de forma mais adequada o fator.

Assim sendo, o modelo para escolha entre metodologias ágeis ou metodologias tradicionais deve levar em consideração os seguintes fatores:

Sobre projetos:

- Para projetos muito complexos, ágil é adequado e tradicional é inadequado;
- Para Incertezas e riscos altos: ágil é adequado e tradicional é inadequado;
- Para projetos longos: ágil é adequado e tradicional depende de outros fatores.

Sobre formatação das equipes:

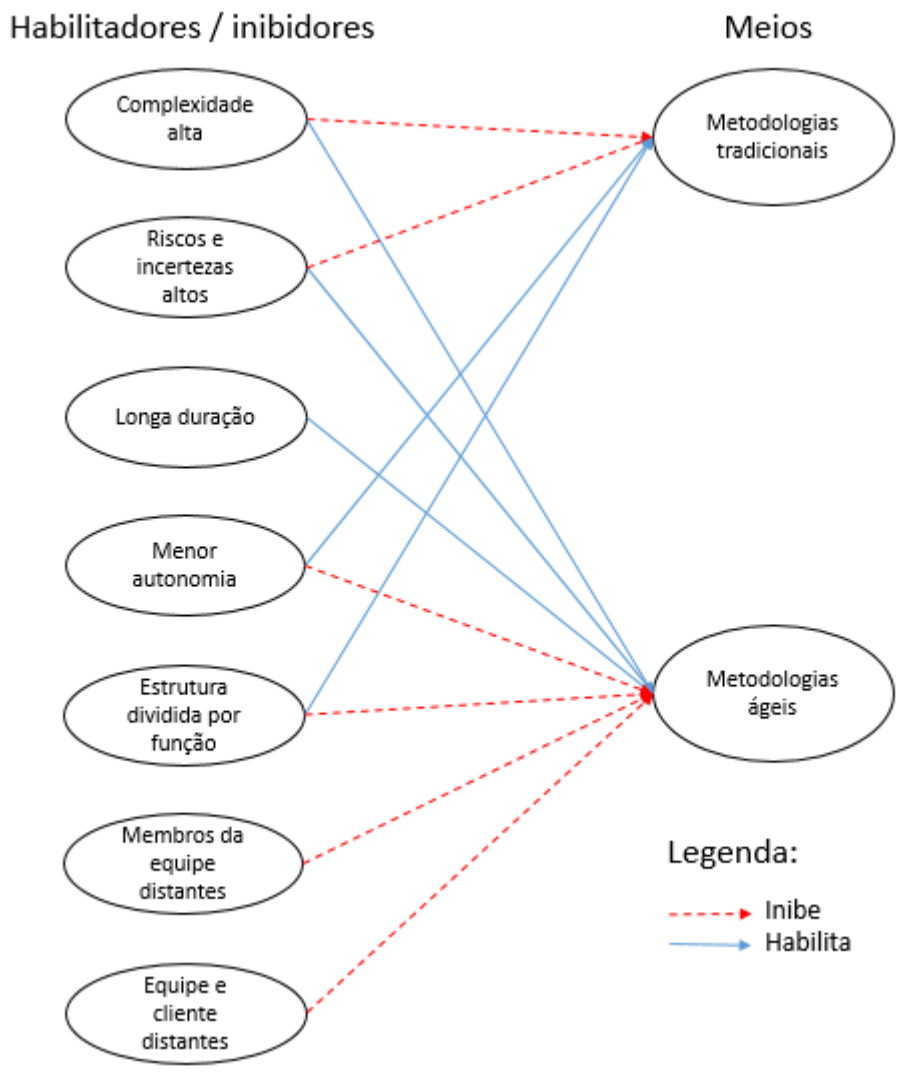
- Em estruturas mais divididas por função: tradicional é adequado e ágil é inadequado;
- Menor autonomia: tradicional é adequado e ágil é inadequado;
- Além disso ágil precisa de membros da equipe próximos ou com comunicação facilidade entre eles e de membros da equipe e clientes próximos ou com comunicação facilitada.

Por fim, Nerur, Mahapatra e Mangalaraj (2005) citam que o desenvolvimento de software é uma atividade complexa, caracterizada por tarefas e requisitos que apresentam um alto grau de variabilidade (BOEHM; TURNER 2004; HIGHSMITH 2003). Incertezas são agravadas pela diversidade e imprevisibilidade das pessoas que se engajam em tais tarefas (COCKBURN; HIGHSMITH, 2001; COCKBURN 2002). A natureza de mudança e sofisticação de ferramentas também pode aumentar os problemas causados por tais fenômenos. Kettunen e Laanti (2008) também citam os turbulentos ambientes do mercado global, onde estão inseridas as companhias desenvolvedoras de software, como fontes de muitas das incertezas (MULLINS; SUTHERLAND, 1998 apud KETTUNEN; LAANTI, 2008). Neste cenário, segundo os autores, a agilidade fornece uma vantagem competitiva para as organizações. O que também vai de acordo com o que foi encontrado nesta

pesquisa sobre a intenção da empresa em ter equipes trabalhando inteiramente de forma ágil, e não para cada projeto.

Portanto, em um modelo para escolha entre metodologias ágeis ou tradicionais, os fatores para escolha estariam dispostos como na figura 8, respondendo a questão dessa pesquisa.

Figura 8 – Modelo para escolha entre metodologias ágeis e metodologias tradicionais.



Fonte: autor

Abrahamsson (2000) também reforça em seu trabalho que é importante considerar outras dimensões de sucesso além da avaliação pelo usuário do processo, pois embora esta seja considerada a mais significativa para a melhoria do processo, as

outras são também, pelo menos, relevantes. Este trabalho, dada a metodologia utilizada por ele, tem seu foco limitado à essa principal dimensão, o que representa um avanço para as evidências empíricas no domínio. Todavia, para medir a efetividade total dos processos e metodologias adotadas, o modelo ainda precisa ser mais incrementado, indo além do escopo aqui abordado em estudos futuros.

Também é limitação do escopo deste trabalho tratar dos casos onde pode ser escolhido um ou outro tipo de metodologia, sem propor a integração entre ambos, ou tratar de momentos de transição para o uso de algum tipo de metodologia, embora possa servir para identificar o que é necessário para habilitá-las.

Por fim, o trabalho baseia-se em um estudo de caso único, ainda que este atendesse a necessidade de ter todos os pontos avaliados presentes na organização e em seus projetos de desenvolvimento de software. Também se sugere que, em próximos estudos, o questionário com fatores de escolha seja respondido por uma quantidade maior de pessoas envolvidas de maneira que possa haver tratamento estatístico, incluindo-se pesos aos fatores do modelo resultado.

Referências

ABRAHAMSSON, P. Measuring the success of software process improvement: the dimensions. **EUROSPI'00**, Copenhague, Dinamarca, 2000.

AGILE ALLIANCE, **Agile alliance: The Agile Manifesto**, 2013. Disponível em: <http://www.agilealliance.org/the-alliance/the-agile-manifesto>. Acesso em: 16 Mar. 2015.

BECK, K. **Extreme Programming Explained**. Addison Wesley, Reading,1999.

BOEHM, B. W. A Spiral Model of Software Development and Enhancement. **IEEE Computer**, v. 21, n.5, 61-72, 1988.

BOEHM, B. Get Ready for Agile Methods, with Care**. IEEE Computer**, v. 35, n.1, 64-69, 2002.

BOEHM, B.; TURNER, R. **Balancing Agility and Discipline: A Guide for the Perplexed**. Addison-Wesley Longman Publishing CO., Boston, 2004.

CAVALERI, S.; OBLOJ, K. **Management Systems: A Global Perspective**. Wadsworth Publishing Company, Belmont, 1993.

CHARETTE, R. N. Challenging the fundamental notions of software development. **Cutter Consortium Reports**, v. 4, n. 6, Arlington, 2003.

CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Patterns and Practice**. Addison-Wesley, 2001.

COAD, P.; PALMER, S. **Feature-Driven Development**. Prentice Hall, Englewood Cliffs, 2002.

COCKBURN, A.; HIGHSMITH, J. Agile Software Development: The People Factor. **IEEE Computer**, v. 34, n.11, 131-133, 2001.

COCKBURN, A. **Crystal Clear: A Human-Powered Software Development Methodology for Small Teams**. Addison-Wesley, Reading, 2001.

COCKBURN, A. **Agile Software Development**. Addison-Wesley, Boston, 2002.

DINGSØYR, T.; DYBÅ, T.; MOE, N. B. A teamwork model for understanding an agile team: A case study of a Scrum project. **Information and Software Technology**, v. 52, 480–491, 2010.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and Software Technology**, v. 50, 833–859, 2008.

DYBÅ, T.; DINGSØYR, T. What Do We Know about Agile Software Development?. **IEEE Software**, v. 26, n. 5, 2009.

ERICKSON, J.; LYYTINEN, K.; SIAU, K. Agile modeling, Agile software development, and extreme programming: the state of research. **Journal of Database Management,** v. 16, n. 4, 88–100, 2005.

HIGHSMITH, J.; COCKBURN, A. Agile Software Development: The Business of Innovation. **IEEE Computer**, v. 34, n.9, 120-122, 2001.

HIGHSMITH, J. **Adaptive Software Development: A Collaborative Approach to Managing Complex Systems**. Dorset House Publishing, New York, 2000.

HIGHSMITH, J. **Agile Software Development Ecosystems**. Addison-Wesley, Boston, 2002.

HIGHSMITH, J. Agile Project Management: Principles and Tools. **Cutter Consortium Reports**, v. 4, n. 2, Arlington, 2003.

HILKKA, M. R.; MATTI, R.; TUURE, T. Is extreme programming just old wine in new bottles: a comparison of two cases. **Journal of Database Management**, v.16, n.4, 41-61, 2005.

HODA, R.; NOBLE, J.; MARSHALL, S. The impact of inadequate customer collaboration on self-organizing Agile teams. **Information and Software Technology**, v. 53, 521–534, 2011.

KETTUNEN, P.; LAANTI, M. Combining Agile Software Projects and Large-scale Organizational Agility. **Software Process Improvement and Practice**, v. 13, 183–193, 2008.

KRAFCIK, J. F. **Comparative analysis of performance indicators at world auto assembly plants**. jan, 1988. Dissertação – Massachusetts Institute of Technology.

LAANTI, M.; SALO, O.; ABRAHAMSSON, P. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. **Information and Software Technology**, v. 53, 276–290, 2011.

LARMAN, C.; BASILI, V. R. Iterative and Incremental Development: A Brief History. **IEEE Computer**, v. 36, n.6, 47-56, 2003.

MANNARO, K.; MELIS, M.; MARCHESI, M. Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software Development Methodologies. **Lecture Notes in Computer Science**, v. 3092, 166–174, 2004.

NERUR, S.; MAHAPATRA, R.; MANGALARAJ, G. Challenges of Migrating to Agile Methodologies. **Communications of the ACM**, v.38, n. 5, 73-78, 2005.

PALMER, S.; FELSING, M. **A Practical Guide to Feature-Driven Development**. Pearson Education, 2001.

PAULK, M. C. Extreme programming from a CMM perspective. **IEEE Software**, v. 18, n. 6, 2001.

PETERSEN, K.; WOHLIN, C. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. **The Journal of Systems and Software**, v. 82, 1479–1490, 2009.

POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development: An Agile Toolkit**. Addison-Wesley, 2003.

POPPENDIECK, M., POPPENDIECK, T. **Implementing Lean Software Development from Concept to Cash**. Addison Wesley Professional, 2006.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software: uma abordagem professional**. 8ª ed. AMGH, Porto Alegre, 2016.

RIES, E. **A startup enxuta**. Lua de Papel, São Paulo, 2012.

ROYCE, W. Managing the Development of Large Software Systems. **Proceedings Westcon**, IEEE CS Press, 328-339, 1970.

SCHWABER, K.; BEEDLE, A. **Agile Software Development with SCRUM**. Prentice-Hall, Upper Saddle River, 2002.

SCRUM ALLIANCE. Disponível em: <http://www.scrumalliance.org/>. Acesso em: 17 Mar. 2015.

SILVA, I. F.; NETO, P. A. M. S.; O'LEARY, P.; ALMEIDA, E. S.; MEIRA, S. R. L. Agile software product lines: a systematic mapping study. **Software - Practice and Experience**, v.41, 899–920, 2011.

STAPLETON, J. **Dynamic Systems Development Method**. Addison Wesley, 1997.

TAKEUCHI, H.; NONAKA, I. The New New Product Development Game. **Harvard Business Review**, 137-146, Jan-Fev, 1986.

WANG, X.; CONBOY, K.; CAWLEY, O. “Leagile” software development: An experience report analysis of the application of lean approaches in agile software development. **The Journal of Systems and Software**, v. 85, 1287–1299, 2012.

WHEELWRIGHT, S. C.; CLARK, K. B. **Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency and Quality**. Free Press, New York, 1992.

WILLIAMS, F., MONGE, P. **Reasoning with Statistics.** Thomson Wadsworth, Belmont, 2001.

5CQUALIBR. **Desenvolvimento Ágil de Software**. Disponível em: <http://www.softwarepublico.gov.br/5cqualibr/xowiki/Interoperabilidade-Semantica>. Acesso em: 12 nov. 2014.