Analisando a Agilidade em Processos Ágeis

Guilherme Schoepping[1], Patrícia Vilain[1]

[1]Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

88040-900, Florianópolis/SC, Brasil

schoepping@gmail.com, vilain@inf.ufsc.br

Abstract. *The main objective of agile methods is to promote efficient software development through practices that prioritize the communication with the client and frequent deliveries. Each agile method presents its unique set of practices. This diversity of practices may lead to the definition of new agile processes that include only the more appropriate practices from these methods. The problem, however, is that combining practices from different methods does not guarantee that the resulting process can be considered agile. This work presents some informal criteria for analyzing the agility of new processes that come up from existing agile practices.*

Resumo. *O principal objetivo dos métodos ágeis existentes é promover o desenvolvimento eficiente de software através de práticas que priorizam a comunicação com o cliente e entregas frequentes. Cada método ágil apresenta um conjunto próprio de práticas. Com esta diversidade de práticas torna-se interessante a construção de novos processos ágeis que contemplem apenas as práticas mais adequadas a partir destes métodos. O problema, entretanto, é que a combinação de práticas de diferentes métodos ágeis não garante, necessariamente, que o novo processo definido seja ágil. Este trabalho apresenta critérios informais para analisar a agilidade de novos processos que surgem com a seleção das práticas ágeis existentes.*

1. Introdução

Sistemas de informação exercem um importante papel no mundo moderno, e seu rápido desenvolvimento e correto funcionamento são uma grande vantagem de negócio. A metodologia ágil se apresenta como uma importante alternativa para o desenvolvimento de sistemas de informação.

O desenvolvimento ágil surgiu para transpor obstáculos que a engenharia de software convencional enfrenta: atrasos na entrega de software, dificuldade na mudança de requisitos e pouca comunicação entre o cliente e a equipe de desenvolvimento [2]. Em 2001, diversos autores da área de software declararam, através do Manifesto Ágil [5], estar propondo melhores modos de desenvolvimento de software onde passaram a valorizar: indivíduos e interações em vez de processos e ferramentas, softwares funcionando em vez de documentação abrangente, colaboração com o cliente em vez de negociação de contratos, e resposta a modificações em vez de seguir um plano. É importante destacar que a agilidade pode ser adicionada em qualquer processo [2] e traz diversos benefícios, e também não é contrária às sólidas práticas de engenharia de

software. Uma das principais características de um processo de desenvolvimento ágil é seu comportamento quanto ao acolhimento a mudanças [6].

Diversos métodos considerados ágeis podem ser encontrados: Extreme Programming (XP), Scrum, Feature Driven Development (FDD), Crystal, Adaptative Software Development (ASD), Dynamic System Development Method (DSDM), entre outros. Naturalmente, existe muita semelhança na abordagem e nas práticas destes diferentes métodos, ainda que hajam características que tornam cada um deles singular.

Diante da variedade de métodos ágeis e das semelhanças e diferenças entre eles, foi proposto um framework que auxilia na construção de um novo processo ágil a partir das práticas dos principais métodos existentes [1]. Este framework facilita o trabalho de qualquer organização que queira definir seu próprio processo de desenvolvimento ágil, através da seleção de práticas que os envolvidos julguem necessárias. Evita-se, com o uso deste framework, a necessidade de avaliar diferentes métodos ágeis e escolher apenas um deles ou um conjunto de práticas deles.

A definição de um novo processo ágil traz como principal qualidade a personalização, pois funcionará de acordo com a filosofia da organização ou projeto. Um novo problema, todavia, pode surgir. Mesmo que cada prática constituinte do novo processo faça parte de um método preexistente e seja considerada ágil, é possível que a combinação de algumas delas diminua o grau de agilidade do novo processo como um todo. É este o problema que motiva este trabalho, e a proposta é analisar a agilidade de novos processos definidos a partir do framework, considerando cada prática em conjunto e separadamente.

O restante do artigo está dividido em quatro seções. A seção 2 descreve o framework para a construção de novos processos ágeis. A seção 3 mostra a abordagem usada para analisar o nível de agilidade de novos processos construídos com o framework. Por fim, as conclusões são apresentadas na seção 4.

2. Framework para a construção de novos processos ágeis

O framework proposto em [1] consiste em um conjunto de práticas ágeis de desenvolvimento de software. Estas práticas foram reunidas a partir dos métodos ágeis atualmente mais relevantes: Extreme Programming (XP), Scrum, Feature Driven Development (FDD), Adaptive Software Development (ASD), Dynamic System Development (DSDM), Crystal Clear, Lean Software Development (LSD) e Agile Modeling (AM). No framework, o agrupamento destas práticas está baseado na lista de atividades do processo de desenvolvimento incremental [3].

As práticas selecionadas no framework também foram avaliadas comparativamente, identificando relações de dependência e exclusão entre elas, aqui não apresentadas. A tabela 1 mostra a lista de todas as práticas presentes no framework e os métodos dos quais elas foram extraídas. Uma breve descrição destas práticas, agrupadas por atividades, é apresentada a seguir; maiores informações podem ser encontradas em [1].

Tabela 1. Origem das práticas do framework.

Prática	XP	Scrum	FDD	ASD	DSDM	Crystal	LSD	AM
Lista de Requisitos		X	X		X	X	X	

Estórias do Usuário	X					X		
Casos de Uso			X					
Documentação Inicial								X
Planejamento da Iteração	X	X	X	X	X	X	X	
Duração das Iterações	X	X	X	X	X	X		
Distribuição dos Requisitos			X					
Projeto do Sistema	X	X	X		X	X	X	X
Desenvolvimento	X	X	X	X	X	X	X	
Projeto Detalhado do Sistema	X	X	X		X	X	X	X
Testes de Unidade	X							
Testes de Aceitação	X							
Desenvolvimento Coletivo de Código	X							
Programação em Pares	X							
Refatoração	X							
Reuniões Diárias	X	X				X		
Integração Paralela	X					X		
Controle de Versões					X			
Desenvolvimento Lado a Lado						X		
Integração do Incremento antes da Validação	X				X			
Execução dos Testes de Unidade	X		X	X	X	X	X	
Execução dos Testes de Aceitação	X	X		X	X	X	X	
Inspeção de Código			X	X				
Integração do Incremento Resultante		X	X					
Reunião de Revisão da Iteração		X	X				X	
Validação no Ambiente do Cliente	X	X						
Entrega do Sistema	X	X	X	X	X	X		
Documentação Breve	X	X						
Documentação Operacional						X		X

2.1. Atividade de Definição dos Requisitos

Lista de Requisitos: Consiste no documento contendo uma lista com os requisitos do sistema. Descrições breves sobre os requisitos e informações adicionais podem estar presentes.

Estórias do Usuário: Para cada funcionalidade do sistema, o usuário escreve uma estória do usuário, com o auxílio do desenvolvedor, para elucidar informações sobre a funcionalidade do sistema.

Casos de uso: As funcionalidades podem ser descritas através da especificação dos casos de uso e outros diagramas. Esta especificação é feita pelo desenvolvedor, responsável pela geração do artefato, e com o auxílio do cliente, conhecedor das informações de negócio.

Documentação inicial: Alguns documentos são criados para gerir as informações do projeto. Estes documentos podem incluir a análise de viabilidade, estimativa de custos, gestão de riscos, entre outros.

2.2. Atividade de Atribuição dos Requisitos às Iterações

Planejamento da Iteração: Aqui ocorre o planejamento da iteração, onde os requisitos que serão desenvolvidos na iteração corrente são escolhidos. Normalmente este planejamento ocorre no início das iterações e na forma de reunião.

Duração das Iterações: A equipe de desenvolvimento deve definir o período de tempo das iterações de acordo com as necessidades do projeto. Para não comprometer os princípios ágeis, recomendam-se iterações de uma a oito semanas.

Distribuição dos Requisitos: Os requisitos são atribuídos aos seus responsáveis pelo desenvolvimento.

2.3. Atividade de Projeto da Arquitetura do Sistema

Projeto do Sistema: O projeto do sistema é definido de acordo com os seus requisitos. Artefatos como diagramas de classe e de sequência podem ser gerados.

2.4. Atividade de Desenvolvimento do Incremento do Sistema

Desenvolvimento: Aqui está compreendida a codificação dos requisitos.

Projeto Detalhado do Sistema: A cada iteração, o projeto geral do sistema pode ser detalhado de acordo com os requisitos selecionados na iteração.

Testes de Unidade: Testes de unidade são desenvolvidos antes ou durante a codificação do sistema.

Testes de Aceitação: Testes de Aceitação são definidos para auxiliar o cliente durante a entrega do incremento.

Desenvolvimento Coletivo do Código: Todos os participantes da equipe têm conhecimento total do sistema e são responsáveis pelo seu desenvolvimento.

Programação em Pares: A codificação é feita em pares, o que diminui a dificuldade em transpor obstáculos e aumenta a qualidade e confiabilidade do código.

Refatoração: Prática para melhorar a estrutura interna do código sem alterar o seu comportamento.

Reuniões Diárias: Reuniões diárias para que os participantes informem seus progressos, tarefas e impedimentos. As reuniões devem ser curtas e recomenda-se que os participantes fiquem em pé.

Integração Paralela: O código produzido na iteração corrente é integrado simultaneamente ao código do incremento anterior.

Controle de Versões: As mudanças feitas no projeto são armazenadas para fins históricos e para situações em que seja necessário revertê-las.

Desenvolvimento Lado a Lado: Semelhante à prática de Programação em Pares, mas com um computador para cada desenvolvedor, permitindo o desenvolvimento paralelo.

2.5. Atividade de Validação do Incremento

Integração do Incremento antes da Validação: Integração do incremento antes da validação.

Execução dos Testes de Unidade: Execução dos testes de unidade desenvolvidos anteriormente.

Execução dos Testes de Aceitação: Execução dos testes de aceitação elaborados anteriormente.

Inspeção de Código: Inspeção de código para detectar defeitos e verificar compreensibilidade do código.

2.6. Atividade de Integração do Incremento

Integração do Incremento Resultante: O incremento é integrado no fim da iteração, após todas as validações e inspeções.

2.7. Atividade de Validação do Sistema

Reunião de Revisão da Iteração: Reunião onde os participantes avaliam os eventos ocorridos durante a iteração.

Validação no Ambiente do Cliente: O sistema é validado no ambiente de trabalho do cliente.

2.8. Atividade de Entrega Final

Entrega do Sistema: Entrega do sistema ao cliente se este estiver satisfeito e aquele puder ser utilizado.

Documentação Breve: Documentação breve sobre o desenvolvimento, se a equipe não gerou uma documentação detalhada durante o processo.

Documentação Operacional: A documentação gerada durante o processo pode ser expandida e novos documentos podem ser gerados, como manuais de referência e materiais de treinamento.

3. Análise do nível de agilidade do novo processo

Como citado anteriormente, o framework descrito na seção anterior permite que a equipe de desenvolvimento selecione as práticas mais adequadas ao seu ambiente e projeto. Esta seleção de práticas, entretanto, pode resultar em um processo pouco ágil. Para evitar esta situação, está sendo proposta neste trabalho uma análise quantitativa das práticas selecionadas que permita, ao fim da construção de um novo processo, avaliar o seu nível de agilidade. Para tanto, foi necessário avaliar todas as práticas do framework, de forma separada e conjunta, e determinar o impacto de cada uma delas na agilidade do processo. Ao fim desta avaliação, o framework será complementado com esta análise.

3.1. Grupos de Práticas

Pode-se perceber no conjunto de práticas do framework que algumas delas compartilham características semelhantes. Este aspecto nos permite agrupá-las de forma a facilitar a análise do impacto na agilidade que estas podem causar no processo.

3.1.1. Práticas Essenciais

As práticas consideradas essenciais estão presentes em qualquer processo de desenvolvimento por envolverem necessidades elementares dos métodos ágeis. Este grupo inclui as seguintes práticas:

Lista de Requisitos: A lista de requisitos, mesmo que não definitiva, é o ponto de partida para qualquer tipo de desenvolvimento de software.

Desenvolvimento: Esta prática é obrigatória e também a mais importante do processo. O tamanho do projeto é diretamente proporcional ao esforço despendido nesta prática, mas isto não afeta o processo de desenvolvimento em si.

Entrega do Sistema: O objetivo de todo projeto é entregar algo ao cliente. Consequentemente todo processo possui alguma prática semelhante, o que torna desnecessário considerar o impacto da entrega na agilidade do processo.

3.1.2. Práticas com Esforço Mínimo

Existem práticas que não demandam tempo ou que a demanda pode ser considerada insignificante. Seguem as práticas com estas características:

Duração das Iterações: Definir o tamanho da iteração é uma decisão, junto com diversas outras, que costuma ser tomada no início do projeto com a participação do cliente. Normalmente esta decisão é válida para todas as iterações, mas uma iteração em particular pode ter uma duração diferente. Mesmo nesta situação, a definição de uma nova iteração não consome um tempo relevante.

Distribuição dos Requisitos: A distribuição de requisitos para os responsáveis pelo seu desenvolvimento é de execução quase imediata e não costuma criar conflitos em equipes pequenas e ágeis.

3.1.3. Práticas de Requisitos

Este grupo apresenta duas práticas de natureza semelhante – a especificação dos requisitos –, mas com esforço distinto. O trabalho feito nestas atividades pode ser feito todo antes das iterações, o que pode entrar em conflito com a natureza iterativa dos métodos ágeis, assim como durante as iterações, de acordo com os requisitos selecionados. As duas práticas são:

Estórias de Usuário: Escrever estórias de usuário envolve tanto o cliente quanto alguém da equipe de desenvolvimento. O esforço despendido nesta prática pode ser relevante.

Casos de Uso: Assim como as estórias de usuário, o cliente precisa estar presente, mas o desenvolvedor tem uma participação mais ativa na geração dos artefatos. O esforço necessário nesta prática costuma ser maior que o das Estórias de Usuário, o que faz com este grupo possua práticas com diferentes impactos na agilidade.

3.1.4. Práticas de Desenvolvimento

Este grupo engloba as diversas práticas que estão em uso no cotidiano do desenvolvimento de código. Em geral, a equipe despenderá algum esforço com estas práticas, o que poderia prejudicar a agilidade. Entretanto, elas não prejudicam a agilidade, pois o benefício obtido com estas práticas acaba contrapondo o trabalho feito. A seguir, as práticas que foram identificadas:

Desenvolvimento Coletivo do Código: A equipe precisa ter conhecimento das diversas partes do código e isso contribui para resolver questões de mudança na equipe.

Programação em Pares: A programação em pares aparentemente reduz a capacidade de produção da equipe, mas traz alguns benefícios como o código de maior qualidade.

Refatoração: As práticas sugerem a refatoração sempre que necessário. O esforço utilizado na refatoração costuma ocorrer simultaneamente ao desenvolvimento devido à sua íntima relação com a codificação.

Reuniões Diárias: As reuniões diárias sugerem a participação de toda a equipe. Pelo fato destas reuniões serem de curtíssima duração e corroborarem para a boa comunicação da equipe, esta prática não possui grande impacto negativo na agilidade.

Integração Paralela: Assim como a refatoração, a integração paralela ao desenvolvimento possui estreita relação com a codificação. O esforço utilizado exclusivamente nesta prática não é pontual e, dependendo da natureza do projeto, também pode ser insignificante.

Controle de Versões: Considerando que toda a infraestrutura e o conhecimento de controle de versão sejam dominados, esta prática costuma não exigir esforço adicional. Podem existir casos, no entanto, em que seja necessária a resolução de conflitos no código. Estes conflitos costumam ocorrer com pouca frequência em equipes pequenas.

Desenvolvimento Lado a Lado: Alocar a equipe de desenvolvimento não traz impacto negativo na iteração. Pelo contrário, pode facilitar a resolução de conflitos.

3.1.5. Práticas de Testes

O desenvolvimento de testes de unidade e de aceitação sempre exige um esforço considerável, que tende a diminuir na medida em que a equipe se habitua ao desenvolvimento orientado a testes. A execução destes testes, porém, costumam ser automatizadas e poupam bastante esforço. Não se pode deixar de considerar que o desenvolvimento de testes aumenta a qualidade do software e sua confiabilidade, o que pode reduzir um esforço futuro. Este grupo inclui: *Testes de Unidade*, *Testes de Aceitação, Execução dos Testes de Unidade, Execução dos Testes de Aceitação*.

3.1.6. Práticas de Reunião

Este grupo apresenta as práticas que propõem encontros dos participantes do projeto. Nota-se, entretanto, a ausência da prática de Reuniões Diárias neste grupo devido à sua curta duração e também à sua estreita relação com o aspecto iterativo do desenvolvimento ágil. Assim, este grupo é formado pelas seguintes práticas ágeis:

Planejamento da Iteração: Esta reunião ocorre no início da iteração para se discutir o que será feito nela. O impacto da reunião se resume ao tempo despendido pelos participantes nesta atividade, que não costuma ser grande, pois esta tende a ser guiada de maneira bem objetiva. Se o tamanho da iteração for muito pequeno, o impacto destas práticas no processo tende a aumentar, pois elas comprometem a força de trabalho de toda a equipe.

Reunião de Revisão da Iteração: O esforço desta reunião de fechamento da iteração tem um comportamento semelhante ao planejamento da iteração e também sofre a mesma influência do tamanho da iteração.

3.1.7. Práticas de Integração e Validação

Este grupo inclui práticas referentes à validação e integração do incremento. Estas práticas são:

Integração do Incremento antes da Validação: É uma prática em que a expectativa de esforço é moderada, mas a presença de alguns imprevistos é recorrente e pode aumentar a demanda de esforço.

Inspeção de Código: A inspeção de código pode representar um grande esforço em uma equipe com pouca sintonia. Na medida em que os envolvidos se habituam ao estilo de codificação recomendado, o esforço desta prática tende a diminuir.

Integração do Incremento Resultante: Funciona de forma semelhante à integração antes da validação, onde o esforço é moderado se nenhum imprevisto ocorrer.

Validação no Ambiente do Cliente: O ambiente do cliente é conhecido e estudado previamente, mas a validação neste ambiente pode se deparar com situações técnicas inesperadas, por isso o impacto é moderado.

3.1.8. Práticas de Documentação

Parte da documentação abordada no processo está presente neste grupo. Diferentemente das outras práticas, estas não são realizadas durante as iterações, mas sim no início ou na conclusão do projeto. Além disso, a documentação sofre influência do tamanho do projeto, pois um projeto maior naturalmente envolve mais informações. Suas práticas são:

Documentação Inicial: A documentação inicial do projeto é elaborada antes das iterações e depende do tamanho do projeto.

Projeto do Sistema: O projeto geral também é realizado antes das iterações e seu esforço é diretamente proporcional à complexidade do projeto.

Documentação Breve: A documentação breve ocorre no final de todas as iterações.

3.1.9. Práticas de Documentação Detalhada

As práticas relacionadas à documentação que não foram incluídas no grupo anterior abordam as informações de forma mais granular e despendem um esforço consideravelmente maior. Estas práticas são:

Projeto Detalhado do Sistema: A arquitetura do sistema é projetada de forma detalhada. Isto ocorre durante as iterações de acordo com os requisitos selecionados, e um grande esforço é necessário. Sofre influência da complexidade e/ou tamanho do projeto.

Documentação Operacional: Esta documentação ocorre no fim do projeto e inclui uma série de documentos.

3.2. Análise da Agilidade

O agrupamento das práticas em diferentes grupos ajuda a identificar o impacto das diversas práticas nos processos ágeis. Considerando estes grupos e as observações feitas na seção anterior, é possível valorar o impacto na agilidade do processo de cada uma das práticas. Quanto maior o esforço necessário em uma prática maior foi o valor de impacto atribuído a ela. Este impacto reflete a influência negativa que a prática exerce sobre o processo.

A tabela 2 apresenta a lista das práticas e seus respectivos valores de impacto. No grupo G1 todas as práticas receberam o valor 0 (zero) por se tratar de práticas essenciais que não afetam o processo propriamente. Outro grupo que também teve atribuído o valor 0 (zero) foi o G2, pois, como explicado anteriormente, o tempo para definir estas práticas é insignificante. O grupo G3 aborda duas técnicas de levantamento de requisitos conhecidas; como a escrita de estórias de usuário é mais simples e direta que a especificação de casos de uso, esta prática teve a metade do impacto. O grupo que reúne diversas práticas de desenvolvimento, o G4, teve um pequeno impacto por entender que o esforço para a sua execução pode ser pequeno e ao mesmo tempo ele é convertido em pontos que aumentam a agilidade, recebendo o valor 1. O grupo G5 trata dos testes e aqui é necessário um esforço maior para esta definição; na execução o impacto é bem menor por esta ser geralmente automática. O grupo G6 inclui dois tipos de reunião que são bastante diretas e por isso elas tiveram um valor pequeno (valor 2). O grupo G7 agrupa as práticas de integração e validação que dificultam uma análise mais objetiva; optou-se, portanto, por um valor médio que não penalize as situações extremas, ou seja, o valor 3. Os dois últimos grupos abordam as práticas de documentação que sempre costumam ser mais trabalhosas; o grupo G8, com a documentação mais breve, teve a metade do impacto atribuído ao grupo G9, com as práticas que exigem um maior nível de detalhamento.

Tabela 2. Impacto na agilidade das práticas.

Grupo	Prática	Impacto
G1	Lista de Requisitos	0
G1	Desenvolvimento	0
G1	Entrega do Sistema	0
G2	Duração das Iterações	0
G2	Distribuição dos Requisitos	0
G3	Use Stories	3
G3	Casos de Uso	6
G4	Desenvolvimento Coletivo de Código	1
G4	Programação em Pares	1
G4	Refatoração	1
G4	Reuniões Diárias	1
G4	Integração Paralela	1
G4	Controle de Versões	1
G4	Desenvolvimento Simultâneo	1
G5	Testes de Unidade	5
G5	Testes de Aceitação	5
G5	Execução dos Testes de Unidade	2
G5	Execução dos Testes de Aceitação	2
G6	Planejamento da Iteração	2

G6	Reunião de Revisão da Iteração	2
G7	Integração do Incremento antes da Validação	3
G7	Inspeção de Código	3
G7	Integração do Incremento Resultante	3
G7	Validação no Ambiente do Cliente	3
G8	Documentação Inicial	10
G8	Projeto do Sistema	10
G8	Documentação Breve	10
G9	Projeto Detalhado do Sistema	20
G9	Documentação Operacional	20

Com os valores expostos e as devidas explicações, propõe-se o seguinte cálculo de impacto na agilidade:

$$I = \sum_{i=1}^{7} IGi + \frac{\sum_{k=8}^{9} IGj}{N}$$

Onde:

I = Impacto na agilidade

N = Número de iterações do projeto

IGi = Impacto causado do grupo 1 ao grupo 7

IGj = Impacto causado pelos grupos 8 e 9

Para ilustrar esta fórmula, será calculado o impacto na agilidade de um dos processos acompanhados durante este trabalho. Este processo foi definido para um projeto de extensão desenvolvido em um laboratório da Universidade Federal de Santa Catarina, e é composto por dois desenvolvedores, um gerente e um cliente. Este processo é composto pelas práticas mostradas na tabela 3 e ainda se encontra em andamento. Para fins dos cálculos serão consideradas as doze iterações realizadas até o presente momento.

Tabela 3. Processo acompanhado para ilustrar a fórmula.

Atividade	Grupo	Prática	Impacto
Definição dos Requisitos	G1	Lista de Requisitos	0
	G3	Estórias do Usuário	3
Atribuição dos Requisitos às Iterações	G6	Planejamento da Iteração	2
	G2	Duração das Iterações	0
	G2	Distribuição dos Requisitos	0
Desen. do Incremento do Sistema	G1	Desenvolvimento	0
	G4	Refatoração	1
Validação do Incremento	G7	Integração do Incremento antes da Validação	3
	G5	Execução dos Testes de Aceitação	2
Validação do Sistema	G6	Reunião de Revisão da Iteração	2
Entrega Final	G1	Entrega do Sistema	0

	G8	Documentação Breve	10

Nesta situação, teríamos os seguintes valores:

$$I = (3+1+2+2+2+3) + 10/12 = 13,83.$$

Para se ter uma base comparativa e interpretar o valor obtido, o mesmo cálculo será feito com o método Extreme Programming (XP), considerando também 12 iterações:

$$I = (3+1+1+1+1+1+5+5+2+2+2+3) + (10+10+20)/12 = 30,33.$$

O impacto calculado para o XP, como esperado, foi superior ao impacto calculado para o processo em acompanhamento. Esta diferença pode ser entendida pelo fato do XP apresentar um considerável número de práticas a mais. Como a literatura classifica o XP como um método ágil, podemos interpretar que o processo acompanhado seria bastante ágil, o que vai ao encontro do enxuto conjunto de práticas presentes nele.

É interessante também considerar um processo em que o valor obtido represente um método não ágil. Para este propósito, definiu-se um processo hipotético fortemente baseado em XP, mas com algumas pequenas alterações. As práticas deste processo são: *Lista de Requisitos, Casos de Uso, Documentação Inicial, Planejamento da Iteração, Projeto do Sistema, Desenvolvimento, Projeto Detalhado do Sistema, Testes de Unidade e de Aceitação, Desenvolvimento Coletivo de Código, Programação em Pares, Refatoração, Reuniões Diárias, Controle de Versões, Integração do Incremento antes da Validação, Execução dos Testes de Unidade e de Aceitação, Inspeção de Código, Reunião de Revisão da Iteração, Validação no Ambiente do Cliente, Entrega do Sistema e Documentação Operacional*. Usando a fórmula proposta anteriormente, tem-se o seguinte valor:

$$I = (6+2+5+5+1+1+1+1+1+3+2+2+3+2+3) + (10+10+20+20)/12 = 43,00.$$

O resultado mostra que a adição de várias outras práticas fez o valor se distanciar do valor obtido no método XP e quase triplicar o valor obtido com o processo definido no projeto em acompanhamento. Isto pode sinalizar a existência de certos conflitos entre as práticas que corroboram para a diminuição da agilidade no processo e precisam ser mais bem investigados.

4. Conclusão

Este artigo teve como objetivo incorporar ao framework para definição de novos processos ágeis [1] informações acerca da agilidade no desenvolvimento de sistemas da informação. Para tanto, identificou-se relações de semelhança entre as diversas práticas do framework. Estas relações, assim como o acompanhamento na execução de processos, corroboraram para a realização de uma análise a respeito da agilidade que estas práticas apresentam no contexto do processo. Esta análise agrega ao framework [1] novas informações sobre agilidade, que permitem os envolvidos na definição do processo saberem, no momento da definição, o impacto na agilidade de cada uma das práticas selecionadas. A vantagem de ter acesso a esta informação facilita a revisão do

processo definido a fim de eliminar eventuais práticas que poderiam comprometer a agilidade do processo e, consequentemente, as demandas do sistema de informação.

A análise de processos ágeis é um assunto ainda não muito explorado na literatura. Um método para avaliar o grau de agilidade de processos de desenvolvimento é apresentado em [4]. Nele, a avaliação, feita em seis diferentes métodos ágeis, ocorre através do uso de uma ferramenta denominada 4-DAT. A 4-DAT avalia os processos sob o prisma de quatro dimensões. A primeira delas é relativa ao escopo do método e abrange os métodos em um mais alto nível, considerando itens como tamanho do projeto, tamanho da equipe, estilo de codificação, entre outros. A segunda dimensão, a única de maneira quantitativa, considera as seguintes características ágeis: flexibilidade, velocidade, leveza, aprendizado e sensibilidade. A terceira e quarta dimensões tratam da caracterização dos valores ágeis e do processo de software, respectivamente. A grande diferença da abordagem mostrada neste trabalho e a apresentada aqui é que aquela faz a análise de agilidade dos diferentes métodos baseados no que é encontrado na literatura, além da maioria das informações serem de natureza qualitativa.

Os valores apresentados aqui foram obtidos a partir do acompanhamento de dois processos em andamento, um em um ambiente acadêmico e outro em uma empresa. Os pontos não abrangidos por estes processos foram resolvidos através de referências na literatura e em outras experiências. Estes valores, entretanto, não são definitivos. Como trabalho futuro, espera-se aprofundar o estudo e o acompanhamento das práticas ágeis do framework em diferentes projetos para que a valores mostrados possam ser mais fundamentados e granularizados. Para tanto, novos processos ágeis estão sendo aplicados em outros projetos para que mais dados sejam obtidos e analisados.

5. Referências

[1] Vilain, Patrícia. Fagundes, Priscila B. Machado, Thiago L. A Framework for Selecting Agile Practices and Defining Agile Software Processes. SEKE, 2007.

[2] Pressman, Roger S. Engenharia de Software – 6. ed. McGraw-Hill, 2006.

[3] Sommerville, Ian. Software Engineering. Addison-Wesley, 2003.

[4] Qumer, Asif. Henderson-Sellers, Brian. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Information and Software Technology, 2008.

[5] Beck, Kent. Beedle, Mike. Cockburn, Alistair. Fowler, Martin. et al. Manifesto for Agile Software Development. http://www.agilemanifesto.org/.

[6] Jacobson, Ivar. A Resounding Yes to Agile Processes – But Also More. Cutter IT Journal, 2002.

[7] Beck, Kent. Fowler, Martin. Planning Extreme Programming. Addison-Wesley, 2001.