Análise da Aderência das Práticas Recomendadas pelos Métodos Ágeis: XP, Scrum e TDD com a Utilização de Ferramentas CASE

Márcia M. Savoine[1], André Magno C. de Araújo[1], Leandro B. de Moura[1]

[1]Curso de Sistemas de Informação – Instituto Tocantinense Presidente Antônio Carlos (ITPAC) – Av. Filadélfia, 586 - Setor Oeste - 77.816-540 – Araguaína – TO – Brazil

{savoine,amcaraujo,leandrobdemoura}@gmail.com

Abstract. *This paper specifies a metric for assessing the level of features adherence offered by CASE tools Scrum-Half, JUnit, Netbeans and Eclipse, with the key best practices for agile application development. Therefore, this work takes a measurement criterion, in which every evaluated practice is assigned a weight according to the degree of found evidence and also by means of an arithmetic expression is possible to determine the adherence of each practice in relation to the investigated CASE tools. As a contribution, describes how a development team can run the recommended practices by the agile methods in the application development with the help of the most appropriate CASE tool.*

Keywords: *CASE Tool, Agile Development, Level of Adherence.*

Resumo. *Este artigo especifica uma métrica para avaliar o nível de aderência das funcionalidades oferecidas pelas ferramentas CASE Scrum-Half, JUnit, Netbeans e Eclipse, com as principais práticas recomendadas pelas metodologias ágeis no desenvolvimento de aplicações. Para tanto, este trabalho adota um critério de aferição, no qual cada prática avaliada recebe um peso de acordo com o grau de evidências encontrado, e assim, por meio de uma expressão aritmética é possível determinar a aderência de cada prática em relação às ferramentas CASE investigadas. Como contribuição, descreve-se como uma equipe de desenvolvimento pode executar as práticas recomendadas pelos métodos ágeis no desenvolvimento de aplicações com o auxílio da ferramenta CASE mais adequada.*

Palavras-chave*: Ferramenta CASE, Metodologia Ágil, Nível de Aderência.*

1. Introdução

No mundo globalizado e de rápidos avanços tecnológicos, as oportunidades de negócio vêm e vão com a mesma rapidez desses avanços. Este cenário não é diferente para as organizações de desenvolvimento de *software*, que para manterem-se em um mercado competitivo precisam a todo instante buscar metodologias e ferramentas que garantam a agilidade no desenvolvimento, e também, a qualidade na entrega do produto final.

Nesse sentido, as metodologias ágeis preconizam práticas organizadas e passos estruturados durante todo o ciclo de desenvolvimento de um Sistema de Informação

(SI). A adoção de tais práticas torna-se uma alternativa viável nos dias atuais para gerenciar a crescente evolução dos requisitos de um *software*, e também uma forma de aproximar a equipe de desenvolvimento dos usuários finais.

As metodologias ágeis possuem os mais variados focos de uso, dentre as principais destacam-se: *XP, Scrum e TDD*, objetos de estudo deste trabalho.

A metodologia *XP* tem como principal característica a codificação em pares. Nessa modalidade de programação, o código-fonte é armazenado em um repositório compartilhado, e os programadores envolvidos no projeto trabalham sistematicamente revisando a codificação, de modo a reduzir os erros. A metodologia *Scrum* objetiva o gerenciamento de todas as funções e processos que se passam dentro de cada evento do projeto, obtendo uma visão detalhada e completa das funcionalidades desenvolvidas no tempo planejado ou não, enquanto que a metodologia *TDD* assegura que toda interface e funcionalidade desenvolvidas estão livres de problemas ou incompatibilidades presentes no ambiente de produção.

Apesar da crescente adoção das metodologias ágeis nos dias atuais, percebe-se que pouca atenção tem sido dada à investigação sobre como explorar os recursos disponíveis nas ferramentas CASE, na execução das práticas ágeis de desenvolvimento de *software*. Investiga-se nesta pesquisa os recursos oferecidos pelas ferramentas CASE: *Scrum-Half, JUnit, Netbeans* e *Eclipse*. Levou-se em consideração para a escolha destas ferramentas, a consolidação e o seu uso no mercado de trabalho e na academia. Dessa forma, este artigo especifica uma métrica que avalia o quão aderente estão às ferramentas CASE em relação a cada método ágil avaliado.

2. Metodologias Ágeis

Em fevereiro de 2001, um grupo inicial de 17 metodologistas com áreas de formação diferentes, definiram um manifesto para encorajar melhores maneiras de desenvolver software, chamado de Manifesto Ágil (Ambler, 2004).

Voltados para projetos específicos, os métodos ágeis, quando aplicados, proporcionam maior flexibilidade no que diz respeito a mudanças, dispensando boa parte do tempo gasto em análise e planejamento, além de eliminar uma vasta regulamentação e documentação, itens ditados pelos métodos tradicionais de desenvolvimento de *software*.

Para estas metodologias pesadas ou tradicionais, os métodos ágeis é uma resposta no que diz respeito a desenvolvimento de *software*, por possuir enfoque e valores bem diferentes dos métodos tradicionais; ou seja, enfoque nas pessoas e não em algoritmos e processos. A capacidade de adaptação à situação em questão, faz com que elas se adaptem a novos fatores decorrentes do desenvolvimento do projeto, em vez de seguir a risca o cronograma previsto para seu desenvolvimento.

2.1. XP – eXtreme Programming

Proposta por Beck (2000), em meados da década de 1990, a metodologia *XP* propõe uma maneira leve e eficiente, de baixo risco, flexível, previsível, científica e, de certa forma, divertida de se desenvolver *software*.

A XP é composta por 12 práticas extremamente flexíveis de se trabalhar, que especificam:

- **Programação em pares para o desenvolvimento de *software*:** contribui para o aumento da qualidade do código, por consequência o foco da equipe é aprimorado, além de nivelar o conhecimento da equipe de forma surpreendente e rápida.
- **Planejamento antes da execução do projeto:** estipula-se entre desenvolvedores e usuários o que é necessário ser feito, definindo a partir das estórias de usuário e seus requisitos as estimativas de prazo, o processo de desenvolvimento e o cronograma detalhado para que o *software* seja entregue nas datas estipuladas.
- **Desenvolvimento e fases curtas:** priorizar as estórias mais importantes e promover projetos simples através de versões suficientes para este momento e não para sempre.
- **Metáforas são utilizadas para incentivar o time:** consiste em realizar descrições do *software* que auxiliam todos os envolvidos no projeto a entenderem os elementos básicos e seus relacionamentos.
- **Design simples:** a complexidade desnecessária é removida no momento de sua descoberta, priorizando as formas mais simples possíveis de desenvolvimento.
- **Conter teste é um dos fundamentos mais importantes da XP, por dar segurança ao grupo:** testes são fundamentais em qualquer projeto XP.
- **Refatoração:** consiste na técnica de melhoramento do código-fonte, visando à simplicidade e organização do mesmo, sem a perda da finalidade principal que é a disponibilização da funcionalidade.
- **Propriedade coletiva:** deve-se ter a obrigatoriedade de realizar qualquer modificação no código a qualquer momento, visto que, todos são responsáveis pelo sistema inteiro.
- **Integração contínua:** integrar e testar o código é uma tarefa que deve ser feita após algumas horas de desenvolvimento ou no final do dia, recomenda-se ter uma máquina disponível somente para este fim.
- **Semana de 40 horas:** é imprescindível ter controle das horas trabalhadas, sempre tomando cuidado para não exagerar na quantidade de horas trabalhadas.
- **Clientes sempre juntos aos desenvolvedores:** ter o cliente presente significa ter sempre por perto a pessoa que fará uso do sistema, expondo suas opiniões, resolvendo conflitos, mostrando quais são as principais prioridades.
- **Padronização do código:** o padrão deve enfatizar a comunicação e ser adotado voluntariamente por todo o time.

Esta metodologia é utilizada quando ocorre a fase de desenvolvimento e produção do *software*, a fim de empregar suas práticas para auxiliar a equipe de desenvolvimento na escrita de código-fonte coerente, longe de falhas e exaustivamente testado. Tem o foco no desenvolvimento organizado de *software*, bem como, nas boas

relações entre os membros da equipe e os clientes, prioriza-se o diálogo entre as partes envolvidas no projeto, de forma que, sempre haja uma concordância entre esses personagens.

2.2. Scrum

A *Scrum* é um tipo de metodologia ágil com foco no gerenciamento de projetos ágeis. É um *framework* e, irá servir como um guia de boas práticas para se atingir o sucesso do projeto de *software*, além de simples para colaboração em equipe e eficaz em projetos complexos. Fornece um pequeno conjunto de regras a fim de criar uma estrutura suficientemente eficaz para que as equipes sejam capazes de se concentrar na resolução e na criação de inovações.

Para *Schwaber, et al, (2004): Scrum* não é um processo previsível, ele não define o que fazer em toda circunstância.

As equipes são formadas pelo *Product Owner,* o *Team* ou Equipe de Desenvolvimento e pelo *ScrumMaster*. O *Product Owner* ou o Dono do Produto, apresenta-se como a pessoa responsável por administrar e deliberar decisões que incidem diretamente no *backlog* do produto. Tem a função de coordenar e organizar os itens contidos no *backlog*, a fim de definir as prioridades de cada tarefa, promovendo assim, maior clareza na definição das metas do projeto, o que torna a equipe focada na conclusão das missões impostas pelo projeto de *software.*

O *team* ou a equipe de desenvolvimento é, justamente, aquela que produz as funcionalidades a fim de entregar algo utilizável para o cliente. Geralmente as equipes são pequenas (de 5 a 9 pessoas), já as equipes grandes, na maioria, se comportam como várias equipes pequenas.

Equipes *Scrum* são auto-organizadas e transfuncionais. Equipes auto-organizadas escolhem a melhor forma de realizar seu trabalho, ao contrário de serem dirigidas por outros de fora da equipe. [...] A equipe modelo no *Scrum* é projetada para aperfeiçoar a flexibilidade, criatividade e produtividade. (*Schwaber, et al, 2004).*

O *Scrum Master* é a pessoa que garante que o Scrum está sendo aplicado em sua integridade, ou seja, em sua essência. Uma espécie de líder, ao qual, a equipe busca soluções a respeito das teorias, regras e práticas da metodologia. Também tem a função de remover impedimentos e proteger a equipe contra riscos e interferências externas e excesso de otimismo, que podem levar ao insucesso do projeto. O desenvolvimento acontece por uma série de iterações bem definidas, chamadas *Sprints.* Uma reunião é feita antes de cada *Sprint* para promover o planejamento juntamente com o *Product Owner,* a fim de designar prioridades, selecionar e estimar as tarefas que o time irá realizar dentro de cada *Sprint.*

Para *Schwaber, et al,* (2004): A *sprint* é o coração do Scrum, um *time-box* de um mês ou menos durante o qual uma versão potencialmente utilizável de um incremento do produto é criada.

Durante o decorrer da S*print* o controle é feito pelo time, através de reuniões diárias, denominadas *Daily Meeting*, que têm exatamente 15 minutos de duração, e geralmente são feitas no início do dia. Ao final de cada *Sprint* é feita uma reunião de finalização para se verificar se as funcionalidades foram implementadas corretamente.

A lista de itens e suas respectivas prioridades que inclui tudo que é necessário serem realizados durante a Sprint está contida no *Product Backlog*. Os valores de negócio para cada item são especificados neste momento, bem como a descrição da prioridade, desta forma, é possível medir o retorno que trarão para o projeto. Por outro lado, o *Impediment Backlog* é composto pelos itens que possam vir a impedir o andamento do projeto, tais itens muitas vezes estão associados a possíveis riscos, como a instalação de ambientes para desenvolvedores, ou de determinado banco de dados do projeto.

Neste sentido, Scrum *é* ideal para projetos que estão em constante dinamicidade e a mudança de requisitos.

2.3. TDD – Test Driven Development

TDD - Test Driven Development ou Desenvolvimento Guiado por Testes consiste numa metodologia de desenvolvimento de *Software* em que primeiro são criados os testes, e somente posteriormente é escrito o código necessário para passar por eles. Recomenda-se utilizar esta metodologia não apenas em projetos que utilizam metodologias ágeis, mas cabe perfeitamente em qualquer outro projeto em que se deseja realizar os devidos testes antes de se implementar qualquer funcionalidade.

Foi atribuída ao TDD grande responsabilidade, sendo definido como o cerne de todo o desenvolvimento ágil. Fora do ambiente ágil, entretanto, torna-se um subprocesso que acaba fazendo parte de todo o processo de desenvolvimento de *software*. (Germano, 2010)

O teste de unidade consiste em se testar módulos individuais do sistema, como por exemplo: funções ou métodos individuais de um objeto, classes de um objeto com vários atributos e métodos, componentes compostos que constituem diferentes objetos ou funções.

Tais funções e métodos são os componentes mais simples que compõem o código-fonte do *software* e seus testes, a fim de gerar uma sequência de chamadas destas rotinas com diferentes parâmetros de entrada, que devem incluir: testes isolados de todas as operações associadas ao objeto a ser testado; atribuir e indagar todos os atributos associados ao objeto; complementar o exercício do objeto em todos os estágios possíveis, isso implica que, todos os eventos que causam mudanças de estado no projeto devem ser simulados.

O teste de integração indica as falhas ocorridas na fase de junção dos componentes, ou seja, é o teste do sistema resultante dos problemas ocorridos na interação de tais componentes. É neste momento que se realiza a verificação se os componentes funcionam realmente em conjunto, se são chamados corretamente e se a transferência de dados ocorre de maneira correta no tempo correto, por meio de suas interfaces. Um dos principais problemas ocorridos durante o teste de integração é a localização de erros.

O teste de sistema é o processo de teste que será distribuído aos clientes. A meta principal desse processo é aumentar a confiança do fornecedor de que o sistema atende aos requisitos.

3. Ferramentas de Desenvolvimento de Software

Uma ferramenta de desenvolvimento é um *software* que está contido no escopo de uma IDE - *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, ao qual dá apoio ao desenvolvimento de *software*, com o objetivo de agilizar o processo de desenvolvimento. Estas ferramentas simplificam o trabalho de programadores, dando-lhes diversas alternativas e métodos capazes de simplificar o código fonte, que vão desde a técnica de refatoração, por exemplo, até o fornecimento de ambientes gráficos capazes de simularem o *software* em pleno funcionamento.

No desenvolvimento de código e na aplicação das doze práticas exigidas pela metodologia XP, faz-se necessário utilizar várias ferramentas e IDE's de apoio para as mais diversificadas linguagens disponíveis. Neste trabalho serão exemplificadas três: *Netbeans*, *Scrum-Half* e *Eclipse*, por possuírem características explícitas que remetem às práticas da metodologia XP. E, na elaboração dos testes de *software*, premissa pautada na metodologia TDD, utiliza-se um *framework* para a realização dos testes na demonstração de linhas de código, sendo o *JUnit* um exemplo de ferramenta para a realização dos clássicos testes de unidade.

3.1. Características das Ferramentas

Ao realizar a implementação e o gerenciamento de projetos na abordagem ágil, o *Scrum-Half* oferece mecanismos exclusivos da metodologia Scrum. Possui o diferencial de acompanhar as etapas necessárias para o desenvolvimento de um projeto de *software*, auxiliando as reuniões para elaboração e encerramento da *Sprint* e também no dia a dia do trabalho de desenvolvimento. (Marins, *et. al*, 2011)

A ferramenta *JUnit* é específica para o desenvolvimento na linguagem de programação Java, oferece técnicas de manipulação de testes unitários uma das premissas da metodologia TDD; enquanto que o *Netbeans* e o *Eclipse* são robustos ambientes de desenvolvimento de *software* capazes de oferecerem aos desenvolvedores mecanismos clássicos abordados na metodologia XP.

Escolher as ferramentas certas pode auxiliar a equipe de desenvolvimento ágil a ter maior produtividade, qualidade no *software,* facilitando a interação com o cliente e comunicação entre os membros do time. Ressalta-se que cada ferramenta possui sua particularidade, e cada metodologia ágil se utilizará destas particularidades para suprir suas necessidades.

4. Análise da Aderência de Ferramentas CASE com Práticas Ágeis

Nesta fase da pesquisa, foram definidas quais as práticas mais importantes existentes nas metodologias que possuem cunho decisivo na sua definição, sem as quais não seria possível designá-las como metodologia de desenvolvimento ágil.

A ênfase atribuída, em particular, às práticas das metodologias é usada como forma de exprimir se cada ferramenta é capaz de implementá-la de acordo com o que dita à metodologia, sendo assim, uma forma de medir a aderência que envolve as ferramentas em relação ao método ágil.

A Tabela 1 mostra as principais práticas em relação à sua metodologia, em que, observam-se as doze práticas da metodologia XP, a *Sprint* e os artefatos do Scrum com

suas respectivas sub-práticas e, na metodologia TDD, os testes de unidade, integração, sistema e aceitação.

Tabela 1. Principais práticas das metodologias ágeis XP, Scrum e TDD.

METODOLOGIA		PRINCIPAIS PRÁTICAS			
1	**XP**	1.1	Programação em pares para o desenvolvimento de *software*		
		1.2	Planejamento prévio antes da execução do projeto		
		1.3	Desenvolvimento e fases curtas		
		1.4	Metáforas são utilizadas para incentivar o time		
		1.5	Ter um design simples		
		1.6	Conter testes		
		1.7	Refatoração		
		1.8	Propriedade coletiva		
		1.9	Integração continua		
		1.10	Semana de 40 horas		
		1.11	Clientes sempre juntos aos desenvolvedores		
		1.12	Padronização do código		
2	**SCRUM**	2.1	Sprint	2.1.1	Reunião de Planejamento do Sprint
				2.1.2	Objetivo do Sprint
				2.1.3	Cancelamento de um Sprint
				2.1.4	Scrum diário
				2.1.5	Revisão do sprint
				2.1.6	Retrospectiva do sprint
		2.2	Artefatos do Scrum	2.2.1	Backlog do produto
				2.2.2	Backlog do Sprint
				2.2.3	Incremento
				2.2.4	Definição de “Pronto”
3	**TDD**	3.1	Teste de unidade		
		3.2	Teste de integração		
		3.3	Teste de sistema		
		3.4	Teste de aceitação		

Para determinar o índice de aderência dos processos denotados como PAI - *Process's Adherence Index ou* Índice do Processo de Aderência, buscou-se comparar as evidências geradas pelas principais práticas ou características das metodologias ágeis XP, Scrum e TDD, perceptíveis na Tabela 1.

A partir do relacionamento observado entre as práticas das metodologias e o uso das funcionalidades das ferramentas CASE, foi atribuído um grau de aderência para a prática em relação ao uso da ferramenta de desenvolvimento, considerando os critérios mostrados na Tabela 2, em que, a atribuição do peso foi estabelecida mediante o mérito do grau estipulado, ou seja, o grau máximo ou completo terá peso 1; os graus intermediários: fraco e forte terão peso 0,3 e 0,8 respectivamente; e o grau NI(Não Implementado) receberá peso 0. Cada peso estipulado é de fundamental importância para o cálculo do PAI.

Tabela 2. Critério de graduação entre as evidências

GRAU	CONDIÇÃO	PESO
NI (Não Implementado)	Não implementado por metodologias ágeis	0
Fraco	< 50% dos produtos de trabalho são atendidos na prática específica	0,3

Forte	> 50% dos produtos de trabalho são atendidos na prática específica	0,8
Completo	Todos os produtos de trabalhos são atendidos na prática específica	1

Fonte: Andrade, *et. al,* 2011 - Adaptado.

Calcula-se o PAI através da média aritmética simples entre as práticas específicas do processo, de acordo com a expressão (1).

$$\textbf{PAI} = \frac{1}{n}\sum_{i=1}^{n} x_i \ * \ 100\% \qquad (1)$$

Onde, tem-se:

- PAI é o *Process's Adherence Index;*
- **X** é o peso de cada prática específica e;
- **N** é a quantidade de práticas da metodologia em relação àquela ferramenta analisada.

O índice foi alcançado utilizando a expressão (1) justificada pelo PAI, em que foram apontados o nível do grau e seu respectivo peso apresentado na Tabela 2. O peso foi estabelecido de acordo com as evidencias das práticas existentes em cada ferramenta. Desta forma, é possível estabelecer um percentual de quanto cada ferramenta adere às principais praticas das metodologias ágeis *XP, Scrum e TDD.*

4.1. Resultados Obtidos

Utilizando como exemplificação tem-se a ferramenta Eclipse em relação à metodologia XP e obtém-se a relação de peso e grau estabelecidos para esta ferramenta em relação à metodologia em questão, Tabela 3.

Tabela 3. Índice do processo de aderência do Eclipse em relação à metodologia XP

FERRAMENTA	METODOLOGIA	PRÁTICAS	GRAU	PESO
Eclipse	**XP**	1.1	Completo	1
		1.2	NI	0
		1.3	Completo	1
		1.4	NI	0
		1.5	Completo	1
		1.6	Completo	1
		1.7	Completo	1
		1.8	Completo	1
		1.9	Completo	1
		1.10	NI	0
		1.11	NI	0
		1.12	Completo	1
PAI				**67%**

Verifique-se pela Tabela 3 a obtenção do índice de 67% de aderência da ferramenta Eclipse em relação à metodologia ágil XP, realizando o somatório de todos os valores que foram indicados no item PESO da Tabela 3, e utilizando a expressão (1) tem-se o total, como é mostrado na expressão (2).

(2)

$$\textbf{PAI} = \frac{1+0+1+0+1+1+1+1+1+0+0+1}{1} = 0{,}66666 * 100\% = \textbf{67\%}$$

Pode-se perceber que, na Tabela 4, tem-se o agrupamento final dos resultados obtidos com a medição do nível de aderência das demais ferramentas confrontando com as metodologias ágeis estudas.

Tabela 4. Nível de aderência total das ferramentas com as metodologias ágeis

FERRAMENTA	METODOLOGIA	ADERÊNCIA
Eclipse	XP	67%
	Scrum	0%
	TDD	83%
Netbeans	XP	67%
	Scrum	0%
	TDD	83%
Scrum-Half	XP	48%
	Scrum	100%
	TDD	0%
JUnit	XP	62%
	Scrum	0%
	TDD	40%

Observando os resultados, percebe-se que as ferramentas Netbeans e Eclipse conseguem alcançar um índice de aderência alto em relação às metodologias ágeis XP e TDD, respectivamente 67% e 83%; justamente por atenderem e implementarem boa parte das doze práticas da metodologia XP e auxiliarem no apoio de desenvolvimento de testes de *software*, tanto no código-fonte, quanto nos demais tipos de testes elucidados pela metodologia TDD.

Todavia, essas ferramentas não satisfazem os processos exigidos pela metodologia *Scrum*, justamente por não terem o foco no gerenciamento do projeto. Isto comprova que as duas ferramentas, Eclipse e Netbeans, são mais eficazes em metodologias que englobam programação e testes em suas características. Por outro lado, não são indicadas como ferramentas de suporte aos processos como os do *Scrum*.

A ferramenta *Scrum-half* contempla um índice de aderência de 100%; ou seja, atende plenamente as práticas desta metodologia. Porém, possui pouca expressão no apoio ao desenvolvimento de projetos da metodologia XP, por se adequar, a poucas de suas práticas, apresentando um índice de 48%, além de não satisfazer as exigências da utilização dos testes existentes na metodologia TDD, com um índice de 0%.

O *framework JUnit* se mostra eficaz nas práticas da metodologia XP, exibe um índice de processo de aderência com uma relevância de 62%, por ser possível implementar em seu escopo a técnica de testes unitários, uma das práticas valorizadas pela metodologia XP. Em contrapartida, não apresenta significância na metodologia Scrum com um índice de 0%. E como produz somente testes unitários, não atende plenamente as práticas da metodologia TDD, apresentando um índice de 40% nesta metodologia.

É perceptível que, nas metodologias ágeis estudadas *XP, Scrum e TDD*, a capacidade de andarem juntas no desenvolvimento do projeto é extremamente possível

com o uso de ferramentas adiram às suas práticas, o que torna o desenvolvimento harmonioso; ou seja, com o princípio de complementaridade, em que, quando uma ferramenta não é capaz de alcançar um índice satisfatório, a outra se apresenta para suprir as lacunas deixadas.

5. Conclusão

De acordo com a análise realizada, percebeu-se que a metodologia *Scrum* é voltada exclusivamente para o gerenciamento da equipe, pois somente com a ferramenta Scrum-Half ela tem aderência, sendo de 100%. Então, faz-se necessário o uso de outra metodologia para realizar a parte de execução e desenvolvimento do *software* e, as práticas da metodologia XP preencheriam estas lacunas, pois apresenta-se com todas as ferramentas um índice total de aderência, tomando como média 61%; bem como, a metodologia TDD satisfaria as fases de testes dentro do projeto, apresentando uma média de aderência total com todas as ferramentas de 68,7%.

Corroborando com os resultados obtidos é perceptível que uma metodologia complementa e se agrega uma a outra. Todavia, a equipe deve selecionar ferramentas de desenvolvimento compatíveis com cada metodologia em questão; o que foi comprovado com os resultados definidos pelo PAI, a respeito das ferramentas.

A importância de ser extraído o índice do processo de aderência está em proporcionar aos desenvolvedores e às equipes de desenvolvimento, uma solução na determinação e escolha de ferramentas coerentes com a metodologia ágil a ser definida em cada fase do projeto ágil. A análise do índice do processo de aderências das ferramentas *Eclipse, Netbeans, Scrum-Half e JUnit* em relação às metodologias ágeis *XP, Scrum e TDD*, mostra aos desenvolvedores de projetos ágeis mais um mecanismo de apoio à escolhas das melhores ferramentas para o auxílio às práticas de suas metodologias, e torna possível determinar qual ferramenta de desenvolvimento tem melhor adequação ao tipo de projeto realizado. Como também, demonstra que as ferramentas podem trabalhar em consonância, sendo uma extensão uma da outra, ou seja, um desenvolvedor pode obter excelentes resultados em um projeto de *software* utilizando-as, pois existe um domínio em todos os aspectos do desenvolvimento do *software*, visto que nenhum processo fica sem ser explorado.

Referências

Ambler, Scott W., (2004) Modelagem ágil: Práticas eficazes para a programação eXtrema e o Processo Unificado. Porto Alegre-RS: Bookman.

Andrade, Anderson S., Araújo, André M. C., Andrade Hilson G. V., Borges, Wanderson R. (2011) Análise de aderência da Revisão de *software* nos processos do Grupo Engenharia CMMI. Centro de Informática – Universidade Federal de Pernambuco.

Beck, Kent. (2000). Programação Extrema (XP) Explicada, Estados Unidos: Addison-Wesley.

Germano, Victor Hugo. (2010) Desenvolvimento Orientado a Teste. Visão Ágil. Santa Bárbara do Oeste, SP. V. 2, N. 15. http://www.visaoagil.com/static/downloads/edicoes/VA_02.pdf.

Marins, Diego R., NT, José A. Rodrigues, XEXÉO, Geraldo B., SOUSA, Jano M. de. (2011) Scrum-Half: Uma ferramenta Web de apoio ao Scrum. Departamento de Ciência da Computação – COPPE/UFRJ – Universidade Federal do Rio de Janeiro.

Schwaber, Ken, Sutherland, Jeff, (2004) O guia do Scrum: O guia definitivo para o Scrum, as regras do jogo. http://www.scrum.org/storage/Scrum%20Guide %202011%20-% 20PTBR.pdf>.