 ISSN: 2175-1897

Análise de aspectos do processo de desenvolvimento de software em métodos ágeis.

Raquel Bortoluci
Centro Estadual De Educação Tecnológica Paula Souza - SP- Brasil
raquelborto@gmail.com

Prof. Dr. Marcelo Duduchi
Centro Estadual De Educação Tecnológica Paula Souza - SP- Brasil
mduduchi@gmail.com

Prof. Alex Oliveira Estevam
Centro Universitário Senac - SP- Brasil
alexestevam@gmail.com

Resumo

O presente artigo analisa diversos aspectos que envolvem o processo de desenvolvimento de software a partir da perspectiva da metodologia ágil como resposta às novas demandas de mercado que requerem softwares que atendam ao levantamento de necessidades de maneira dinâmica em curto espaço de tempo. O principal objeto de estudo deste trabalho é a entrega do produto ao cliente, as equipes de desenvolvimento e validação e verificação do software em métodos ágeis. Uma vez que existem várias abordagens de desenvolvimento ágil, o *Scrum* foi escolhido para parametrização da pesquisa por ser uma das abordagens mais utilizadas. Dentro desta perspectiva é possível verificar a visão inicial das pessoas envolvidas neste tipo de processo de desenvolvimento de software para que se possa identificar quais são as melhores práticas a serem usadas na implantação de métodos ágeis. Com a pesquisa pode-se verificar que a maioria das pessoas não tem uma visão holística sobre os métodos ágeis dificultando assim a percepção real dos valores ágeis.

Palavras-chave: desenvolvimento de software, métodos ágeis, Scrum

Abstract

This paper analyzes several aspects involving the software development process from agile methodology perspective. The paper presents a perspective on Agile as a response to the growing marketing demand that requires faster software delivery in a constant change ecosystem. The main purpose of this work is to study the software product deliverables to costumers, development team, and validation and verification of the software when using agile methods and techniques. Since there are many agile approaches, the Scrum framework was naturally selected to serve as a basis parameter of the research as it is known to be one of the largest used agile processes. In this perspective, it is also possible to claim the initial view of the persons involved in this software development process in order to identify the best practices to be used in agile methods implementation. With the research, it was possible to verify that the persons do not have a holistic perception about agile methods what become difficult the real perception of agile values.

Keywords: software development, agile methods, scrum.

1 Introdução

A atual demanda do mercado requer novas e diferentes soluções quanto à dinâmica de forma que ofereçam resposta em curto espaço de tempo. Para as empresas de software, a entrega de um produto confiável, de qualidade que atenda às exigências dos clientes rapidamente é uma tarefa difícil. Em resposta à esta problemática, observa-se a utilização crescente de métodos ágeis para desenvolvimento de software.

Segundo Hansmann e Stober (2010), o desenvolvimento ágil tem por objetivo simplificar o processo de desenvolvimento de software reduzindo a complexidade de planejamento, focando nas necessidades do cliente e moldando equipes colaborativas e participativas.

Dos diversos métodos e *frameworks* ágeis existentes na atualidade, os mais reconhecidos são o *Crystal*, o *Scrum*, o *Dynamic Systems Development*, o *Feature Driven Development*, e o *Extreme Programming*. Todos eles apresentam processos efetivos para o desenvolvimento ágil e, apesar de abordagens diferentes, seguem os princípios preconizado pelo manifesto ágil (RICO, SAYNO e SONE, 2009).

O *Scrum* é a prática que mais tem sido utilizada para a implementação dos princípios ágeis (VERSION ONE, 20013). Ela foi utilizada como base para o desenvolvimento desta pesquisa que tem por objetivo verificar a visão geral das pessoas sobre processos ágeis para que posteriormente se possa analisar e discutir as práticas utilizadas na implementação de métodos ágeis, levando em consideração a visão inicial das pessoas envolvidas em desenvolvimento de software ou projetos.

Os principais aspectos analisados foram quais devem ser valores entregues aos clientes ao final de uma iteração, também conhecida como *Sprint no Scrum*, a importância do fator humano no desenvolvimento ágil e quais as principais formas de se validar e verificar um software neste contexto.

2 Desenvolvimento ágil e *Scrum*

O modelo de desenvolvimento ágil é baseado em pequenas iterações e com entregas ao final de cada iteração, seguindo os princípios ideológicos do manifesto ágil (BECK, K. et. al, 2001):

- Indivíduos e interações mais que processos e ferramentas
- Software funcional mais que documentação abrangente
- Colaboração do cliente mais que negociação de contrato
- Responder às mudanças mais que seguir o planejamento

O *Scrum* é definido como um *framework*, no qual são empregados diversos processos e técnicas a fim de gerenciar o desenvolvimento de produtos complexos. O framework não define qual o processo e a técnica a ser utilizada para elaboração do projeto (SCHWABER; SUTHERLAND, 2013). Ele é utilizado para orientar atividades de desenvolvimento dentro de um processo que incorpora as atividades de especificação de requisitos, análise, projeto, evolução e entrega do software (PRESSMAN, 2011).

De acordo com Hansmann e Stober (2010), no desenvolvimento utilizando *Scrum* funciona a partir de um time onde existem três papéis fundamentais. Destaca-se primeiramente a equipe, responsável por desenvolver e testar o produto. Verifica-se também a importância do *scrum master* que é responsável por todo o processo de *Scrum*

e ajustes no projeto e na organização. E finalmente, o *product owner*, ou o cliente, é quem representa os interesses de todos os *stakeholders* (a parte interessada). Ele quem é responsável por determinar os requisitos.

Todos os requisitos são listados e priorizados em uma lista que é chamada de *product backlog*. O Desenvolvimento é feito em pequenas iterações, chamadas de *sprints*. A *sprint* é composta por atividades de design, desenvolvimento, teste e documentação e ainda outras atividades pertinentes ao desenvolvimento e completude de um produto de software. No começo de cada *sprint*, uma lista de casos de usos é definida pelo *product owner*, pelo *scrum master* e pelo time de desenvolvimento. Os casos de usos que são completados durante as *sprints* são documentados no documento de *sprint result*. Utilizando a terminologia mais conhecida em *Scrum* e fazendo analogia aos casos de uso selecionados para o *product backlog*, verifica-se a utilização do termo estórias de usuários, que são definidas por Moreira (2013) como o principal fator para determinar o que precisa ser construído pela equipe, onde as funcionalidades que serão úteis aos clientes são apresentadas.

No início de cada *sprint* é feita uma reunião de planejamento, na qual o *scrum mast*er, o *product owner* e o time de desenvolvimento definem o *sprint backlog*. O *product owner* prioriza os requisitos, já o time de desenvolvimento determina o trabalho estimado e os itens que se compromete a entregar no final da *sprint* (HANSMANN, STOBER. 2010)

Durante toda a execução do projeto, uma reunião diária de cerca de quinze minutos é organizada pelo *scrum master* com o time de desenvolvimento, a fim de verificar o progresso e identificar itens que requerem mais atenção ou até mesmo ajuste no *sprint backlog*. No final de cada *sprint* é feita uma reunião para revisar o resultado alcançado (HANSMANN, STOBER. 2010).

No *Scrum* as equipes devem ser auto organizadas onde os seus membros devem se planejar para alcançar o resultado esperado, diferentemente do desenvolvimento de software preditivo onde o planejamento é feito por um gerente de projetos que tem a função de verificar que todos estão fazendo o seu trabalho. Neste caso, por um lado se tem um risco menor de falhas de planejamento, mas por outro lado, como cada pessoa é responsável por fazer tarefas que cabem a sua função, elas não precisam ser colaborativas e inovativas durante o projeto.

Visto que no *Scrum* as equipes se auto-organizam, as pessoas com melhores qualidades vão avançar e poderão apoiar os outros membros da equipe, destacando, assim, a importância de equipes colaborativas e multifuncionais. Neste caso, apesar da equipe toda ser responsável pela entrega do produto, o time irá definir como fazer as tarefas e cada pessoa será responsável por sua parte. (SCHWABER; SUTHERLAND. 2012).

O *Scrum* não determina um processo de teste a ser seguido, mas muitas equipes ficam em dúvida de qual seria a melhor forma de validar e verificar o software neste contexto. Conceitua-se a seguir os níveis de testes de acordo com o ISTQB (2001) usados na análise deste trabalho.

No nível de teste de componente, também conhecido como teste unitário, o objetivo principal é garantir que cada unidade individual do software esteja funcionando de acordo com sua especificação. O foco está em testar módulos separadamente, tais como programas, objetos e classes. Os testes são muitas vezes desenvolvidos e executados de forma isolada do resto do sistema e geralmente envolvem o programador que escreveu o código.

O teste de Integração verifica a interface entre os componentes e interações com diferentes partes de um sistema. Pode haver mais de um nível de teste de integração, tais como a integração de componente que testa as interações entre os componentes de

software e é feito após testes de componentes e os testes de integração do sistema que testa as interações entre sistemas diferentes, ou entre hardware e software.

No nível de teste de sistema, o foco está no comportamento do sistema como um todo. Os casos de teste são derivados de especificações de requisitos, processos de negócios, casos de uso, estórias de usuários ou outras descrições de texto de alto nível ou modelos de comportamento do sistema.

Os testes de aceitação servem para criar confiança no sistema, ele não tem por objetivo encontrar defeitos. Casos de teste de aceitação são baseados nos requisitos dos clientes e devem demonstrar que o sistema ou componente está pronto para entrega aos *stakeholders*.

Na figura 1 de Crispin e Gregory (2009) apresenta-se a abordagem que divide os testes em quadrantes.

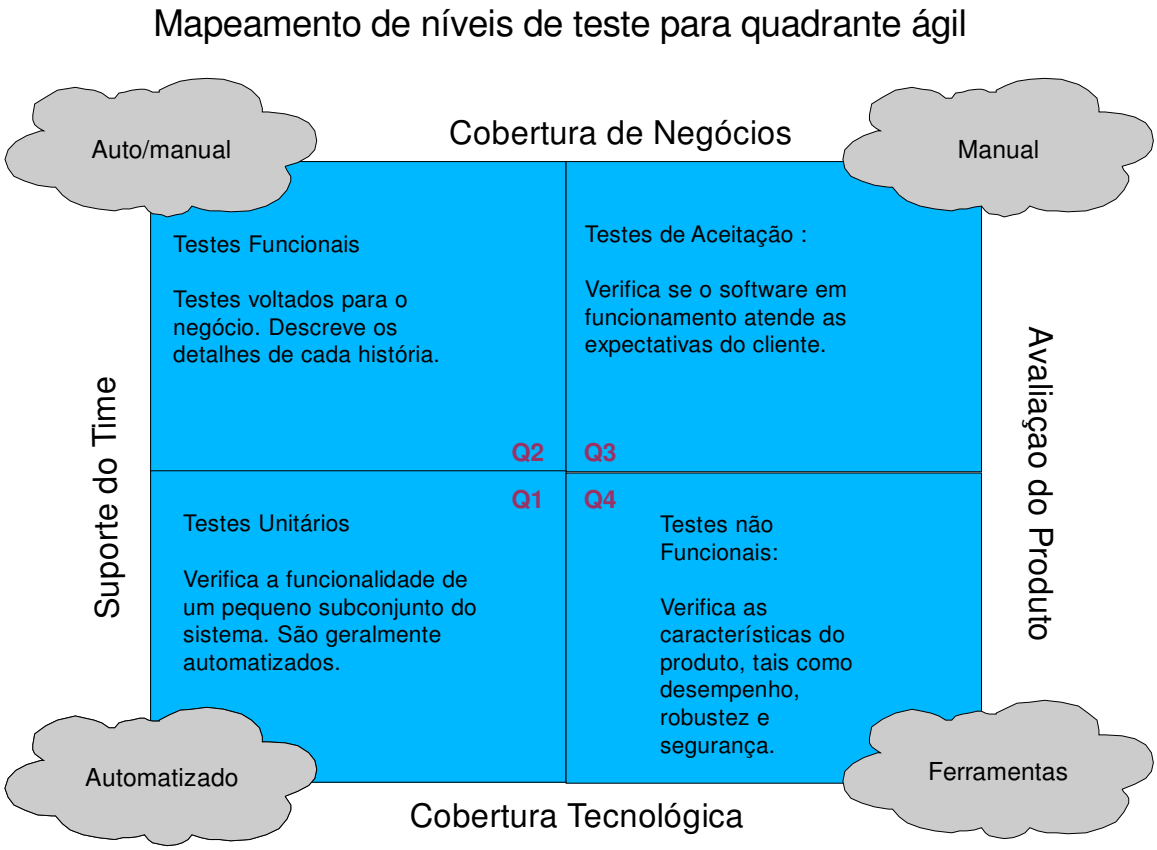

Figura 1: Quadrante de teste ágil

Fonte: Adaptado de Crispin e Gregory (2009) – *Agile Testing*

No quadrante 1 (Q1) aparecem os testes unitários que são, geralmente, automatizados. Estes testes ajudam o programador a entender o que o código precisa realmente fazer sendo um guia de projeto.

No quadrante 2 (Q2) aparecem os testes funcionais. Estes testes devem ser escritos para cada estória antes de começar a codificação, pois eles ajudam o time a entender o código que deve ser escrito. Seu principal objetivo é verificar se o software atingiu as expectativas de negócio, no qual as estórias são utilizadas para

desenvolvimento. Parte destes casos de testes deve ser automatizada e utilizada para a verificação continua do software.

No quadrante 3 (Q3) aparece a verificação se software atendeu as necessidades dos clientes ou de mercado. Costuma-se fazer uma demonstração no final da *sprint* para que os clientes analisem o que está sendo entregue.

No quadrante 4 (Q4) aparecem as de ferramentas para verificar requisitos não funcionais, tais como performance, segurança e robustez. Quando se está planejando as atividades não se deve pensar apenas nas necessidades de negócio, mas também nos riscos que se pode ter em termos não funcionais e incluir estes testes no planejamento.

3 Descrição da pesquisa

A pesquisa foi realizada em uma grande empresa do setor de tecnologia da Informação durante uma apresentação não presencial sobre transformação ágil e customização de processos. Utilizou-se para apresentação e para definição de perguntas o *Scrum* como abordagem ágil.

O público era formado por 172 participantes envolvidos em desenvolvimento de software e projetos de tecnologia da Informação com diferentes perfis. Algumas pessoas buscavam apenas aprofundar o conhecimento em métodos ágeis, enquanto outras possuíam nenhum ou pouco conhecimento.

Para coleta dos dados utilizou-se um questionário da ferramenta de conferência *web* que foi apresentando aos participantes durante a apresentação.

A avaliação dos resultados obtidos na pesquisa foi feita a partir da comparação da visão geral dos indivíduos obtida no questionário da ferramenta de conferência *web* com o que é apresentado na literatura, levantado em uma pesquisa bibliográfica.

4 Resultados e Discussão

Para a apresentação dos resultados, primeiramente são relacionados os principais pontos encontrados na pesquisa bibliográfica e em seguida são apresentados os resultados da pesquisa realizada que indica a visão geral das pessoas sobre métodos ágeis no desenvolvimento de um software ou projeto considerando os fatores entrega ao cliente, equipes de desenvolvimento e validação e verificação do software.

Sobre a entrega ao cliente, em geral, indica-se que ao final de um *sprint* deve-se fazer uma demonstração para o cliente com a funcionalidade implementada para que esta possa ser avaliada pelo cliente. É importante que ao final de cada *sprint* o cliente seja capaz de verificar se as funcionalidades entregues atendem suas necessidades e funcionam de acordo com o esperado (PRESSMAN, 2011). Uma das vantagens de se utilizar ágil é que se o software não estiver sendo desenvolvido de acordo com as necessidades do cliente ou do mercado, já é possível adotar mudanças de adequação, por isso é importante que o cliente tenha as funcionalidades planejadas entregues ao final de um *sprint*. Entregando as funcionalidades com todas as etapas de desenvolvimento já realizadas é possível verificar quais os déficits do software, assim como os benefícios de se utilizar o desenvolvimento seguindo o processo estabelecido pela equipe. Neste contexto é possível avaliar se o processo deve persistir, ser aprimorado ou até mesmo descontinuado.

Já na pesquisa realizada para este trabalho durante a apresentação não presencial sobre transformação ágil e customização de processos constatou-se, que a maioria das pessoas pesquisadas, isto é cinquenta e sete por cento, considera que ao final de uma *sprint* deve-se entregar um produto funcionando aos clientes, no entanto, quarenta e três

por cento dos participantes, acreditam que é possível entregar aos clientes apenas um protótipo.

Com relação a equipes de desenvolvimento, em geral, ressalta-se a importância dos fatores humanos no desenvolvimento de software utilizando métodos ágeis. Destaca-se a competência dos indivíduos para o desenvolvimento de software e de processos que a equipe escolheu para aplicação, a importância de todos estarem focados em objetivo comum, a colaboração dos participantes, a habilidade para tomada decisão e solução de problemas confusos, a capacidade de se auto-organizar, assim como, a confiança e o respeito entre os membros das equipes (PRESSMAN, 2011).

Desta forma, verifica-se que apesar dos indivíduos terem habilidades diferentes e cada um contribuir com o time de acordo com a função que exerce, é de extrema importância que todos estejam trabalhando em conjunto para definição do planejamento, resolução de problemas e entrega de um produto com qualidade.

Sobre as equipes em projeto ágeis, a pesquisa feita para este trabalho apontou que a maioria dos entrevistados compreende que é importante o trabalho com equipes integradas, nas quais todos são responsáveis pela entrega com qualidade do produto. Sessenta e cinco por cento dos participantes responderam que a equipe toda deve ser responsável pela entrega do produto, inclusive por testes, enquanto trinta e cinco por cento selecionaram que cada pessoa no time é responsável por entregas que cabem a sua função, isto é, sem inter-relacionar as atividades de funções diferentes.

A pesquisa realizada teve como principal interesse sondar os entrevistados acerca de sua percepção sobre a verificação e validação de software e os testes que eram mais importantes para entrega do produto ao cliente no final de uma *sprint* já que este é o foco do trabalho de mestrado em curso de um dos autores do trabalho.

É importante destacar que o *framework* de *scrum* não define um processo de verificação e validação uma vez que a equipe define qual o processo deve ser executado para desenvolver um requisito solicitado pelo cliente. Esta questão, no entanto, ajuda as equipes na definição de estratégias de testes dentro do desenvolvimento ágil com o framework de *Scrum*. Em uma pesquisa bibliográfica pode-se concluir que várias abordagens podem ser utilizadas.

Cavalcante, et al. (2011) relatam uma experiência da implementação de testes em ambientes ágeis de desenvolvimento utilizando *Scrum*. Entre as lições aprendidas na implementação, destaca-se a adoção de integração continua que forçou a implementação de testes unitários para todos os *build*s e automação de testes funcionais, que reduziu o tempo de execução de testes e possibilitou que as equipes pudessem focar na descoberta e conserto de defeitos.

Destacou-se também a cooperação de testadores e desenvolvedores na criação de testes automatizados, que fez com que fosse possível acomodar as mudanças no software com mais confiança, além de incentivar os testadores a contribuir com uma visão diferente sobre o planejamento e o projeto.

Um dos problemas enfrentados foi o gerenciamento de defeitos. No começo da implementação ágil, não se encontrava tempo para consertar e testar todos os defeitos dentro de uma estória dentro de uma *sprint*. Só com o amadurecimento do processo foi possível fazer com que todos os defeitos pudessem ser resolvidos dentro da *sprint*. Aqueles de menor impacto puderam ser adiados para *sprints* seguintes. Deve se ter cuidado especial com novos times para que se aloque tempo necessário para o teste, pois uma história só é considerada pronta depois de testada.

Farley e Humble (2011) destacam, no entanto, a importância de consertar e testar todos os defeitos a cada iteração sem que estes sejam postergados para as próximas. Desta forma diminui-se o risco de um defeito com baixa prioridade em uma sprint, ser postergado para as próximas *sprints* e impactá-las de maneira não esperada. Como

estratégias de teste para que isso seja possível, os autores sugerem o máximo de automação em testes unitários, de integração e até mesmo nos testes de aceitação de usuário.

Pham e Pham (2011) também abordam a importância de testes automatizados e testes de integração contínua, pois estes sustentam o objetivo da *scrum* que é entregar um produto ao final de cada *sprint*. Com testes automatizados, é possível fazer testes de regressão mais facilmente quando novas funcionalidades são implementadas e os testes de integração permitem verificar que a nova funcionalidade esta funcionando juntamente com o todo.

Delamaro, Maldonado e Vicente (2009), após um estudo sistemático sobre teste de software na metodologia ágil, verificaram que as estratégias mais utilizadas em ambientes ágeis foram TDD (*Test-driven development*) para testes de unidade e testes de aceitação de usuário. Também foi verificado que a atividade de teste é executada durante todo o ciclo de desenvolvimento e elas servem tanto para práticas de validação quanto para documentação do software. Entre as boas práticas, identificou-se a automação de testes como atividade recorrente no desenvolvimento ágil.

No levantamento bibliográfico apresentado constatou-se que os níveis de testes mais utilizados para verificação e validação do software foram os testes unitários e os testes de aceitação. Entre as práticas mais utilizadas constatou-se que a combinação de testes de regressão e automação é de extrema importância no desenvolvimento ágil e a equipe de desenvolvimento também foi um fator recorrente e determinante no processo de testes no desenvolvimento ágil.

Na pesquisa realizada junto aos entrevistados, ao perguntar-se sobre quais testes eram mais importantes para entrega do produto ao cliente no final de uma *sprint, v*erificou-se que os testes de aceitação de usuário, com escolha por sessenta e quatro dos atendentes do treinamento, foram os mais priorizados. Em seguida, com vinte e três por cento de priorização, foram definidos com maior importância os testes de integração, depois os de sistemas com oito por cento. Por último, com seis por cento de priorização os testes unitários selecionados.

4.1 Conclusão

Os resultados desta pesquisa permitiram identificar e analisar a visão geral de um grupo de pessoas relacionadas ao desenvolvimento de software e projetos de tecnologia da informação sobre métodos ágeis no desenvolvimento de um software ou um projeto considerando três fatores: entrega ao cliente, equipes de desenvolvimento e validação e verificação do software. Com a pesquisa, verificou-se que grande parte dos entrevistados não possuía total compreensão dos valores ágeis.

A partir da identificação e análise realizada é possível perceber qual o entendimento sobre métodos ágeis das pessoas envolvidas em desenvolvimento de software e projetos de tecnologia da Informação direcionando ações para discussão e aprimoramento da implementação dentro das organizações.

5 Referências

BECK, K. et al. *Manifesto for Agile Software Development***,** 2001**.** Disponível em: <http://Agilemanifesto.org>. Acesso em: 06/06/2014.

CAVALCANTE, A, M. et al. Testing in an Agile Product Development Environment: An Industry Experience Report. *IEEE***,** 2011

CRISPIN, L; GREGORY J. *Agile Testing: A Practical Guide for Testers and Agile Teams*. 1. ed. Boston: Addison Wesley, 2009.

DELAMARO, M, E; MALDONADO, J, C; VICENTE, A, A. Uma Revisão Sistemática sobre a atividade de Teste de Software no contexto de Métodos Ágeis. XXXV CONFERENCIA LATINO AMERICANA DE INFORMÁTICA. Pelotas: CLEI, 2009.

FARLEY, D; HUMBLE, J. *Continuous Delivery*. 1. ed. Boston: Addison Wesley, 2011.

HANSMANN, U; STOBER, T. *Agile Software Development: Best Practices for Large Software Development Projects*. 1. ed. New York: Springer, 2010.

ISTQB. International Software Testing Qualifications Board. *Foundation Level Syllabus*. 2011. Disponível em: <http://www.istqb.org/downloads/viewcategory/16.html> Aceso em: 15/11/2013

MOREIRA, M., E. *Being Agile: Your Roadmap to Successful Adoption of Agile***.** Apress, 2013.

PHAM, A.; PHAM, P. V. *Scrum in Action: Agile Software Project Management and Development***.** Boston: Cengage Learning. 2011

PRESSMAN, R. S. *Engenharia de Software: Uma abordagem Profissional*. 7 ed. Porto Alegre: AMGH, 2011.

RICO, D., F.; SAYANI H, H.; SONE, S. *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation***.** 1 ed. Fort Lauderdale J. Ross Publishing. 2009.

SCHWABER, K.; SUTHERLAND, J. *Scrum Guide*. 2013. Disponível em https://www.scrum.org/Scrum-Guide. Acesso em: 05/06/2014.

SCHWABER, K.; SUTHERLAND, J. *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors In the Dust*. New Jersey: Wiley Publishing, 2011

VERSION ONE. *Survey: The state of agile survey*. 2013. Disponível em http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf . Acesso em: 05/06/2014