Aplicando Testes Ágeis com Equipes Distribuídas: Um Relato de Experiência

Nayane Maia[1], Gisele Macedo[2], Eliane Collins[2], Arilo Dias-Neto[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6.200, Campus Universitário Senador Arthur Virgílio Filho – Setor Norte - Manaus - CEP 69.077-000 – Manaus – AM – Brasil

[2]Instituto Nokia de Tecnologia (INdT)
Manaus – AM – Brasil

{eliane.collins,gisele.macedo}@indt.org.br
{nayane.maia, arilo}@icomp.ufam.edu.br

Abstract. *Software testing is a highly collaborative and cooperative activity. With the growing of agile methodologies, these characteristics are becoming more and more necessary. However, this collaboration and cooperation are more difficult when the testing team is geographically separated (Distributed Software Development – DSD). In this paper, we report an industrial experience of a testing team separated geographically in the context of a software project that followed an agile methodology. We observed the feasibility and success of applying testing in an agile software project (agile testing) with distributed teams.*

Resumo. *Teste de software é uma atividade que requer colaboração e cooperação entre os envolvidos. Com o crescimento das metodologias ágeis, estas características estão se tornando cada vez mais necessárias. No entanto, esta colaboração e cooperação se tornam mais difíceis quando os membros da equipe de teste estão separados geograficamente (desenvolvimento distribuído de software – DDS). Neste artigo, é reportado um relato de experiência de uma equipe de teste distribuída no contexto de um projeto de software que adota metodologia ágil, tendo como observações a viabilidade e sucesso da aplicação de testes neste cenário.*

1. Introdução

Atualmente, Engenharia de Software tem se adaptado às evoluções que tem ocorrido no cenário de desenvolvimento de software global. Neste contexto, duas evoluções podem ser destacadas e são o foco deste artigo: métodos ágeis e desenvolvimento distribuído de software (DDS). Recentemente, estas abordagens têm sido combinadas, aplicando métodos ágeis em contextos geograficamente distribuídos [Paasivaara et al., 2008].

Analisando o cenário de métodos ágeis, eles são baseados nos quatro valores publicados no manifesto ágil [Beck et al., 2001]. Além disso, eles definem práticas de engenharia de software a serem seguidas visando o desenvolvimento de software com alta produtividade e qualidade. Os principais métodos ágeis aplicados na indústria são Scrum [Schwaber e Beedle, 2001], XP [Beck, 2000] e Crystal [Cockburn, 2004].

Analisando o cenário de DDS, pode ser observada a descentralização do desenvolvimento de software, distribuindo esforços entre equipes geograficamente distribuídas [Carmel e Prikladnicki, 2010]. Os benefícios atingidos pela adoção de DDS podem ser mensurados por meio da redução dos custos de desenvolvimento, aumento da efetividade do uso de fuso-horário entre equipes (sobreposição dos fusos) e

modularização do trabalho de desenvolvimento entre equipes e acesso a um amplo laboratório de habilidades (múltiplas equipes) [Conchúir et al., 2009]. Por outro lado, DDS introduz novos elementos que precisam ser atendidos para garantir o sucesso de um projeto de software, classificados, de acordo com [Herbsleb e Moitra, 2001], em: estratégicos, culturais, comunicação, gerência de conhecimento, gerenciamento de projeto e questões técnicas.

Ambos os cenários (métodos ágeis e DDS) descrevem teste de software como um elemento essencial para atingir a qualidade do produto. No entanto, alguns aspectos são tratados de forma diferente durante os testes nestes cenários (ex: processo, documentação e alocação de tarefas). No contexto de métodos ágeis, as atividades de teste de software são chamadas de testes ágeis [Crispin e Gregory, 2009].

Este artigo descreve um relato de experiência da aplicação de testes ágeis com equipes de teste geograficamente distribuídas. Serão reportadas as adaptações aplicadas no processo de testes de um projeto real para viabilizar a realização de testes distribuídos em um ambiente de desenvolvimento ágil (utilizando a metodologia Scrum), as soluções implementadas para reduzir o impacto de questões relacionadas à comunicação e alocação de tarefas, reuniões de acompanhamento do projeto e automação dos testes e integração contínua, características importantes para o funcionamento de métodos ágeis e que são impactadas pela separação física de parte da equipe de testes.

Este artigo está organizado da seguinte forma: Na seção 2 é discutida a aplicação de testes distribuídos em ambientes ágeis de desenvolvimento de software de acordo com a literatura técnica, contextualizando o cenário onde esta experiência foi conduzida. Na seção 3 é apresentada a aplicação de testes distribuídos em um projeto de software que seguiu uma metodologia ágil para seu desenvolvimento, suas adaptações e decisões. Na seção 4 são descritos algumas observações e lições aprendidas a partir desta experiência. Finalmente, na seção 5 são apresentadas as conclusões e trabalhos futuros.

2. Desenvolvimento Distribuído de Software e Testes Ágeis

Como descrito em [Crispin e Gregory, 2009], testes ágeis não apenas significam testes em projetos ágeis, mas testar uma aplicação com um plano para aprender sobre ela e deixar as informações dos clientes guiarem os testes, tudo respeitando os valores ágeis.

Automação de teste é considerada a chave para o desenvolvimento ágil e o elemento principal dos testes ágeis. Enquanto que testes tradicionais focam principalmente em testes manuais e exploratórios criticando o produto, mas não desempenhando um papel produtivo no processo apoiando a criação do produto. Além disso, seu foco é relevar falhas, e não preveni-las. Nos testes ágeis, a equipe não está envolvida apenas em identificar falhas, mas também preveni-las. Assim, teste ágil é um desafio para testadores acostumados à estratégia tradicional, principalmente porque eles não precisam esperar pela entrega do sistema para iniciar os testes, mas sim precisam ser proativos e iniciar os testes desde o início do projeto junto com os desenvolvedores.

Analisando a aplicação de teste de software em ambientes DDS, em [Shah et al., 2011] os autores descrevem um estudo revelando que teste de software é a segunda maior atividade terceirizada, após a codificação. Os resultados deste estudo sugeriram que para os engenheiros de testes manuais a fase de preparação foi considerada relativamente mais curta, mas a fase de execução seria consideravelmente mais longa. No entanto, para os engenheiros de teste automatizados, a fase de preparação seria mais longa, porque é necessário um esforço considerável para criação de *scripts* de teste automatizados, mas a fase de execução seria relativamente mais curta.

Em outro estudo [Sengupta et al., 2006], os autores observaram as atividades de teste em um projeto de software terceirizado. Os desafios relatados pelos autores foram: a) como realizar testes de unidade eficazes em um local remoto, onde dados de teste não podem ser acessados diretamente é inteiramente devido à privacidade de dados e problemas de replicação, b) como facilitar a integração de módulos desenvolvidos em locais separados e reduzir os erros de integração, e c) como testar um sistema de software utilizando documentos de interface imprecisos. Em [Causevic et al., 2010], os autores descreveram um estudo onde pesquisas relacionadas ao desenvolvimento de software estão de acordo com cinco aspectos: 1) Desenvolvimento Ágil de Software, 2) Desenvolvimento de Software Distribuído, 3) Domínio do produto; 4) Segurança de Criticalidade do produto; 5) Quantidade de testes executados por participante. Como resultados, os autores concluíram:

- Os participantes que trabalham com desenvolvimento ágil de software não estão felizes com a prática usual de *test-first* (provavelmente porque eles começam a utilizar este método);
- Participantes "não-ágeis" não conhecem as práticas de testes ágeis;
- A ausência de infraestrutura é considerada um fator importante para retardar a implantação de software neste ambiente.

Além de caracterizar o cenário de teste em DDS, estes trabalhos estão relacionados com a situação em que o teste é realizado em um ambiente em que a equipe de desenvolvimento está distribuída e o software produzido precisa ser testado. No entanto, eles não cobrem o cenário onde o desenvolvimento é realizado em um mesmo local e as atividades de teste são distribuídas geograficamente entre diferentes equipes. Neste relato, será descrito o cenário onde testes sao realizados usando duas equipes separadas geograficamente, o que introduziu novos desafios e resultados, conforme descrito na próxima seção.

3. Aplicando Testes Ágeis Distribuídos em Projetos de Software

O projeto de software a ser descrito neste relato foi conduzido no contexto de um instituto de pesquisa sediado em Manaus-AM. Seu objetivo foi desenvolver uma aplicação web para gerenciar campanhas publicitárias e suas propagandas, chamada NAP (*Nokia Advertisement Platform*). Para este projeto, adotou-se a metodologia ágil Scrum [Schwaber e Beedle, 2001], e seu *Product Backlog* foi composto por 30 estórias (que correspondem a 8 telas funcionais) divididas em 9 *sprints* (iterações) de 15 dias. Sua equipe de desenvolvimento foi formada por 1 *Scrum Master* e 3 desenvolvedores. Sua equipe de teste, foco deste relato, foi formada por 6 profissionais, sendo 2 atuando em tempo integral no mesmo ambiente físico dos desenvolvedores (equipe de teste local) e 4 atuando em meio período em outro ambiente físico separado geograficamente (equipe de teste remota), resultado de um projeto de cooperação entre o instituto e a Universidade Federal do Amazonas.

3.1. Processo de Testes

O processo de teste corrente nos projetos da empresa acompanhava a metodologia Scrum, onde a equipe de teste atuava como time do projeto, participando das cerimônias do Scrum (*Sprint Review*, *Retrospective* e *Planning*). Cada integrante do time de teste tinha a responsabilidade de especificar casos de testes através das estórias descritas no *Product Backlog* e especificar critérios de aceitação, além de utilizar ferramentas de automação de testes para agilizar a execução dos testes a cada *sprint*. Como a equipe de teste estava dividida em 2 ambientes, o processo de teste precisou ser

adaptado para atender às características de um ambiente DDS, de forma que a distribuição geográfica da equipe de testes não interferisse no andamento das atividades de teste do projeto.

Para contornar a distância de parte da equipe de testes, foi necessário fornecer uma infraestrutura de testes que permitisse o acesso às informações do projeto por todos os envolvidos de forma eficiente, composta por:

- **Ferramenta de gestão de tarefas Scrum (*FireScrum*):** utilizada por ambas as equipes de teste para registrar e atualizar o andamento de tarefas definidas para cada *sprint* do projeto.
- **Ferramenta de gestão de teste (*TestLink*):** utilizada para organizar os casos de teste definidos manualmente (de forma externa à ferramenta) para cada estória, provendo acesso a estes artefatos para as equipes local e remota.
- **Ferramenta de Gestão de Defeitos (*Mantis*)**: utilizada para que testadores locais e remotos pudessem gerenciar incidentes identificados durante os testes e atribui-los a desenvolvedores para correção.
- **Ferramenta de Versionamento (*Subversion*):** possibilitou a criação de um repositório único compartilhado entre as equipes para armazenar informações sobre o projeto.
- **Servidor de aplicação para testes:** servidor no qual versões (releases) do projeto eram disponibilizadas para teste com acesso ao banco de dados.
- **Ferramentas de comunicação**: foi criado um email do projeto para todos os membros do projeto para comunicação segura entre as equipes.

Para garantir o funcionamento desta infraestrutura e facilitar a comunicação entre as equipes (teste local, teste remota e desenvolvimento), a equipe de teste contava com um profissional que exerceu o papel de Líder de Teste e ficou locado no ambiente físico do projeto. Suas responsabilidades podem ser definidas da seguinte forma:

- Prover acesso à infraestrutura necessária para a execução das tarefas de teste;
- Planejar e orientar os envolvidos nas atividades de teste de cada *sprint*;
- Reportar o andamento das atividades de teste na reunião diária (*daily meeting*) do projeto Scrum, substituindo os membros da equipe remota;
- Elaborar o relatório de execução de teste no fim de cada *sprint*.

Nessa infraestrutura, apenas o líder de teste pode se comunicar com a equipe de desenvolvimento, evitando uma grande rede de comunicação na equipe de testes. Os demais membros, tanto da equipe local como remota, eram responsáveis por tarefas como: criar os casos de teste, automatizar e executar casos de teste para as estórias desenvolvidas em cada *sprint*. A divisão de responsabilidades e canais de comunicação das equipes estão descritos na Figura 1.

O ciclo reiniciava para as próximas funcionalidades em ordem de prioridade. Ao chegar ao fim da *sprint*, o relatório sumarizado com todos os defeitos encontrados era gerado e enviado para a equipe. Este processo está descrito com mais detalhes, suas atividades, papéis, produtos e responsabilidades em [Collins et al., 2012].

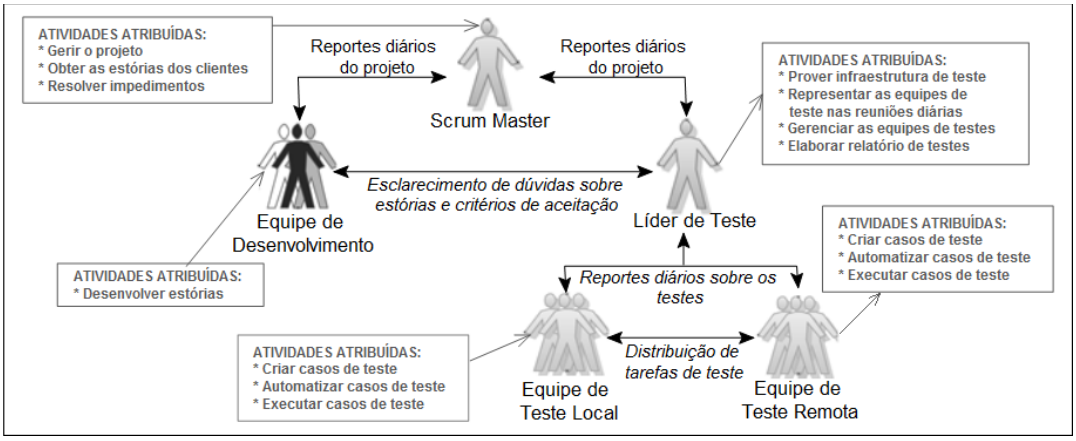

Figura 1. Distribuição de Responsabilidades e Canais de Comunicação

3.2. Cerimônias Scrum

A metodologia Scrum define cerimônias como sendo importantes para o planejamento e controle do andamento do projeto: reuniões de planejamento para definição do escopo das *sprints*, reuniões diárias para acompanhamento das evoluções no projeto e reunião de retrospectiva para discutir pontos positivos e de melhorias no projeto.

Na versão tradicional do Scrum, essas cerimônias devem ser presenciais com a participação de todos do projeto. No entanto, com parte da equipe de testes distribuída geograficamente, atender a essa premissa se tornou inviável. Em um primeiro momento, optou-se por ter apenas o líder de teste representando as equipes de teste local e remota nas cerimônias. Isso dificultou o andamento dos testes ágeis, uma vez que por não participarem das reuniões, a equipe remota não participava das estimativas de prazo e definição do escopo dos testes, o que impactava no entendimento das estórias e no comprometimento da equipe com as tarefas alocadas.

Ao perceber isso, optou-se pela participação de todos os envolvidos no projeto de forma presencial nas reuniões de planejamento e retrospectiva (a equipe remota se deslocava à sede do projeto). Com isso, a apresentação das estórias e suas estimativas eram discutidas entre todos os envolvidos. Para as reuniões diárias, estas passaram a ser realizadas por meio de videoconferências e utilizando uma ferramenta de *dashboard* para Scrum, onde os membros da equipe remota reportam suas atividades, impedimentos e avanços nas tarefas.

3.3. Automação dos Testes e Integração Contínua

As tarefas do processo de teste durante cada *sprint* eram incrementais e iterativas, seguindo o processo de Scrum. A tarefa de execução de teste incremental incluiu a execução de testes automatizados. Para o projeto utilizado, a ferramenta escolhida para a automação de testes funcionais foi o Selenium RC. Esta ferramenta utiliza a abordagem de apoio *capture-replay* em testes de aplicações web.

A implantação de uma estratégia de integração contínua quando integrada a uma estratégia de automação dos testes neste projeto apresentou ótimos resultados. Assim, à medida que desenvolvedores integravam novas funcionalidades/módulos do sistema, testes já especificados eram executados visando avaliar a qualidade do novo sistema com um esforço bastante reduzido e o conjunto completo de casos de testes eram executados, o que seria difícil de ser atingido por equipes distribuídas sem a adoção destas práticas.

Na próxima seção são descritos resultados e lições aprendidas com estas práticas.

4. Observações e Lições Aprendidas

Nesta experiência de conduzir testes ágeis com equipes geograficamente distribuídas, percebeu-se que esta seria viável e com resultados bastante positivos. No entanto, alguns aspectos precisam ser gerenciados para evitar os riscos introduzidos pela combinação de duas práticas de engenharia de software que apesar de integráveis, possuem aspectos divergentes. Assim, a seguir algumas lições chaves que foram aprendidas para minimizar o impacto da distância geográfica entre equipes de teste:

4.1. Decisões impactantes devem ser tomadas com a presença de todos

A presença física dos membros da equipe de teste remota em reuniões de planejamento e retrospectiva aumentou sua motivação, pois estes se sentiam participando de forma mais ativa do projeto, premissa esta valorizada em metodologias ágeis. Com o envolvimento de todos do projeto nas tomadas de decisões (estimativas de prazo e alocação de tarefas), o cumprimento ao cronograma definido para a atividade de teste passou a ser constante, pois estas passaram a ser decididas com a perspectiva de todos aqueles influenciados pela tarefa.

Como consequência dessa participação, a precisão nas estimativas nas tarefas de teste teve uma evolução, chegando a 100% em diversas *sprints*. Essa evolução também é resultado do amadurecimento do time com relação ao projeto.

4.2. Organização e coordenação são fatores essenciais em testes ágeis distribuídos

Um dos princípios básicos do manifesto ágil é que a organização e coordenação da equipe ágil é mais importante que a definição de processos, pois apenas a definição não garante que estes estejam sendo seguidos. Da mesma forma, pesquisas na área de DDS indicam coordenação como aspecto importante para se obter sucesso em um projeto distribuído, e isso pôde ser confirmado nesta experiência. A implantação de algumas práticas foi essencial para o sucesso do projeto:

- Alocação de uma pessoa (líder de teste – coordenador) como um link entre as equipes de teste local e remota foi muito importante para evitar uma rede de comunicação ampla e, consequentemente, ruídos. Assim, todas as informações e decisões passam sempre por este profissional, que é responsável por distribuí-las, resolver impedimentos e problemas de comunicação, deixando a equipe de teste livre apenas para realizar suas tarefas técnicas.
- Formalização de um protocolo de comunicação, regulamentando como as equipes podem se comunicar. Isso inclui a estruturação de emails, definição de ferramentas de comunicação (chats, videoconferências), reuniões periódicas. Isto é essencial para evitar perda de dados, esforço e qualidade no projeto.
- Alocação tempo integral de pessoas chaves do projeto (*Scrum master*, líderes de desenvolvimento e teste), ficando sempre disponíveis para esclarecer dúvidas. Nesta experiência, uma ferramenta de chat online foi usada para este propósito.
- Agendamento antecipado das reuniões entre as equipes de desenvolvimento e testes, permitindo o planejamento e a participação integral dos envolvidos.

4.3. A dependência entre as equipes distribuídas deve ser reduzida

Apesar da alocação de um líder de teste na equipe local que seria o responsável por atender às demandas da equipe remota, percebeu-se que a ausência de autonomia prejudicava o andamento das tarefas da equipe remota. Isto demandava muito esforço

do líder de teste, que tinha outras responsabilidades dentro do contexto do projeto (projeto dos testes, controle da equipe local, preparação da infraestrutura de testes). Assim, soluções foram aplicadas, com resultados positivos, minimizando esta dependência:

- A padronização da especificação de casos de teste e relato de defeitos permitiu que um testador completasse tarefas iniciadas por outro profissional (algo comum em ambientes distribuídos) ou validasse um defeito reportado por outro testador sem a intervenção do líder de teste.
- Implantação de uma ferramenta de gerenciamento dos testes para controlar as atividades, especificação e validação de casos de teste, criação de novas *releases* a serem testadas, execução dos testes e rastreamento dos defeitos reportados.
- Alocação de um repositório do projeto, onde todas suas informações eram disponibilizadas a todos os envolvidos (ex: estórias do usuário e seus detalhamentos, critérios de aceitação, artefatos de teste).

4.4. Automação dos testes e integração contínua são práticas necessárias para redução de esforço ocasionado pela distribuição da equipe

Automação de testes possuiu um papel essencial para o sucesso de testes ágeis distribuídos em diversos pontos. Primeiramente, facilitou a comunicação dentro da equipe de teste pela adoção de uma notação única (uma linguagem de programação ou uma ferramenta) para relato dos casos e resultados de teste. Outro ponto, isso facilitou a execução dos testes a qualquer momento sem depender da sobreposição de horário entre as equipes local e remota, pois ambas as equipes tinham acesso à execução do conjunto de testes de regressão, o que acelerou, em diversos momentos, o tempo necessário para execução dos testes, mesmo quando novas *releases* eram liberadas pela equipe de desenvolvimento próximas à data de entrega do software.

À medida que a taxa de automação dos testes foi aumentando no projeto, o esforço de teste e a necessidade de comunicação entre as equipes foram diminuindo. Neste projeto, com a junção das duas práticas (automação de testes e integração contínua) chegou-se a uma taxa de automação dos testes (considerando todos os níveis) em torno de 50%. Ao final do projeto, ambas as equipes, desenvolvedores e testadores, utilizam apenas a ferramenta de gestão de defeitos como mecanismo de comunicação. Assim, o processo de relato, correção e validação de defeitos se tornou bastante ágil, o que contribuiu para o sucesso do projeto.

5. Conclusões

Este artigo apresentou um relato de experiência sobre a realização de testes ágeis com equipes distribuídas geograficamente. Os resultados desta experiência indicam a viabilidade em integrar práticas de testes ágeis com DDS. Este sucesso depende de fatores como: organização e coordenação dos testes (aspecto importante em ambos os cenários), criação de uma infraestrutura que possibilite implantar as práticas ágeis independentemente da distancia geográfica entre as equipes, implantação de uma estratégia de automação dos testes e integração contínua. Soluções tecnológicas contribuem significativamente para o sucesso da aplicação de testes ágeis com equipes distribuídas. No entanto, a organização e comprometimento da equipe são fatores essenciais para atingir este objetivo.

Algumas lições aprendidas foram extraídas desta experiência e elas podem contribuir para minimizar o impacto da aplicação de testes ágeis distribuídos em outros projetos. Como próximo passo, está sendo avaliada a efetividade das atividades de teste ágil distribuído em comparação ao cenário anterior aplicado no instituto, onde testes eram realizados por uma única equipe local. Métricas de esforço de teste, número de falhas detectadas pré e pós entrega e aderência ao cronograma estão sendo analisadas.

Agradecimentos

Os autores agradecem à FAPEAM, CNPq (processo 575696/2008-7), INCT-SEC (processos 573963/2008-8 e 08/57870-9) e INdT pelo apoio para a realização desta pesquisa.

Referências Bibliográficas

Beck K. (2000). Extreme Programming Explained: Embrace Change, Addison Wesley, 2000.

Beck, K.; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved February 2012.

Carmel, E.; Prikladnicki, R.; (2010). Does Time Zone Proximity Matter for Brazil? A Study of the Brazilian I.T. Industry (July 22, 2010). Available at SSRN: http://dx.doi.org/10.2139/ssrn.1647305.

Causevic, A., Sundmark, D., & Punnekkat, S. (2010). An Industrial Survey on Contemporary Aspects of Software Testing. 2010 Third International Conference on Software Testing, Verification and Validation, 393-401. IEEE. doi:10.1109/ICST.2010.52.

Cockburn, A.; (2004). Crystal Clear a Human-Powered Methodology for Small Teams (First ed.), Addison-Wesley Professional, 2004.

Collins, E. F.; Lobão, L. M. A.; Lucena Jr, V. F. (2011). Experiência em Aplicação de Processo de Teste de Software Iterativo e Automático. In: Brazilian Workshop on Systematic and Automated Software Testes, São Paulo, v. 1. p. 1-6.

Conchúir, E.; Ågerfalk, P.; Olsson, H.; Fitzgerald, B; (2009) Global software development: where are the benefits?. Commun. ACM 52, 8 (August 2009), 127-131. DOI=10.1145/1536616.1536648.

Crispin, L.; Gregory, J.; (2009). Agile Testing: A Practical Guide for Testers and Agile Teams, Addison-Wesley, 2009, ISBN 0-321-53446-8.

Herbsleb, J.; Moitra, D.; (2001). Global Software Development. IEEE Software, v.18(2).

Paasivaara, M.; Durasiewicz, S.; Lassenius, C.; (2008). Using scrum in a globally distributed project: a case study. Software Process: Improvement and Practice, 13, 527–544, 2008.

Schwaber K, Beedle M; (2011). Agile software development with scrum[M]. New Jersey:Prentice Hall,2001, pp. 29–51.

Sengupta, B.; Chandra, S.; Sinha. V.; (2006). A research agenda for distributed software development. In: 28th International Conference on Software Engineering (ICSE '06). ACM, New York, NY, USA, 731-740, 2006. DOI=10.1145/1134285.1134402.

Shah, H.; Sinha, S.; Harrold, M.J.; (2011). Outsourced, Offshored Software-Testing Practice: Vendor-Side Experiences. In: International Conference on Global Software Engineering, Finlândia, pp. 131-140.