Capítulo 3

Processos para Desenvolvimento Distribuído de Software

Camila Cunha Borges[1]

Este capítulo discute como os modelos de processos e práticas de desenvolvimento de software podem ser aplicados em um ambiente de desenvolvimento distribuído de software. Aqui serão apresentados, de forma prática, os desafios e casos de sucesso de projetos de desenvolvimento distribuído de software.

3.1 Introdução

Nas últimas décadas temos observado que o software passou a ser o elemento-chave da evolução de sistemas e produtos baseados em computador [PRESSMAN, 2005], tornando-se necessário desenvolver software com rapidez e qualidade. É importante observar que, à medida que o tempo passa, a forma de se desenvolver um software vem passando por mudanças e os problemas relativos ao desenvolvimento continuam os mesmos: usuários insatisfeitos, longo tempo de desenvolvimento, nível de qualidade baixa, dificuldade de comunicação, etc.

Com a globalização dos negócios, surgem grandes desafios para o processo desenvolvimento de software, que tornou-se cada vez mais distribuído e global [PRIKLADNICKI & AUDY, 2008]. Sendo assim, observamos a necessidade da implantação de métodos e ferramentas para a melhoria do processo de desenvolvimento distribuído de software.

3.2 Desenvolvimento Distribuído de Software

Grandes investimentos têm permitido uma movimentação do mercado local para o global. Neste contexto, muitas organizações encontraram no Desenvolvimento Distribuído de Software (DDS) uma chance para obter sucesso nos negócios.

1 camila.cunha@gmail.com

Segundo [PRIKLADNICKI & AUDY, 2008], o DDS ganha cada vez mais força, motivado por três fatores ligados ao ambiente de negócios: (1) a globalização; (2) o crescimento da importância dos sistemas de informação nas empresas; (3) os processos de terceirização que geram um ambiente propício a esse cenário de desenvolvimento.

CARMEL [1999] afirma que as principais características que diferenciam o desenvolvimento co-localizado (desenvolvimento no mesmo espaço físico) do desenvolvimento distribuído são: distância, diferenças de fuso horário e diferenças culturais. Distância refere-se à distribuição geográfica dos desenvolvedores e clientes finais; e diferenças culturais referem-se a idioma, tradições, costumes e comportamento.

De acordo com PRIKLADNICKI [2003], o desenvolvimento distribuído criou uma nova classe de problemas a serem resolvidos pelos pesquisadores na área de desenvolvimento de software. Esta nova forma de desenvolver software impacta não apenas no mercado propriamente dito, mas também na maneira como os produtos são criados, modelados, construídos, testados e entregues aos clientes.

3.2.1 Motivações para o DDS

O desenvolvimento de software, realizado por pessoas com alto grau de especialização, trabalhando em centros de processamento de dados (CPDs) em países avançados, vem ocorrendo de uma forma cada vez mais distribuída [PRIKLADINIKI, 2003].

As organizações visam obter vantagens competitivas associadas ao custo, qualidade e flexibilidade no desenvolvimento de software. Na maioria dos casos esse processo ocorre no mesmo país ou em regiões com incentivos fiscais Algumas empresas buscam soluções em outros países (soluções globais), assim obtendo maiores vantagens competitivas.

Segundo a *International Data Group* [2006] apud [NAVARRO et. al., 2007], pode-se ter uma economia entre 25% e 50% em termos de custo quando grandes projetos são transferidos para operações *offshore* (em outro país). Incentivos de governos locais têm contribuído para essa nova forma de desenvolver software. Neste contexto, existem diversas razões para a aplicação do DDS, entre as quais podem ser citadas:

- **Demanda e custo:** Com o aumento na demanda por profissionais de software, o custo da mão-de-obra sofreu um aumento conforme as organizações competiam por suas contratações [KAROLAK, 1998]. Assim a disponibilidade de recursos equivalentes em outras localidades tornou-se um grande atrativo.
- **Rapidez de resposta ao mercado:** A possibilidade de um desenvolvimento *follow-the-sun*, possibilitando 24 horas contínuas de trabalho, é um grande atrativo para empresas que visam reduzir o *time-to-market* (tempo para colocar o produto no mercado).
- **Mercado e presença global:** À medida que os custos são reduzidos e há um aumento no poder computacional, o mercado global de informática cresce. Assim o DDS é uma opção para atender a demanda do mercado global.

3.2.2 Níveis de Dispersão

O nível de dispersão dos atores envolvidos no processo de desenvolvimento é uma característica importante do DDS. O entendimento dos níveis de dispersão auxilia na identificação de possíveis dificuldades. Se existe uma distância de 30 metros ou mais entre os colaboradores, a frequência da comunicação diminui na mesma proporção que em uma distância de milhares de metros [HERBSLEB, 2001a & HERBSLEB, 2001b]. É importante entender o nível de distâncias e suas implicações para as equipes. A seguir apresentamos tipos de distância física (Figura 3.1) e suas principais características:

- **Mesma Localização Física:** A empresa possui todos os atores em um mesmo lugar. Nesta situação, não existem dificuldades de reuniões e há uma interação constante entre membros das equipes. Além disso, não há diferença de fuso-horário e/ou diferenças culturais. Neste cenário as dificuldades são as já existentes no desenvolvimento centralizado.
- **Distância Nacional:** Os grupos de atores estão localizados em um mesmo país, podendo reunir-se em curtos intervalos de tempo. Em alguns países podem ocorrer diferenças no fuso-horário.
- **Distância Continental:** As equipes de atores estão localizadas em países diferentes, necessariamente no mesmo continente. As reuniões são mais difíceis de acontecerem face a face. Além disso, o fuso-horário dificulta as interações entre os membros das equipes.
- **Distância Global:** Os grupos de atores estão em países e continentes diferentes. A comunicação e diferenças culturais neste cenário podem ser barreiras para o andamento do projeto e as reuniões face a face geralmente acontecem no inicio do projeto.

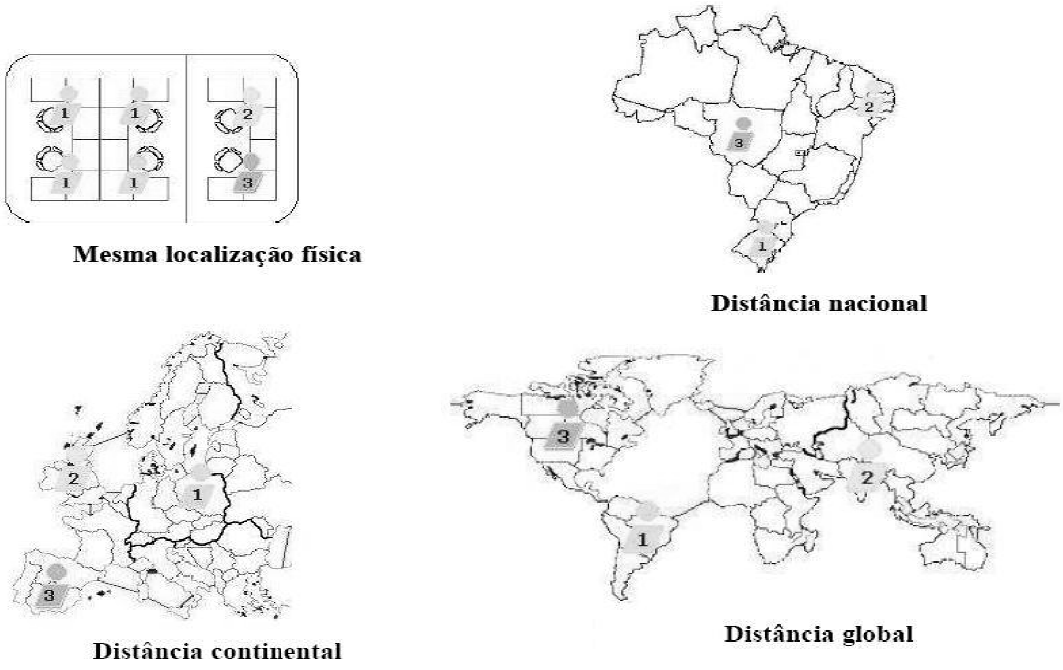


Figura 3. 1 – Tipos de Distância Física, adaptado de [PRIKLADINIKI, 2007]

3.2.3 Modelos de Negócio

De acordo com [PRIKLADNICKI & AUDY, 2008], entre as relações existentes entre clientes e provedores de serviço, podemos caracterizar *Outsourcing* (desenvolvimento feito por empresas terceirizadas) e *Insourcing* (desenvolvimento feito por subsidiárias da mesma empresa) como as principais relações existentes. Quanto à dimensão relacionada com a distância geográfica, a distribuição pode ser *Onshore* (em um país diferente) ou *Offshore* (no mesmo país). A seguir, os vários tipos de modelos de negócio são definidos:

- ***Onshore Insourcing*:** Existe um departamento dentro da própria empresa ou uma subsidiária da empresa no mesmo país (*onshore*) que provê serviços de desenvolvimento de software através de projetos internos (*insourcing*).
- ***Onshore Outsourcing* ou *Outsourcing*:** É a contratação de uma empresa terceirizada (*Outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software. A empresa terceirizada está localizada no mesmo país da empresa contratante (*onshore*).
- ***Offshore Outsourcing* ou *Offshoring*:** É a contratação de uma empresa terceirizada (*Outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software, sendo que a contratada está localizada em um país diferente da contratante (*offshoring*).
- ***Offshore Insourcing* ou *Internal Offshoring*:** É a criação de uma subsidiária da empresa para prover serviços de desenvolvimento de software (*Insourcing*) em um país diferente da empresa contratante (*offshore*).

É importante que seja observado que, além de outras formas de relacionamento entre empresas, também podem surgir outros tipos de distribuição geográfica, resultando em outros tipos de modelos de negócio.

3.2.4 Desafios

Conforme apresentado anteriormente, o DDS apresenta níveis de dispersão física, distância temporal e diferenças culturais, com isso alguns desafios foram acrescentados ao processo de desenvolvimento. O ambiente global apresenta grande impacto na forma como os produtos são concebidos, desenvolvidos, testados e entregue aos clientes [PRIKLADNICKI & AUDY, 2008]. Diferentes tecnologias e características são necessárias para o suporte ao DDS. Entre muitos desafios relacionados ao DDS, nesta seção vamos detalhar desafios focados no processo de desenvolvimento.

- **Arquitetura do Software:** Conforme KAROLAK [1998], uma arquitetura apropriada é um dos fatores mais utilizados para a diminuição do esforço entre as equipes para o DDS e esta deve se basear no princípio da modularidade, pois permite alocar tarefas complexas de forma distribuída. Com isso há uma redução na complexidade e é permitido um desenvolvimento em paralelo simplificado.

- **Engenharia de Requisitos:** A engenharia de requisitos contém diversas tarefas que necessitam de alto nível de comunicação e coordenação. Com isso os problemas apresentados são mais complexos em um contexto de DDS.

- **Gerência de Configuração:** O gerenciamento de configuração (GC) é a chave para controlar as múltiplas peças em um projeto distribuído. Controlar modificações nos artefatos em cada uma das localidades que participam do processo de desenvolvimento de um produto pode ser complexo. Apesar da utilização de práticas de GC auxiliarem no controle da documentação e do software, a gerência de modificações simultâneas a partir de locais diferentes é um grande desafio. Além disso, o uso de ferramentas de GC compartilhadas por duas ou mais equipes de forma inadequada gera diversos riscos e problemas em projetos DDS.
- **Processo de Desenvolvimento:** Em projetos DDS, o uso de uma metodologia[2] que auxilia a sincronização e a padronização das atividades é essencial. Isto permite que todos os membros utilizem uma nomenclatura comum em suas atividades.
- **Testes e Garantia da Qualidade:** As práticas de teste e de garantia da qualidade do processo de desenvolvimento têm apresentado grande importância em projetos de software. A globalização do processo de desenvolvimento de software tem destacado a importância da adoção de modelos de qualidade internacionalmente aceitos, tais como ISO 9001 e CMMI que serão abordados no segundo módulo deste livro, chamando a atenção para a adoção de padrões visando reduzir os problemas existentes na área de Engenharia de Software.

3.3 Processos para Desenvolvimento Distribuído de Software

Como visto anteriormente, em um ambiente DDS, um processo de desenvolvimento padrão é fundamental, fornecendo aos membros da equipe uma nomenclatura comum para tarefas e atividades, e um conjunto comum de expectativas relacionadas aos elementos envolvidos no processo [PRIKLADNICKI & AUDY, 2008].

A Engenharia de software (ES) está sempre apresentando grandes avanços e transformações relacionadas às técnicas, modelos e metodologias. Esses avanços são destacados quando se trabalha com processo de Desenvolvimento Distribuído de Software, havendo a necessidade do uso de práticas que dêem suporte às dificuldades encontradas nas definições de requisitos que mudam de forma dinâmica no decorrer do tempo.

A forma como um produto de software é concebido, desenvolvido, testado e entregue ao cliente sofre grande impacto quando o ambiente de desenvolvimento é distribuído [HERBSLEB, 2001a & HERBSLEB, 2001b]. Assim, a estrutura necessária para o suporte desse tipo de desenvolvimento se diferencia da utilizada em ambientes centralizados. Diferentes características e tecnologias se fazem necessárias, crescendo a importância de alguns detalhes anteriormente não percebidos.

Estratégias, soluções e práticas para tornar esta abordagem um sucesso tornam-se imperativas. O desenvolvimento de ambientes, modelos e ferramentas para gerenciar processos de software neste contexto tornam-se cada vez mais importantes.

[2] Os termos metodologia e processo estão sendo usados como sinônimos.

Neste contexto, são apresentados a seguir, alguns processos específicos para o desenvolvimento distribuído, criados a partir de adaptações de processos voltados para o desenvolvimento co-localizado.

3.3.1 Modelo de Karolak

Karolak [1998] aborda o DDS seguindo o ciclo de vida tradicional de um projeto de desenvolvimento de software. O autor propõe um modelo para desenvolver projetos DDS abrangendo as atividades que devem ocorrer ao longo do ciclo de vida. A Figura 3.2 ilustra o modelo proposto:

Figura 3. 2 – Modelo para projetos DDS, adaptado de [KAROLAK, 1998]

A seguir são descritas as atividades do modelo proposto:

- **Alinhar o negócio:** Nesta atividade é definido se existirão interações com outras empresas ou se serão criadas unidades da empresa em outras localidades.
- **Identificar a equipe distribuída:** Nesta atividade são definidos os integrantes da equipe, seus respectivos papéis e responsabilidades. A formação da equipe deve considerar os seguintes aspectos: aquisição de confiança, diferenças culturais e relacionamento.
- **Identificar as tecnologias:** Devido à grande demanda de comunicação em projetos DDS, há a necessidade de um apoio tecnológico considerável. Nesta atividade é identificada a infraestrutura disponível para os membros das equipes se comunicarem, considerando o nível de dispersão destas equipes.
- **Definir o contrato:** Um contrato é um documento que define o escopo do que deve ser feito. Quando o projeto é distribuído esta atividade se torna mais complexa.
- **Dividir o trabalho:** Após a identificação da equipe, tecnologia e definição do contrato, é proposta a divisão do esforço de trabalho entre os membros de uma equipe. Deve ser levado em consideração o nível de experiência e a modularidade do projeto.
- **Identificar ferramentas e métodos:** Identificação dos recursos técnicos que serão utilizados na modelagem e implementação do projeto. Deve-se considerar o nível de dispersão da equipe e o processo de desenvolvimento.

- **Estabelecer responsável por SCM:** A gerência de configuração de software (SCM – Software *Configuration Management*) tem como objetivo controlar modificações nos artefatos, dando suporte ao controle de versões. O autor sugere a existência de um grupo responsável pelo controle de configuração e versões do sistema. Por este motivo, esta atividade visa identificar os membros deste grupo, bem como as ferramentas que eles utilizarão e a frequência necessária de reuniões para discutir o andamento do trabalho.
- **Identificar e gerenciar riscos:** Esta atividade faz parte de qualquer projeto e deve acontecer em todas as fases do desenvolvimento. De acordo com o autor, os riscos em projetos DDS tendem a ser mais centrados em aspectos não tão visíveis. Em projetos DDS podem existir três categorias de risco: organizacional, técnico e de comunicação.
- **Controlar a documentação:** É conhecida a resistência em documentar por partes de equipes de desenvolvimento. Em projetos DDS, uma documentação pobre pode causar ineficiência na colaboração. Uma boa documentação pode evitar ambiguidades e facilitar futuras manutenções.
- **Desenvolver plano e casos de teste:** KAROLAK [1998] menciona que um projeto distribuído necessita de pelo menos dois artefatos de teste: o plano de teste com as estratégias, métodos e ambiente de testes documentados; e a descrição dos casos de teste[3] para as funcionalidades que serão testadas.
- **Criar matriz de rastreabilidade:** uma matriz de rastreabilidade é um artefato que identifica as funcionalidades do projeto e os módulos que as implementam. Este artefato tem o objetivo de mostrar a ligação entre os requisitos e como um requisito influencia em outro.
- **Criar matriz de versão de módulos:** uma matriz de versão de módulos é um artefato que identifica qual versão de um módulo foi utilizada na compilação do código de um projeto. Este artefato é essencial principalmente para a coordenação das atividades e divisão do trabalho entre os membros da equipe do projeto.
- **Criar grupo de manutenção:** O modelo sugere a criação de um grupo responsável por revisar solicitações de mudanças após o produto ser entregue ao cliente.
- **Controlar a qualidade do software:** Devem existir atividades que melhoram a qualidade do software a ser desenvolvido, tais como revisões de modelagem, inspeções de código e teste.
- **Gerenciar a propriedade intelectual**: O autor prevê uma atividade onde se busca a devida proteção da propriedade intelectual, levando-se em consideração leis e restrições do local onde o projeto foi desenvolvido (alguns locais fisicamente dispersos podem ter leis muitas vezes desconhecidas pelas organizações).

[3] Cenários de testes, descrevendo o resultado esperado para cada entrada possível.

3.3.2 Uso de Práticas Ágeis

Como visto no capítulo 2, atualmente há uma tendência para o desenvolvimento ágil de aplicações, devido a um ritmo acelerado de mudanças e inovações na tecnologia da informação, em organizações e no ambiente de negócios.

Quando o ambiente de desenvolvimento é distribuído, o uso de práticas ágeis parece ser incompatível, pois estas práticas necessitam de comunicação face a face constantemente, e a comunicação é um grande desafio em ambientes DDS. Apesar disso, o uso de metodologias ágeis de desenvolvimento de software tem se tornado uma demanda em equipes distribuídas de software devido à necessidade de aumento na velocidade de desenvolvimento, ao alinhamento dos objetivos individuais com os organizacionais e à melhoria no desenvolvimento [SUTHERLAND, 2007].

A seguir são apresentadas algumas propostas de adaptações de metodologias ágeis aplicadas ao contexto de ambientes DDS.

DXP – *Distributed eXtreme Programming*

De acordo com KIRCHER [2001 & 2008], a proximidade entre Metodologias Ágeis e DDS ocorre devido a algumas características comuns, como *feedback* contínuo, *releases* frequentes, valorização da comunicação e padrões de codificação. Neste contexto o autor propõe uma adaptação do uso do XP para ambientes DDS, o *Distributed eXtreme Programming* (DXP). Estudos mostram que a aplicação do DXP pode ser eficaz e gratificante em projetos cujas equipes estão geograficamente dispersas. A Tabela 3.1 resume alguns dos aspectos de XP que são relevantes para DXP e que não são impactados devido à distribuição das equipes.

Tabela 3. 1 – Adaptação do XP para ambientes DDS [KIRCHER, 2001]

Práticas do XP	É necessário a equipe ser co-localizada?
Jogo do Planejamento **Programação em Par** **Integração Contínua** **Cliente local**	**Sim. Estes fatores dependem de uma aproximação entre o negócio, cliente e pessoal técnico.**
***Releases* Pequenos** **Metáforas** **Projeto de Software Simples** **Teste** **Refatoração** **Propriedade Coletiva** **40 horas semanais** **Padrão de codificação**	**Não. Independem de a equipe ser co-localizada ou não.**

O DXP assume a existência de algumas ferramentas e tecnologias condições para que seja eficaz, tais como:

- **Conectividade:** Alguma forma de conectividade precisa existir entre os membros da equipe. Para longas distâncias a *Internet* é utilizada como meio de comunicação.
- ***E-Mail*:** É um meio de troca de informação muito utilizado no DXP.

- **Gerenciamento de Comunicação:** Para uma gestão eficaz dos artefatos de programação é necessário que seja utilizada uma ferramenta de gerenciamento de configuração.
- **Compartilhamento de Aplicação:** Aplicações ou Softwares de compartilhamento de *desktop* devem estar disponíveis para as equipes distribuídas.
- **Uso de Vídeo Conferência:** O uso de áudio e vídeo entre equipes distribuídas é importante para uma comunicação eficaz. Além disso, há o envolvimento do cliente neste meio de comunicação, pois o mesmo não tem disponibilidade de estar no local da reunião.
- **Integração entre os membros de uma equipe móvel:** Caso necessitem se deslocar, podem utilizar equipamentos móveis para participar das atividades de desenvolvimento.

De acordo com KIRCHER [2001 & 2008], o DXP pode integrar membros de equipes remotas e móveis ao processo de desenvolvimento e, portanto, é uma extensão valiosa para o XP tradicional.

Adoção de *Scrum* em um ambiente DDS

A experiência que será descrita aqui foi parte da disciplina de Engenharia de Software da turma de Mestrado em Ciência da Computação de 2009 no Centro de Informática da UFPE. O objetivo da disciplina era o estudo em fábricas de software, fazendo uso de DDS e metodologias ágeis para realizar projetos reais. Os alunos tiveram o período da disciplina (primeiro semestre de 2009) para desenvolver de forma distribuída o produto conforme definido no início do curso. O projeto relatado, denominado *FireScrum* [CAVALCANTI, 2009 & FIRESCRUM, 2009], é uma ferramenta para facilitar o gerenciamento de projetos em ambientes distribuídos utilizando a metodologia *Scrum* (descrita no Capitulo 2).

O desenvolvimento foi dividido em seis módulos: *Core, Task Board, Planning Poker, Test Module, BugTracking* e *Desktop Agent.* A fábrica era composta por sessenta alunos distribuídos em seis equipes, onde cada equipe era responsável por um dos módulos citados. Todos os componentes de todas as equipes realizaram suas atividades de forma distribuída. O módulo que será relatado é o *Bugtracking*, implementado por nove estudantes divididos em três estados: seis no estado de Pernambuco (distribuídos em Recife e no interior), dois no estado da Paraíba e um na Bahia.

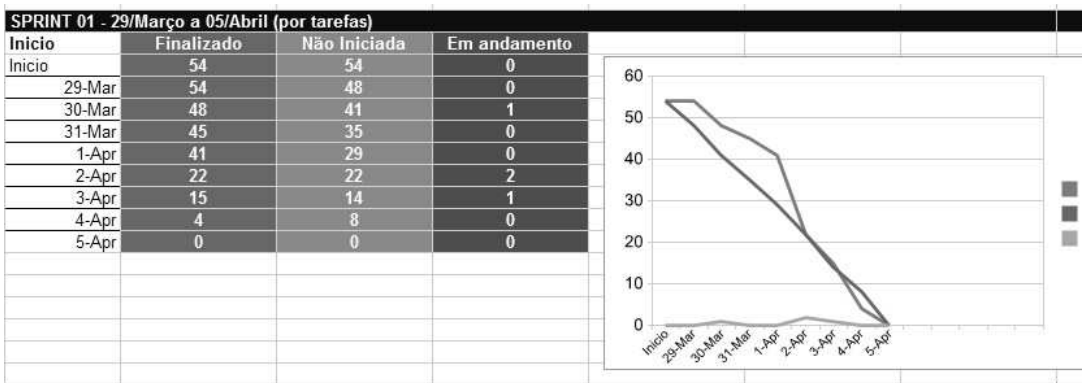
Para o desenvolvimento do módulo *Bugtracking*, a equipe realizou um estudo entre ferramentas *open source.* Assim, foi possível identificar as vantagens e desvantagens de cada uma para que o módulo fosse desenvolvido de forma diferenciada e inovadora, prezando pela simplicidade e usabilidade. O *Firescrum* foi desenvolvido utilizando a ferramenta *Adobe Flex,* o banco de dados utilizado foi o *Postgree SQL* e para o controle de versão foi utilizado o SVN.

O processo de desenvolvimento seguiu a metodologia *Scrum*. As *Sprints* tinham duração de 15 dias. A *Sprint Planning* 1, reunião para definir os itens que seriam atendidos na *sprint*, acontecia de forma presencial, quinzenalmente após a aula. A *Sprint Planning* 2, reunião na qual são definidas as tarefas necessárias à implementação das funcionalidades definidas na *Sprint Planning* 1, acontecia de forma remota

utilizando os seguintes recursos: *skype, MSN* e a planilha de gerenciamento criada no *Google Docs*. As reuniões diárias (*Daily Scrum Meeting*), com o objetivo de acompanhar a realização das tarefas, inicialmente aconteciam com o auxílio do *Skype e MSN*, posteriormente foi adotado um grupo de *e-mail*, pois os horários dos membros da equipe eram incompatíveis e nem sempre todos podiam participar das reuniões no horário marcado. Para os participantes que residiam na mesma cidade, aconteciam encontros em duplas (uso da prática de programação em pares) para discutir sobre o desenvolvimento, em seguida as dúvidas e conclusões eram postadas no grupo de *e-mail*.

Ao longo do desenvolvimento, a equipe manteve sempre evidente a filosofia de que cada membro era seu próprio gerente e responsável pelos resultados do projeto. Algumas práticas foram acordadas entre os membros para que os resultados necessários à conclusão da *sprint* fossem alcançados.

Os membros acompanhavam a evolução de três artefatos: a planilha de tarefas no *Google Docs*, o *burndown* e o grupo de *e-mail*. Isso possibilitou que a equipe mantivesse durante todo o processo de desenvolvimento um autogerenciamento satisfatório ao atendimento das *sprints*. A Figura 3.3 mostra o gráfico de *Burndown* das cinco *sprints* do projeto e a Figura 3.4 mostra a planilha de gerenciamento do *Google docs*.



SPRINT 01 - 29/Março a 05/Abril (por tarefas)			
Inicio	Finalizado	Não Iniciada	Em andamento
Inicio	54	54	0
29-Mar	54	48	0
30-Mar	48	41	1
31-Mar	45	35	0
1-Apr	41	29	0
2-Apr	22	22	2
3-Apr	15	14	1
4-Apr	4	8	0
5-Apr	0	0	0

Figura 3. 3 – Gráfico *Burndown*

BLI3 - Definir a severidade do bug						
	Tarefa	Dur. (h)	Owner	Status	Dia que iniciou	Dia que concluiu
P	Atualizar documento descrevendo os requisitos	2	Adelnei	CONCLUÍDO	31-Mar	31-Mar
R	Atualizar modelo de dados (entidades)	1	Adelnei	CONCLUÍDO	31-Mar	31-Mar
D	Atualizar implementação entidades de negócio	1	Emanoel	CONCLUÍDO	2-Apr	2-Apr
E	Atualizar implementação DAOs	1	Ana/Emanoel	CONCLUÍDO	2-Apr	2-Apr
S	Atualizar implementação Services	1	Ana	CONCLUÍDO	2-Apr	2-Apr
N	Atualizar implementação VOs	1	Yuri/Lubni	CONCLUÍDO	3-Apr	3-Apr
V	Atualizar implementação Business	1	Yuri	CONCLUÍDO	4-Apr	4-Apr
	Atualizar implementação Eventos	1	Yuri	CONCLUÍDO	3-Apr	3-Apr
	Atualizar implementação Comandos	1	Yuri	CONCLUÍDO	4-Apr	4-Apr
T	Atualizar Plano de Teste com a BLI 3	1		CONCLUÍDO*	3-Apr	3-Apr
E	Executar testes da BLI 3	1		CONCLUÍDO	5-Apr	5-Apr

Figura 3. 4 – Planilha de Gerenciamento

(A) Adaptação da metodologia para solução dos problemas encontrados no projeto

A grande dificuldade foi à realização da *Daily Scrum Meeting* (DSM), pois os integrantes da equipe possuíam atividades paralelas, tais como outras disciplinas da pós-graduação e alguns até trabalhavam, assim os horários disponíveis eram incompatíveis. De forma geral, estão listados abaixo problemas encontrados e adaptações na metodologia para obtenção de um melhor resultado.

- Foi determinado pela equipe que a DSM seria realizada a cada dois dias, evitando assim o risco de algum membro não ter nada a informar do que foi feito no dia, pois nem todos os membros da equipe estavam disponíveis todos os dias para executar tarefas relacionadas ao projeto.
- Para realizar a DSM, primeiramente foram utilizados o *Skype* e o *MSN* e foi acordado o horário das 20h30min para a reunião remota. Esta foi uma tentativa que não deu certo, pois a reunião tornava-se cansativa e eram frequentes as perdas na conexão.
- A demora na construção de frases claras era característica da perda de foco nas reuniões, assim as reuniões se prolongavam mais que o necessário e acabava-se entrando em peculiaridades dos problemas.
- Foi criado um grupo de e-mail, e foi acordado que a cada DSM seria postado até às 23h as respostas para as perguntas: O que foi feito até hoje?; O que será feito até a próxima DSM?; e Quais os impedimentos?.
- Foi criada uma planilha no *Google Docs*, na qual constava o *Product Backlog*, o objetivo de cada *sprint* e suas respectivas tarefas, o *burndown* e os impedimentos. Assim era possível acompanhar a dinâmica da equipe.

(B) Lições aprendidas com o uso do *Scrum* em um ambiente DDS

- **Integrantes da equipe autogerenciáveis:** A utilização do *Scrum* mostrou que para o sucesso do projeto é indispensável que os participantes sejam autogerenciáveis e, apesar da dispersão entre os pares, foi possível obter um bom resultado da equipe.
- **Implementação:** A experiência adquirida pela equipe com a utilização do ***Scrum*** em um ambiente DDS revela que a programação realizada à distância é possível de ser aplicada com o suporte de ferramentas disponíveis, tais como *e-mails* ou *MSN*.
- **Comunicação:** Várias ferramentas foram utilizadas e algumas não atingiram o objetivo da equipe por não suportarem vários usuários conectados ao mesmo tempo, por exemplo.

3.4 Considerações finais

O desenvolvimento de software sempre se apresentou de forma complexa, pois existe uma série de problemas e desafios inerentes ao processo. No contexto da globalização, a distribuição das equipes no tempo e no espaço tem tornado os projetos distribuídos cada vez mais comuns. O DDS, ao acrescentar fatores como dispersão geográfica, dispersão

temporal e diferenças culturais, acentuou alguns dos desafios existentes e acrescentou novos desafios ao processo de desenvolvimento. O trabalho em ambientes de DDS é mais complexo do que em ambientes centralizados e ainda não existem práticas maduras para o ambiente de desenvolvimento distribuído.

3.5 Tópicos de Pesquisa

Nos últimos anos muitos trabalhos apresentaram propostas de solução para as dificuldades e desafios existentes em projetos DDS. As pesquisas podem se concentrar em questões como o investimento em um modelo de negócio de DDS, a estrutura da operação, a relação com outras empresas ou unidades da própria empresa, projetos para equipes distribuídas e outros. A seguir apresentamos uma lista de tópicos de pesquisa no contexto de DDS.

- **Processo de desenvolvimento em um ambiente DDS aderente a modelos de qualidade de software:** Os modelos de qualidade de software reconhecidos internacionalmente e nacionalmente (ex: CMMI e MPS-BR) orientam as organizações para o desenvolvimento de processos, mas não propõem modelos para projetos distribuídos. A definição de um processo que considere o contexto de uma equipe distribuída e que seja aderente a modelos de qualidade de software é um tema de pesquisa relevante.
- **Ferramentas de colaboração:** Atualmente existem muitas ferramentas que oferecem suporte às atividades do ciclo de vida do desenvolvimento de um *software*. A maioria destas ferramentas não está adaptada para o cenário distribuído. Neste contexto, observa-se a necessidade de ferramentas que oferecem suporte ao *awareness* de atividade (quem está fazendo o quê), de processo (quem deve fazer o quê) e de disponibilidade (quem está disponível quando).

3.6 Sugestões de Leitura

A proposta deste capítulo foi apresentar uma visão de como é o processo de desenvolvimento de software em um ambiente distribuído. Neste contexto foram apresentadas adaptações de alguns processos de desenvolvimento plicados a este cenário, bem como foram discutidas as maiores dificuldades encontradas no âmbito de DDS.

De modo a aprofundar o conhecimento nos assuntos discutidos neste capítulo, algumas leituras adicionais são recomendadas.

- O livro **Desenvolvimento Distribuído de Software** de Jorge Audy e Rafael Prikladiniki, Elsevier, 2008, oferece uma visão geral sobre o tema.
- O livro **Global Software Teams**, de Erran Carmel, Editora Prentice Hall, 1999, aborda a formação de equipes globais de desenvolvimento de software e os fatores que devem ser levados em consideração ao montar uma equipe para um projeto DDS. Além disso, categoriza os fatores que levam a equipe ao sucesso e ao fracasso.
- A dissertação de mestrado **Munddos: Um Modelo de Referência para Desenvolvimento Distribuído de Software**, de Rafael Prikladiniki, Pontifícia

Universidade Católica do Rio Grande do Sul, propõe um modelo de referência para desenvolvimento distribuído de software. O modelo é baseado em um estudo empírico (estudo de caso) realizado em duas unidades de desenvolvimento de software de uma multinacional.

- O artigo **Distributed Scrum - Agile Project Management with Outsourced Development Teams**, de Jeff Sutherland et. al., Proceedings of the 40th Hawaii International Conference on System Sciences, 2007, disponível em http://www.computer.org/comp/proceedings/hicss/2007/2755/00/27550274a.pdf , Último acesso em 08/06/2010, analisa e recomenda boas práticas para o uso de *Scrum* em ambientes DDS.
- O artigo **Global Software Development – Managing Virtual Teams and Environments,** de Karolak, D. W., Los Alamitos, IEEE Computer Society, 1998, apresenta uma descrição de como funciona o gerenciamento de projeto em ambiente distribuído de desenvolvimento.

3.7 Exercícios

1. Defina o que é Desenvolvimento Distribuído de Software.
2. Quais as vantagens que uma organização tem ao utilizar um processo DDS?
3. Quais são os possíveis níveis de dispersão em um ambiente DDS? Exemplifique.
4. Quais os modelos de negócio em um ambiente DDS? Exemplifique.
5. Quais as principais dificuldades ao realizar um projeto DDS?
6. Como o processo tradicional RUP poderia ser adaptado para suportar um processo DDS?
7. Existem empresas/organizações que utilizam somente DDS em seus projetos? Cite exemplos.
8. Quais os desafios da utilização de práticas ágeis em ambientes DDS?
9. Como práticas ágeis podem ser aplicadas em ambientes DDS onde o encontro face a face não ocorre?
10. Em sua opinião, é possível ter um produto de software de qualidade com equipes trabalhando de forma dispersa?

3.8 Referências

Carmel, E. (1999) *Global Software Teams – Collaborating Across Borders and Time-Zones*, Prentice Hall, EUA, 269p.

Cavalcanti, E. (2009) FIRESCRUM: Ferramenta de Apoio à Gestão Ágil de Projetos Utilizando *Scrum.* Dissertação de Mestrado, CESAR, Recife – PE, Brasil.

Firescrum (2009). FireScrum – the Open Source Scrum Tool, 2007. Disponível em http://www.firescrum.com/. Último acesso em: 16/10/2009.

Herbsleb, J. D., Moitra, D. (2001) *Global Software Development*, IEEE Software, March/April, EUA, p. 16-20.

Herbsleb, J.D., Mockus, A., Finholt, T.A. e Grinter, R. E. (2001) *An empirical study of global software development: distance and speed*, ICSE, Toronto, Canada.

Karolak, D. W. (1998) *Global Software Development – Managing Virtual Teams and Environments*. Los Alamitos, IEEE Computer Society, EUA, 159p.

Kircher, M. (2001). *eXtreme Programming in Open-Source and Distributed Enviroments*, JAOO *(Java And Object-Orientation) conference*, Aarhus, Dinamarca.

Kircher, M., Jain, P., Levine, A. (2008) *Distributed Extreme Programming*, IEEE.

Navarro, Mirelli S., Alessi, Haroldo C. A Importância da Formação Universitária do Profissional de Sistemas de Informação. Disponível em www.mirelli.info/arquivos/artigo.pdf. Último acesso em 04/06/2010.

Prikladnicki, R. (2003) MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software. Dissertação de Mestrado, PPGCC – PUCRS, Brasil.

Prikladnicki, R., Audy, J. L. N. (2008) *Desenvolvimento Distribuído de Software*, Editora Elsevier.

Pressman, Roger S. (2005) *Software Engineering: a practitioner's approach.* EUA: McGraw Hill.

Sutherland, J. (2007) *Distributed Scrum: Agile Project Management with Outsourced Development Teams*, Proceedings of the 40th Hawaii International Conference on System Sciences, 2007. Disponível em http://www.computer.org/comp/proceedings/hicss/2007/2755/00/27550274a.pdf. Último acesso em 08/06/2010.