Capítulo

3

Processos para desenvolvimento distribuído de Software

Camila Cunha Borges[[1]](#footnote-2)

O objetivo do capítulo é apresentar 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 quatro décadas podemos observar que o software passou a ser o elemento-chave da evolução de sistemas e produtos baseados em computador [PRESSMAN 2007], 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, dificuldade de comunicação, etc.

Com a globalização dos negócios, surgem grandes desafios para o processo desenvolvimento de software, que está cada vez mais distribuído e global [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, assim são criadas novas formas de colaboração e competição na área de Engenharia de Software [DAMIAN 2006]. Neste ambiente, muitas organizações encontraram no Desenvolvimento Distribuído de Software (DDS) uma chance para obter sucesso nos negócios. Segundo AUDY e PRIKLADNICKI (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 refere-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. Estas mudanças impactam 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 (CPD) 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) pode-se ter uma economia entre 25% e 50% em termos de custo quando grandes projetos são transferidos para operações *offshore* (em outro país). Além da redução de custo, é observada a disponibilidade de profissionais habilitados para trabalhar em outro idioma. Incentivos de governos locais contribuem para essa nova forma de desenvolver software*.*

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*, onde há 24 horas contínuas, 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 de 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 em um processo 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 uma freqüência da comunicação diminui na mesma proporção que em uma distância de milhares de metros [HERBSLEB 2001]. É importante entender o nível de distâncias e suas implicações para as equipes. A seguir apresentamos tipos de distância física (Figura 3.1) e 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 presente 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, formando distribuição global.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.

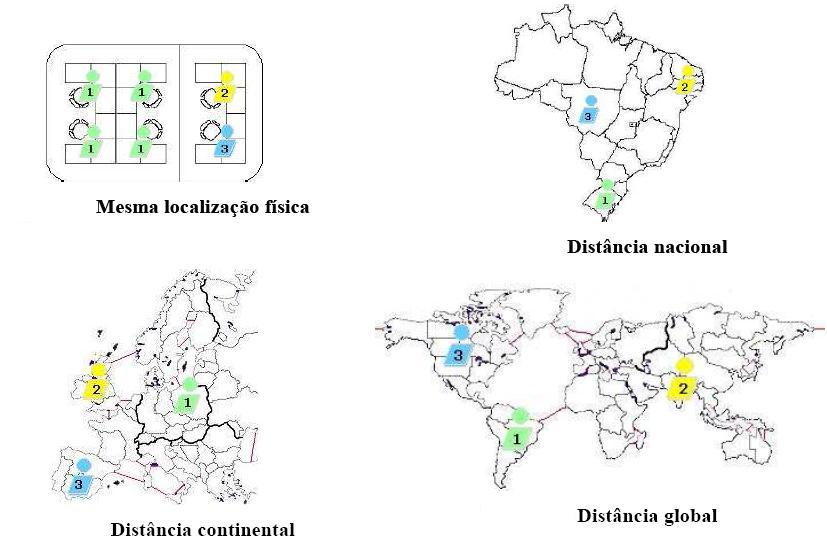
****

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

## 3.2.3 Modelos de Negócio

De acordo com PRIKLADINIKI (2007), entre as relações existentes entre clientes e provedores de serviço, podemos caracterizar *Outsourcing* (terceirização) e *Insourcing* (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 modelos 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 ela 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. Neste livro serão abordados apenas os modelos de negócio já citados nesta seção.

## 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. O ambiente global apresenta grande impacto na forma como os produtos são concebidos, desenvolvidos, testados e entregue aos clientes [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:** É um dos fatores mais utilizados para a diminuição do esforço entre as equipes. Conforme KAROLAK (1998), uma arquitetura apropriada para o DDS 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 (CM) é a chave para controlar as múltiplas peças em um projeto distribuído. Controlar modificações nos artefatos em cada uma das localidades com o processo de desenvolvimento de todo produto pode ser complexo. Apesar da utilização de práticas de CM auxiliar 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 CM 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 que auxilia a sincronização e a padronização das atividades é essencial. Com isso todos os membros utilizam uma nomenclatura comum em suas atividades.
* **Testes:** As práticas de teste e a verificação da qualidade 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

Em um ambiente DDS, um processo de desenvolvimento comum à equipe é fundamental, tendo em vista que uma metodologia auxilia diretamente na sincronização, fornecendo aos membros da equipe uma nomenclatura comum de tarefas e atividades, e um conjunto comum de expectativas aos elementos envolvidos no processo [PRIKLADNICKI 2008].

A Engenharia de software (ES) sempre está 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 (DDS), havendo uma necessidade do uso de práticas que dêem suporte às dificuldades encontradas nas definições de requisitos que mudam de forma dinâmica no decorrer do tempo. Estudos relacionados a processo para DDS ainda é escasso, sendo assim este capítulo relata o uso de práticas do desenvolvimento tradicional que podem ser implantadas em um ambiente distribuído e as possíveis adaptações.

# 3.4 Processos e adaptação das Práticas em projetos DDS

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 2001]. 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 torna-se cada vez mais importantes. A seguir é apresentada uma abordagem relacionada ao processo de desenvolvimento distribuído.

## 3.4.1 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:

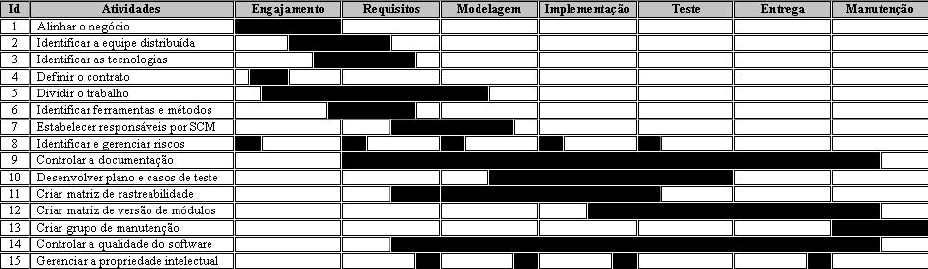


Figura 3. 2 – Modelo para projetos DDS [KAROLAK 1998]

A seguir serão descritas as atividades do modelo proposto:

* **Alinhar o negócio:** Primeira atividade necessária para desenvolver projetos DDS, pois será identificado o tipo de estrutura que será utilizada. 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 da equipe.
* **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 freqüência necessária de reuniões para discutir o andamento do trabalho.
* **Identificar e gerenciar riscos:** Esta atividade faz parte de qualquer projeto. De acordo com o autor, os riscos em projetos DDS tendem a ser mais centrados em aspectos não tão visíveis. Esta atividade deve acontecer em todas as fases do desenvolvimento. 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 documentados e o Caso de teste com 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, 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).

## 3.4.2 Uso de Práticas Ágeis

A partir do ano 2000, surgiu 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. Desde então, vários métodos ágeis foram surgindo, entre eles: **Adaptive Software Development, Crystal, Dynamic Systems Development, eXtreme Programming (XP), Feature Driven Development (FDD) e Scrum.** Alguns destes métodos foram explicados no capítulo 2.

Quando o ambiente 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 ao aumento na velocidade de desenvolvimento, alinhamento dos objetivos individuais com os organizacionais e melhoria no desenvolvimento [SUTHERLAND 2007].

De acordo com Kircher (2001), a proximidade entre Metodologias Ágeis e DDS ocorre devido a algumas características comuns, como *feedback* contínuo, *releases* freqüentes, valorização da comunicação e padrões de codificação. Além disso, Kircher apresenta um processo de desenvolvimento de software voltado para o ambiente distribuído, como é o caso do DXP (Distributed eXtreme Programming).

**DXP – *Distributed Extreme Programming***

Conforme abordado no Capítulo 2, a metodologia de desenvolvimento XP (*Extreme Programming*) requer uma comunicação forte e eficaz entre os membros de uma equipe de desenvolvimento de software. Para isso a metodologia enfatiza a necessidade de ter os membros da equipe fisicamente próximos uns dos outros. No entanto, nem sempre os membros da equipe de um projeto estão fisicamente próximos uns dos outros. Nesta seção será apresentada uma adaptação do uso do XP em 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.

Segundo KIRCHER (2001), o DXP aplica princípios XP em um ambiente distribuído, onde os membros das equipes também podem ser altamente móveis. A tabela 3.1 resume alguns dos aspectos que são relevantes para DXP e alguns que não são referentes ao fato da distribuição ou não das equipes.

Tabela 3. 1 – Adaptação do XP em DDS [KIRCHER 2000]

|  |  |
| --- | --- |
| **Práticas do XP** | **É necessário o time ser colocalizado?** |
| **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 se a equipe é Colocalizada ou não.** |

Conforme apresentado na tabela 3.1, podemos observar que para a utilização do DXP de forma eficaz é necessário que o Jogo de Planejamento,Programação Pareada*,* Integração Contínua *e* Cliente Local sejam abordadas em uma equipe distribuída. Na figura acima o autor considera que a prática de Refatoraçãonão exige um ambiente colocalizado apesar de esta prática exigir o uso da prática Programação Pareada*.* Kircher [2000] afirma que estas duas práticas podem iniciar separadamente.

O DXP assume a existência de algumas condições para que seja eficaz, tais como a disponibilidade de diversas ferramentas e tecnologias. Além das práticas do XP, o DXP assume:

* **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), o DXP pode integrar membros de equipes remotas e móveis ao processo de desenvolvimento e, portanto, uma extensão valiosa para o XP tradicional. Além disso, permite um envolvimento maior com o cliente quando comparado ao XP, principalmente em situações que é necessário ter o cliente no local*.* O autor enfatiza também a necessidade de atentar para os problemas já existentes em ambientes DDS, tais como comunicação, disponibilidade dos membros das equipes, coordenação, infraestrutura e gestão.

**Adoção de *Scrum* em um ambiente DDS**

Inserido neste contexto de desenvolvimento distribuído desoftware, esta seção apresenta a aplicação da metodologia *Scrum*, abordada no capítulo anterior, no processo de desenvolvimento de uma fábrica de software em um ambiente de desenvolvimento distribuído.

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

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 times, na qual cada time era responsável por um dos módulos citados. Todos os componentes de todos os times realizaram suas atividades de forma distribuída. O módulo que será relatado é o *Bugtracking*, composto 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 B*ugtracking,* o time 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* epara 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 *email*, pois os horários dos membros da equipe eram incompatíveis e nem sempre todos poderiam participar das reuniões no horário marcado. Para os participantes que residiam na mesma cidade acontecia encontros em duplas (uso da prática de programação em pares) para discutir a sobre o desenvolvimento, em seguida as dúvidas e conclusões eram postadas no grupo de *email*.

Ao longo do desenvolvimento, o time 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 *email.* Isso possibilitou que o time mantivesse durante todo o processo de desenvolvimento um autogerenciamento satisfatório ao atendimento das *sprints*. A Figura 3.3 mostra os gráficos das cinco *sprints* do projeto de desenvolvimento e a planilha de gerenciamento do *Google docs* (Figura 3.4).

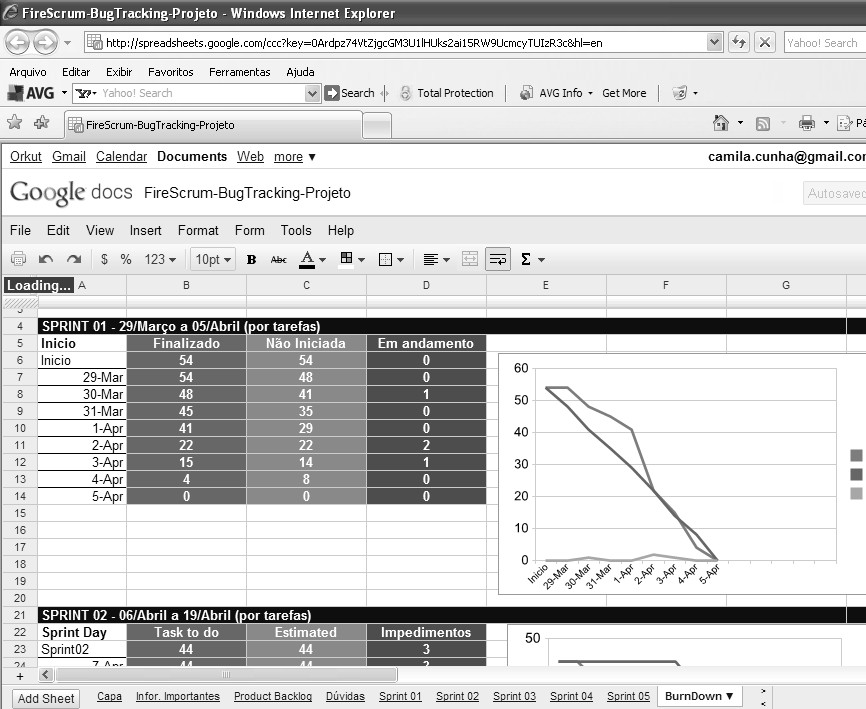


Figura 3. 3 – Gráfico *Burndown*

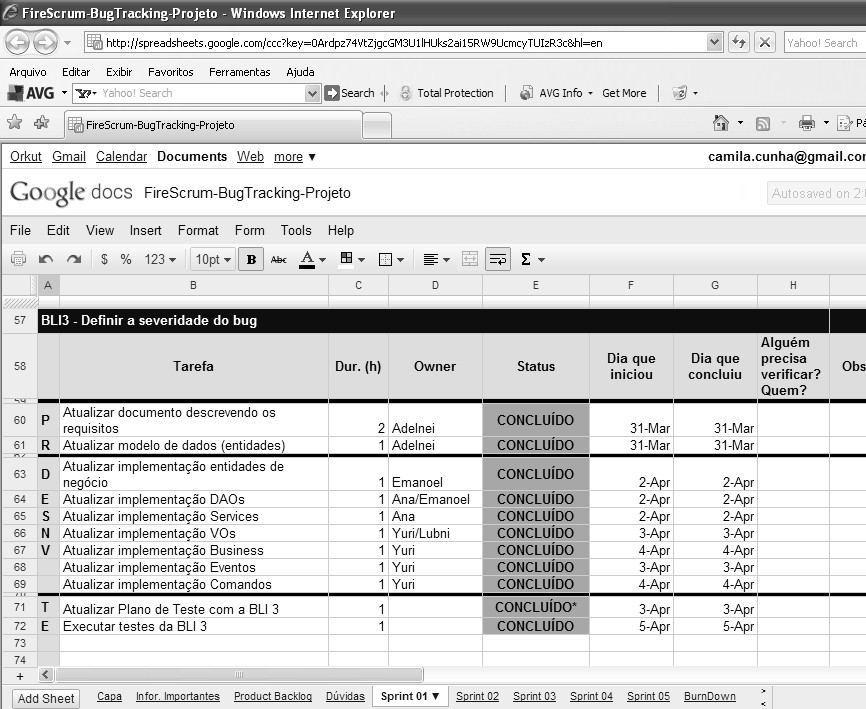


Figura 3. 4 – Planilha de Gerenciamento

1. **Adaptando a metodologia para solução dos problemas**

A grande dificuldade foi à realização da *Daily Scrum Meeting* (DSM), pois os integrantes do time 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 um melhor resultado.

* + Foi determinado pelo time 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 do time estavam disponíveis todos os dias para executar tarefas relacionadas ao projeto.
  + Para realizar a DSM, primeiramente utilizamos o *Skype e 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 frequentes eram as perdas na conexão.
  + A demora na construção de frases claras é característica de perda de foco nas reuniões, assim as reuniões se prolongavam mais que o necessário e acabássemos entrando em peculiaridades dos problemas.
  + Foi criado um grupo de email, e foi acordado que a cada DSM postaríamos até 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 do time.

1. **Lições aprendidas com o uso do *Scrum* em um ambiente DDS**

* Integrantes do time 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 pars, do time foi possível obter um bom resultado.
* Implementação: A experiência adquirida pela equipe com a utilização do *Scrum* em um ambiente DDS revela que a programação realizada a distância é possível de ser aplicada com o suporte de ferramentas disponíveis, tais como *emails* ou *MSN.*
* Comunicação: Várias ferramentas foram utilizadas e algumas não atingiam o objetivo do time por não suportar vários usuários conectados ao mesmo tempo, por exemplo.

# 

# 3.5 Considerações finais

O desenvolvimento de software sempre se apresentou de forma complexa. Existe uma série de problemas e desafios inerentes ao processo. Assim como o processo de desenvolvimento de software tem se tornado cada vez mais complexo, 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.

# 3.6 Tópicos de Pesquisa

Mecanismos de coordenação em ambientes DDS não existem ou são falhos [HERBSLEB 2001]. A distância afeta a colaboração entre as equipes, pois há uma menor freqüência de comunicação, comunicação ineficiente, falta de percepção, além da incompatibilidade de processos, ferramentas e práticas de trabalho.

Nos últimos anos muitos trabalhos apresentam 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:** A definição de um processo que considere o contexto de uma equipe distribuída. Os modelos de qualidade desoftwarereconhecidos internacionalmente (CMMI) e nacionalmente (MPS-BR) orientam as organizações no desenvolvimento de processos, mas não propõem modelos para distribuição e distância.
* **Uso de práticas em ambientes DDS:** O uso de uma prática pode ser aplicado em diferentes modelos de negócio de um ambiente DDS? Podemos comparar o modelo *Outsourcing* e o *Insourcing.*
* **Ferramentas de colaboração:**Atualmente existem muitas ferramentas que oferecem suporte as atividades do ciclo de vida do desenvolvimento de um *software.* Estas ferramentas são adaptadas 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).

# 3.7 Sugestões de Leitura

A proposta do capítulo foi dar uma visão de como é o processo de desenvolvimento de software em um ambiente distribuído. Foi apresentado o uso e adaptação de alguns métodos e as maiores dificuldades na prática. O tema DDS é complexo e estão disponíveis na literatura diversas abordagens sobre o tema.

* **Desenvolvimento Distribuído de Software [Prikladiniki]:** Este livro oferece uma visão geral sobre o tema.
* **Global Software Teams [Carmel]:** O autor 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, o autor categoriza os fatores que levam a equipe ao sucesso e ao fracasso.
* **MuNDDoS - Um Modelo de Referência para Desenvolvimento Distribuído de Software [Prikladiniki]:** Neste trabalho o autor 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.
* **Distributed Scrum - Agile Project Management with Outsourced Development Teams** **[Sutherland]:** Para uma leitura sobre o uso do *Scrum* em ambientes DDS.
* **Global Software Development – Managing Virtual Teams and Environments [Karolak]:** Leitura recomendada para o entendimento do gerenciamento do projeto distribuído e como funciona o ambiente de desenvolvimento.

# 3.8 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 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 um 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. Na sua opinião, é possível ter um produto de software de qualidade com equipes trabalhando de forma dispersa?

# 3.9 Referências

ADOBE. Flash Player Penetration, (2009). Disponível em: http://www.adobe.com/products/player\_census/flashplayer/. Acesso: 05/08/2009

BLAZEDS. Overview, (2008). Disponível em: http://opensource.adobe.com/wiki/display/blazeds/Overview. Acesso em: 31/08/2009.

FIRESCRUM. FireScrum ...the Open Source Scrum Tool, 2007. Disponível em: http://www.firescrum.com/. Acesso em: 16/10/2009.

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

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

Kiel, L. (2003) *Experiences in Distributed Development: A Case Study, In: Workshop on Global Software Development at ICSE*, Oregon, EUA, 2003, 4p.

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

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.

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.

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

Marquardt, M. J., Horvath, L. (2001) *Global Teams: how top multinationals span boundariesand cultures with high-speed teamwork*. Davies-Black. Palo Alto, EUA.

Prikladnicki, R., Audy, J. L. N., Evaristo, R. (2004) *Global Software Development in Practice: Lessons Learned, Journal of Software Process: Practice and Improvement – Special Issue on Global Software Development.*

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

J. L. N. Prikladinicki, R.; Audy (2007) Desenvolvimento Distribuído de Software.

Perrelli, Hermano. (2009) Visão Geral do RUP. Centro de Informática, Universidade Federal de Pernambuco. Disponível em: <http://www.cin.ufpe.br/~if717/slides/3-visao-geral-do-rup.pdf>. Acessado em 20 de Setembro 2009.

PostreSQL. Disponível em: http://pt.wikipedia.org/wiki/PostgreSQL. Acesso em: Abril de 2009.

Pressman, Roger S. (2001) *Software Engineering: a practitioner’s approach.* EUA: McGraw Hill, 860 p.

Subversion web site. (2009) Disponível em http://subversion.tigris.org/. Acesso em 20

de Agosto de 2009.

Sutherland, J. (2007) *Distributed Scrum: Agile Project Management with Outsourced Development Teams”*, HICSS.

Teles, Vinícius Manhães (2004) *Extreme Programming*: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. 1. ed. São paulo: Novatec. 320 p.

Travassos, G. H., Abrantes, J. F. (2005) Caracterização de Métodos Ágeis de Desenvolvimento de Software. UFRJ. Disponível em: <http://www.cos.ufrj.br/~jfa/agil_caracterizacao.pdf>. Acessado em 17 de Outubro de 2009.

1. Camila@cin.ufpe.br [↑](#footnote-ref-2)