UM MÉTODO DE ENSINO E PRÁTICAS DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE PARA CURSOS DE GRADUAÇÃO

Alexandre L'Erario [1]; Marcelo Schneck de Paula Pessôa [2]
[1] USP - Escola Politécnica da Universidade de São Paulo - Engenharia de Produção São Paulo, Estado de São Paulo, CEP 05508-900, Brasil
Universidade Tecnológica Federal do Paraná – Campus Cornélio Procópio
Cornélio Procópio, Paraná CEP 86300-000, Brasil
Fundação Educacional do Município de Assis
Assis, Estado de São Paulo, CEP: 19807-634, Brasil
alexandre.lerario@femanet.com.br

[2] USP - Escola Politécnica da Universidade de São Paulo - Engenharia de Produção
São Paulo, Estado de São Paulo, CEP 05508-900, Brasil
mpessoa@usp.br

Resumo: *Este artigo apresenta um estudo realizado colaborativamente entre universidades, cujo objetivo foi incorporar conceitos de desenvolvimento distribuído de software em seus cursos de graduação. Para tanto, alunos de instituições fisicamente distantes trabalharam em um ambiente de desenvolvimento colaborativo de software, similar ao ambiente que organizações utilizam. Este ambiente foi determinado considerando estudos de casos previamente realizados por L'Erario(2007). Para que houvesse sucesso nesta pesquisa foram envolvidas várias disciplinas de graduação.*

Palavras-chave: *Desenvolvimento distribuído de software, ensino de dds*

1 INTRODUÇÃO

Novas necessidades emergentes de mercado fazem com que as organizações que produzem software estabeleçam parcerias com o intuito de produzi-los. Uma maneira de estabelecer parceria entre as empresas que produzem software é construir uma rede de produção. Segundo Alstyne (1997), uma rede de organizações pode ser definida pelos elementos envolvidos na estrutura da rede, processo e propósito. No entanto, estas parcerias nem sempre são locais, ou seja, são compostas por várias organizações distantes fisicamente umas das outras. O resultado, segundo Herbsleb (1999) é a criação de organizações virtuais ou sistemas de times virtuais que podem operar com sucesso sobre as barreiras geográficas e culturais.

O desenvolvimento global de software tem tomado grandes proporções, pois, as organizações das mais diversas categorias descobrem constantemente que o desenvolvimento distribuído de software pode aumentar a produtividade e reduzir o custo.

A utilização de computadores autônimos interligados em ambientes organizacionais favoreceu o surgimento de sistemas de trabalho em grupo, denominado segundo Antunes (2005) como groupware. As mais diversas ferramentas de apoio a groupware, síncronas

(serviço de mensagem instantânea) ou assíncronas (e-mail, fórum) passaram a ser utilizadas amplamente nas organizações. Contudo Grudin (1994) dá provas de que o groupware pode limitar a cooperatividade dos participantes (fator social e motivacional; aspectos políticos do lugar de trabalho, citados por O'Day, 1996). Além disso, o desenvolvimento de atividades em grupo pode acarretar em conflitos que podem desencadear uma má execução das atividades e conseqüentemente comprometem o projeto e o produto final. As redes sociais, segundo Shami (2004) podem reduzir o impacto cultural sobre os sites da rede. Além disso, as redes sociais podem de certa forma induzir a uma interação mais atraente entre os elementos da rede.

As vantagens propiciadas pelo desenvolvimento distribuído favorecem o aumento de organizações que desenvolvem software de forma cooperativa/colaborativa. Este fato motivou este trabalho que busca integrar na graduação conceitos e práticas de desenvolvimento distribuído, em grades curriculares já existentes. Entretanto proporcionar um projeto de DDS em uma única instituição, na qual todos os alunos se conhecem ou tem a possibilidade do contato físico pode não refletir uma real situação que os conceitos de desenvolvimento distribuído de software apresentam. Este fato ocorre, pois, com uma distancia física mínima, os alunos podem ter contato presencial e distorcer os objetivos deste projeto.Por esta razão, neste projeto de pesquisa, foram envolvidas duas instituições de ensino com uma distancia física considerável entre elas. Além disso, foram envolvidos cursos com perfis diferentes. Em uma instituição foi envolvido um curso de Bacharelado enquanto que em outra foi envolvido o curso de tecnologia.

2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

O desenvolvimento distribuído de software (DDS) ocorre quando vários nós/*sites* (unidade independente de produção de software distante das demais) cooperam e/ou colaboram para desenvolver um mesmo produto ou parte dele. Neste cenário, a complexidade do processo de desenvolvimento se amplia. A redução do tempo é a principal razão da divisão de tarefas na produção distribuída de software, porém o tempo de comunicação e de resposta entre os nós pode ser incomensurável. Segundo Herbsleb (2003) e Suzuki (1999), há muitas variáveis neste cenário, tais como a cultura, a língua, a capacidade de cada nó entre outras. Além disso, Becker (2001) afirma que desenvolver software em uma rede de produção requer gerenciamento mais eficaz de tarefas. Martin (1996) afirma também que este gerenciamento deve estender-se a ferramentas, capacidades e informação.

Em sua pesquisa, Sa (2002), afirma que o crescimento da internet e de ferramentas de CSCW (descritas por Borghoff, 2000) influenciaram diretamente e, principalmente na difusão do desenvolvimento global de software, porém, embora a internet ofereça inúmeras vantagens em relação à comunicação, um projeto de software é muito complexo e grande. Por este motivo Suziki (1999) afirma a necessidade de um *framework* que deve esclarecer como a informação é gerenciada na rede e como as tarefas são delegadas, por exemplo.

Empresas que distribuem filiais ou efetuam *offshore* e/ou *nearshore* com o objetivo de aumentar a produtividade, reduzir o custo e tornar mais compatível com características locais são organizações que aderem ao GSD e segundo Carmel (2001), o número delas está crescendo. Todavia dividir a produção de um software entre diversos nós envolve uma série de problemas. Alguns deles que localmente são insignificantes ganham proporção quando uma tarefa é dividida entre organizações fisicamente distantes.

2.1 A dinâmica de um ambiente de dds

A dinâmica de um ambiente de DDS, segundo L'Erario (2007), revela o funcionamento da rede, da sua concepção inicial até seu encerramento. Abordar individualmente os *sites* na rede pode não revelar a influência que o projeto tem sobre sua existência na rede, por isso, a dinâmica da rede, representada na figura 1, aborda conjuntamente dois componentes fundamentais em um ambiente de DDS. O primeiro componente indica a dinâmica de uma unidade de produção de software. O segundo elemento é atrelado ao projeto que é desenvolvido em uma rede de produção de software. Sobre esta dinâmica, há uma intersecção que ocorre quando o *site* precisa desenvolver um subproduto de rede, e há uma instância do projeto para vários *sites*.

As setas de transição na figura 1 indicam a transição dos estados do projeto e do *site*. Na figura 1 o elemento indicado como *Novo Projeto* indica a concepção de um novo produto de software por alguma organização, seja ela em rede ou não. A seta 1 da figura 1 indica que o plano gerencial do projeto em termos de componentização e divisão de tarefas foi realizado. Após esta seta, todos, ou parte dos artefatos estarão prontos para serem disponibilizados e distribuídos para que a rede e os *sites* possam desenvolvê-los. O elemento *pronto*, após a seta 1, indica que um papel de distribuidor de tarefas entra em ação para gerenciar a distribuição das tarefas do projeto na rede.

Na figura 1, o elemento *Novo site* indica que um novo nó da rede está pronto para associar-se a uma rede de produção de software. A elipse *pronto*, após a seta *a*, indica que o *site* já está na rede, ou seja, de alguma forma um acordo foi feito entre o *site* e a rede, e que pode ser, por exemplo, o conceito de matriz e filial. O estado de *pronto* após a seta *a* indica que a organização submeteu-se a alguma análise, resultando essa em uma agregação na rede. Neste estado o *site* está aguardando tarefas.

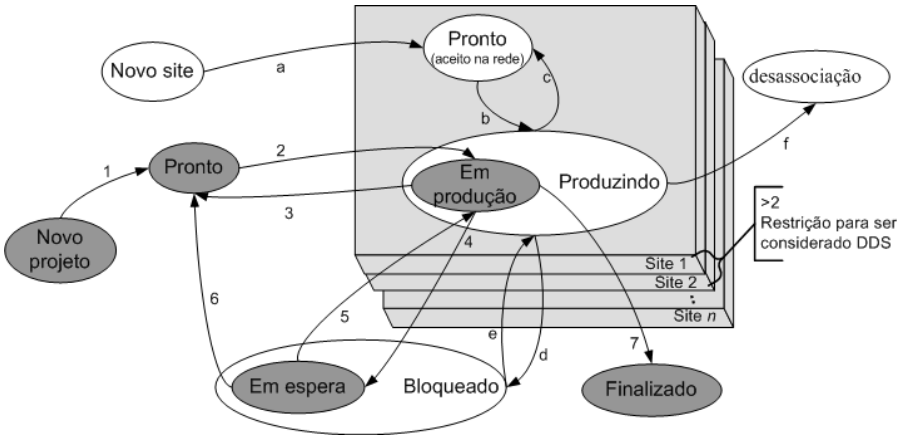

Figura 1 - A dinâmica de um ambiente de DDS

A seta 2 indica que o subproduto foi alocado em produção. Esta ação ocorre paralelamente à ação indicada pela seta *b*, que muda o estado do *site* de *pronto* para *produzindo*. O estado *Produzindo* efetua algumas tarefas de verificação, como por exemplo, aceitar ou não o trabalho. Se aceito, a tarefa entra em produção, caso contrário, a tarefa retorna e o papel do divisor de tarefas precisará realocar o trabalho (seta 3). O *site* eventualmente fica ocioso, aguardando nova tarefa (seta *c*).

As setas 4 e *d* podem ocorrer quando há uma dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subproduto fica bloqueado, pois, por exemplo, não pode ser

desenvolvido pelo fato de sofrer dependência funcional. O *site* fica ocioso, aguardando uma nova tomada de decisão. As seta 5 e *e* , representam respectivamente a resolução de dependência funcional e o retorno do nó na produção do artefato.A seta 7 indica quando o subproduto é finalizado e entregue à rede. Neste caso o estado *finalizado*, indica o fim da subtarefa para um determinado *site*. A ação indicada pela letra *f* indica a desassociação do *site* à rede que é removido da mesma e deixa de fazer parte do ambiente de DDS.

Na dinâmica, representada pela figura 1 há uma regra de restrição imposta sobre a quantidade de *sites* disponíveis. Para que seja considerado um ambiente de desenvolvimento distribuído de software, é necessário que a quantidade de *sites* seja maior ou igual a 2.

2.2 As propriedades dos ambientes de DDS

Há várias propriedades que diferem os ambientes de DDS. Cada propriedade determina o grau de complexidade de trabalho e revela consigo o quão difícil é de se produzir software em equipes fisicamente dispersas. Além disso, cada propriedade agrega consigo uma série de conceitos que são aplicados do nível estratégico ao operacional da rede.

Estas propriedades estão atreladas à dinâmica de funcionamento dos ambientes de DDS. Várias destas propriedades tornam-se eminentes, após determinados processos indicados na figura 1. A figura 2 ilustra as diversas as propriedades encontradas em um ambiente de DDS que são explanadas nas seções seguintes. Ela contém linhas escalares que compõem as diversas propriedades dos ambientes de DDS. A relação entre elas é expressa por uma condição mínima, definida na figura como: quantidade de *sites* maior que 1, distância, granularidade de repasse, difusão do processo e grau de interação maiores do que 0 . Esta condição mínima indica o quanto de cada propriedade deve ser configurada para que o ambiente de desenvolvimento seja considerado distribuído.

Além disso, cada plano, composto por estas propriedades é orientado por um mecanismo de coordenação, indicado por Mintzberg (2003). A abrangência dos mecanismos de coordenação, na qual os ambientes de DDS utilizam englobam ajuste mútuo, padronização e supervisão direta e exercem influência contínua sobre as propriedades dos ambientes de desenvolvimento distribuído de software que define resumidamente como o processo de desenvolvimento distribuído ocorre. Por exemplo, se o mecanismo for ajuste mutuo, indica que ambos os *sites* tem a mesmo grau hierárquico e precisam definir ajustar mutuamente os artefatos trocados entre eles.

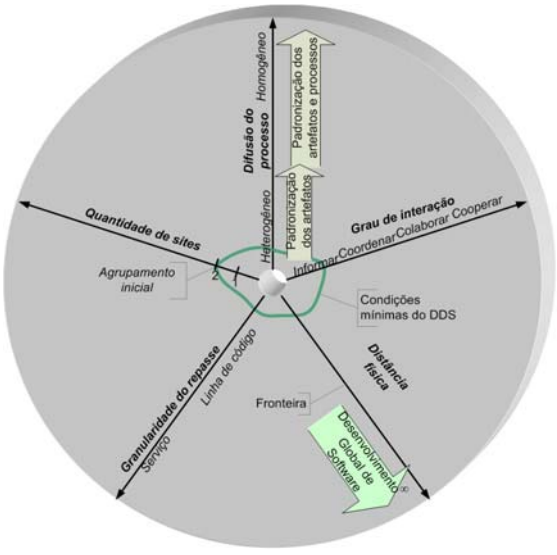

Figura 2 - Propriedades de um ambiente de DDS

O mecanismo de coordenação ajuste mútuo define que não há nenhuma padronização e os grupos precisam de ajustes contínuos entre artefatos trocados. A padronização como mecanismo de coordenação está subdividido em três categorias: padronização de resultados, processos e habilidades. A primeira subdivisão enquadra-se a padronização dos artefatos, a segunda do processo de software como um todo e a terceira, a padronização com relação a habilidades dos desenvolvedores existentes nos grupos. A supervisão direta indica total responsabilidade sobre um grupo, sendo que este deve definir como seus grupos subordinados trabalham.

As retas indicadas pela figura 2 são grandezas escalares. A reta definida como *difusão do processo* indica o quando o processo é homogêneo na rede. Nesta propriedade, a homogeneidade ou heterogeneidade do processo de todos os nodos do *site* são avaliadas. A reta definida como *grau de interação*, indica a intensidade de interação dos *sites*. A reta definida como granularidade do repasse indica a forma na qual o artefato é encaminhado de um *site* para outro. Para ser considerado distribuído os nodos devem ficar há uma distância considerável, um do outro (indicado pela reta definida como *distância física* da figura 2). A última reta, definida como a *quantidade de sites*, neste caso deve ser maior do que 1 (um).

As seções a seguir, explicam com mais detalhes o que cada propriedade aborda efetivamente dentro do ambiente de desenvolvimento distribuído de software.

3 INCORPORANDO TÉCNICAS E PRÁTICAS DE DDS NA GRADUAÇÃO

Em geral, não há uma disciplina específica de graduação que aborda desenvolvimento distribuído de software. O CR99/2003 da SBC (www.sbc.org.br), por exemplo, não contém nenhuma disciplina de desenvolvimento distribuído de software. Porém, este fato não impede que conceitos de DDS sejam contemplados nos cursos de graduação. Para tanto, neste projeto, os conceitos necessários foram incorporados em muitas outras disciplinas, tais como arquitetura de software, banco de dados, disciplinas de engenharia de software e programação. A Figura 3 indica o conjunto de conceitos que foram necessários para este projeto e as disciplinas que foram envolvidas.

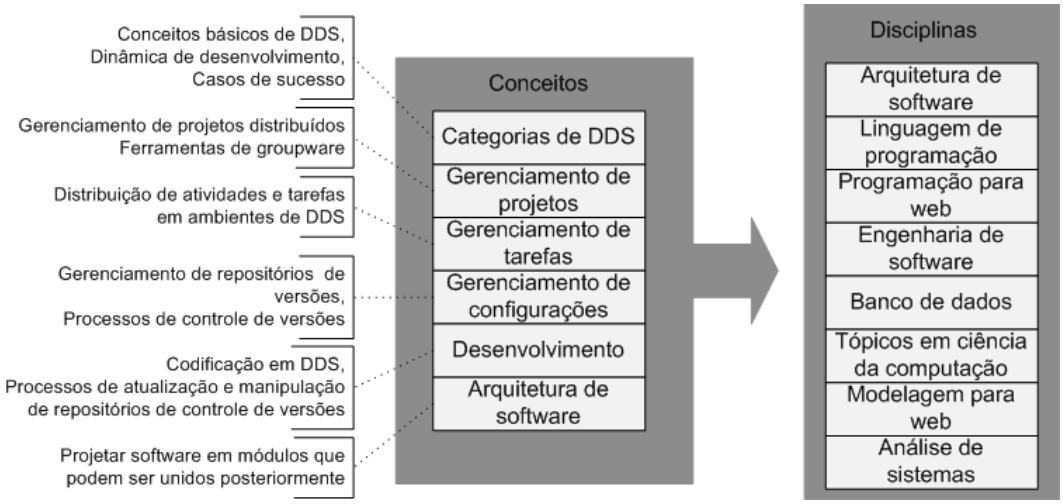

Figura 3 - Conceitos necessários para o projeto e disciplinas envolvidas

Antes do inicio do projeto, foi apresentado aos dissentes e docentes um posicionamento da organização de como seriam os times virtuais. Este posicionamento, baseado na seção 2 deste artigo, identifica a organização dos times, de forma que todos possam discernir as

diferenças entre os diversos tipos de organizações que desenvolvem software de maneira distribuída. Há várias diferenças entre um ambiente acadêmico e um ambiente corporativo real de DDS. A principal diferença, neste caso, é que em um ambiente corporativo, desde que haja um controle de qualidade, todos os funcionários seguem um mesmo conjunto de processos pré-estabelecidos controlados e validados. Neste projeto de pesquisa, os processos são desenvolvidos e validados ao mesmo tempo em que o projeto é desenvolvido. O processo foi homogeneizado conforme as interações e novas versões de documento surgiam. Os alunos, segundo a classificação de Mintzberg (2003), foram agrupados de acordo com o conhecimento e principalmente pelos produtos de trabalhos que geraram (*output*). Estes mesmos produtos de trabalhos foram utilizados como parâmetros nas avaliações dissidentes pelos docentes.

Todos os integrantes deste projeto (alunos e professores) seguiram a dinâmica apresentada na figura 1. Neste caso em particular, o modelo utilizado foi uma instancia mais especifica apresentado na seção 2.1. O perfil do projeto desenvolvido no âmbito acadêmico seguiu uma coordenação com ajuste mútuo, ou seja, houve muitas interações entre os participantes por meio de fórum, chats e e-mail.

Antes de iniciar o projeto com o corpo dissente, foi necessária uma interação entre docentes das instituições a fim de alinhar as disciplinas com o projeto de desenvolvimento distribuído de software. Nesta etapa foram analisadas as grades curriculares das instituições, visando selecionar as disciplinas que participariam do projeto de DDS. Também foi necessário fazer um alinhamento entre o conteúdo da disciplina e o projeto. Este alinhamento não envolveu modificações drásticas no conteúdo das disciplinas, mas ajustes temporais (em qual momento um conceito deveria ser ensinado) e adição de conceitos. Este processo está indicado pelas setas em cinza da figura 4.

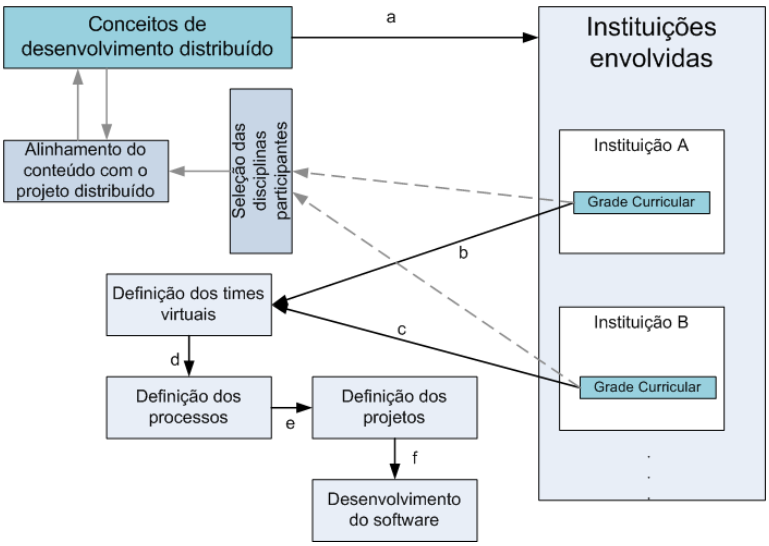

Figura 4 - O processo de incorporação de DDS na graduação

Na primeira etapa deste projeto, foram enumerados quais conceitos seriam ensinados nos cursos de graduação. Após esta etapa, os conceitos de DDS foram distribuídos pelas as instituições envolvidas (seta a da figura 4), de forma que cada disciplina pudesse cumprir um conjunto de metas. Estas metas incluem o ensino de ambientes de DDS, codificação distribuída, etc.

Os times virtuais eram definidos (seta b, c da figura 4) com alunos de várias disciplinas, cursos e instituições. Neste caso em particular, participaram duas instituições de ensino, uma

com o curso de Bacharelado em Ciência da Computação e outra com o curso de Tecnologia e Análise de Sistemas de Informação. Os alunos de instituições distintas não se conheciam pessoalmente e em alguns casos, alunos da mesma instituição, que cursavam disciplinas diferentes também não se conheciam. Nesta etapa, um fórum foi elaborado e mediado pelo professor de forma a apresentar os alunos e iniciar um projeto em conjunto.

A definição de um processo (Seta d da figura 4) foi a primeira atividade desenvolvida pelo time virtual. Nesta fase, arquitetos e engenheiros de software, considerando o projeto e time, elaboraram um processo a ser seguido. Todos os alunos do grupo participaram desta atividade, pois, precisaram estipular uma primeira medida inicial para que os arquitetos e engenheiros fossem capazes de montar um primeiro processo. A definição do projeto (seta e) descreve a fase na qual os alunos determinam qual o projeto que irão desenvolver. O desenvolvimento do software (seta f) é o desenvolvimento do projeto, gerando como produtos de trabalho documentação, codificação, etc.

3.1 Definição e organização dos times virtuais

Os times virtuais foram organizados de forma hierárquica. Neste trabalho, há uma diferença entre time virtual e grupo. A figura 5 ilustra como os times foram elaborados e qual a terminologia adotada neste projeto para definir o agrupamento dos alunos.

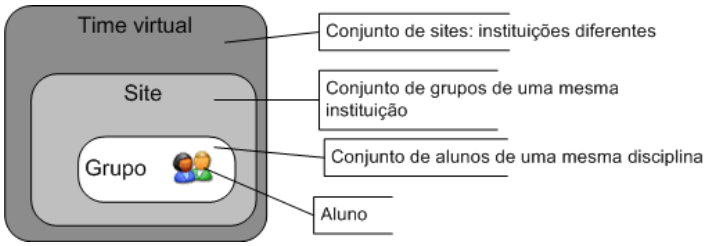

Figura 5 - Organização dos times

Para balancear a distribuição dos alunos nos times virtuais, foi elaborada uma tabela (Tabela 1) contendo um conjunto básico de relação entre papéis no projeto de DDS e as disciplinas na graduação. Esta tabela norteou a escolha do aluno na participação em papéis nos quais eram relevantes para a disciplina que cursava.

Tabela 1 - Relação entre os papéis do projeto e as disciplinas selecionadas

Papel / Disciplina	Programador	Arquiteto	Gerente de projetos	Analista	DBA	Testador
Arquitetura de software	2	1		2		
Banco de dados					1	
Programação Web	1					1
Projeto integrador		2	1			
Tópicos em ciência da computação 2	1	2	2			
Modelagem para web				1		

1. Para cada grupo na disciplina, ao menos um membro precisa atuar neste papel.
2. Papéis opcionais que podem ser desenvolvidos pelo time.

Desta forma foi possível avaliar cada aluno e grupo dentro da disciplina na qual estavam cursando. Por exemplo, se o aluno estivesse cursando modelagem para web, o único papel relevante para a disciplina seria o de analista. Os papéis eram definidos com maior exatidão na próxima atividade do time, representada pela *seta d* da figura 4.

3.2 Definição dos processos

O gerente de projetos neste caso também foi o coordenador do time virtual. Para possibilitar um gerenciamento mais eficaz, foi atribuído a ele um conjunto de tarefas iniciais. A primeira tarefa foi negociar com cada grupo de seu time um processo de software. O modelo em cascata foi utilizado por todos os grupos, pois o tempo que teriam entregar os produtos de trabalho foi restrito a 6 meses. A figura 6 é um exemplo de processo utilizado por um time.

O gerente entregou formalmente ao professou a definição do processo de software do seu time virtual. O documento de especificação do processo precisou definir ao menos os seguintes itens:

a. Fases: Quais fases teriam o projeto, a explicação detalhada do que é cada faze
b. Papéis: Quais papéis teriam seu projeto/processo. A explicação detalhada do que é cada papel. Qual aluno e/ou grupo assumiria cada papel
c. Iterações: Que entrada cada papel receberia e que saídas geraria em uma determinada fase/disciplina do projeto
d. Disciplina: Quais disciplinas (conjunto de atividades de uma área de interesse) e atividades seriam necessárias para concluir o desenvolvimento do projeto.

A figura 6 é um processo que um dos grupos entregou. Há nesta ilustração a separação por fases e papéis. As setas representam que determinado papel gera saída para outro. A tabela 2, é uma exemplo da configuração das setas apresentadas na figura 6.

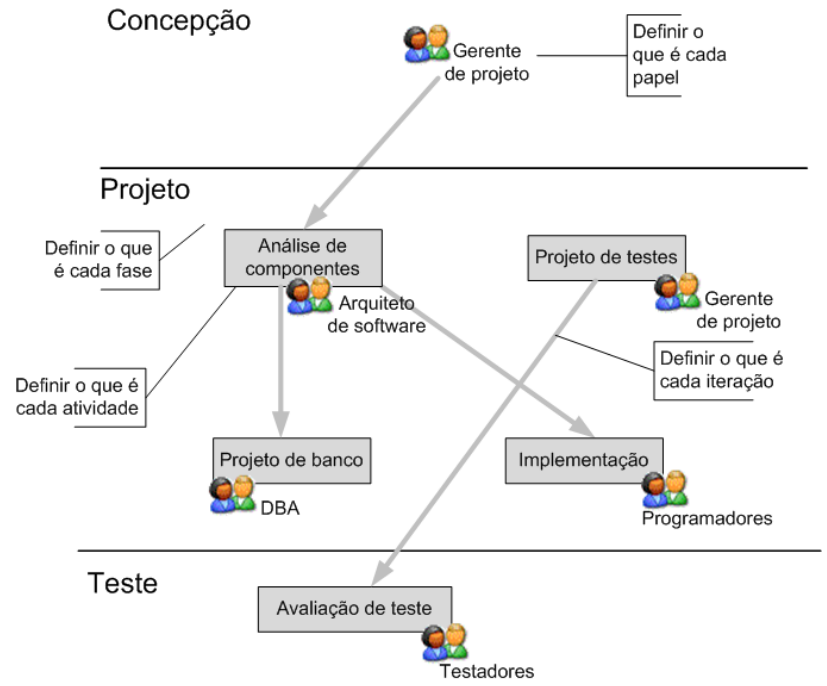

Figura 6 - Processo de software utilizado no projeto de DDS

O gerente também definiu uma tabela relacionando as interações do projeto. Desta forma, todos sabiam quais atividades e o que desenvolveriam no decorrer da disciplina. A tabela 2 é um exemplo de como um dos gerentes de projetos definiu os artefatos e tarefas. A tabela 2 representa uma personalização de um processo de software, criado por dissentes de acordo com seus times e capacidade de produção. A coluna disciplina nesta tabela, refere-se a disciplina no processo e não no curso de graduação.

Tabela 2 - Relação entre papéis, disciplinas e artefatos gerados

Fase	Disciplina	Artefatos de chegada	Artefatos de saída	Papéis
Concepção	Modelagem inicial	Entrevista com o cliente	Documento de visão (v1.0)	Gerente de Projeto
Projeto	Análise de componentes	Documento de visão (v1.0)	Modelo de arquitetura alto nível (v1.0) Modelo de componentes para implementação (v1.0)	Arquiteto de software
	Projeto de testes	Documento de visão (v1.0) Modelo de arquitetura de software alto nível (v1.0)	Documento de teste (v1.0)	Gerente de projetos
	Projeto de banco	Documento de visão (v1.0) Modelo de arquitetura de software alto nível (v1.0)	MER, Meta data, store procedures	DBA
	Implementação	Modelo de componentes (v1.0)	Rotinas, funções, etc.	Programador
Teste	Testes	Software / componente	Documento de revisão de teste	Testadores

3.3 Definição dos projetos

Neste projeto todos os desenvolvedores ainda estão em fase de aprendizagem e muito do que enfrentarão adiante pode representar um grande risco, pois, o que utilizam na implementação, documentação, no banco de dados, pode estar adiante na seqüência de conteúdo ensinado pela disciplina. Nesta etapa ainda há uma forte interação do professor com cada um dos integrantes do grupo, auxiliando e indicando qual literatura particular cada aluno deve procurar.

Cada professor de cada disciplina recebeu uma cópia do processo e projeto. Cada professor fez uma reunião com cada aluno separadamente a fim de averiguar as condições de cumprir o projeto/processo. Muitos processos foram modificados a fim de atender a disciplina em particular e respeitar as limitações dos alunos com relação aos ensinamentos que ainda não foram ministrados pelo professor.

O gerente de projeto foi o papel responsável em coletar requisitos e determinar uma visão compartilhada. Havia dois documentos em particular gerados nesta fase: o documento de visão e a especificação do processo (explicado no item 3.2).

Uma arquitetura comum também foi elaborada. Esta arquitetura inicial, elaborada pelo arquiteto de software serviu de *guideline* para todos os integrantes do grupo. Este *guideline* continha um exemplo de aplicação com operações em banco de dados, relatórios e outros e também um documento explicando como iniciar um novo componente. O arquiteto também ficou responsável em organizar o serviço de diretórios do sistema de controle de versões.

Todos os grupos foram obrigados a utilizar o sistema de controle de versões. A figura 7 é um exemplo da organização do sistema de diretórios de um dos grupos

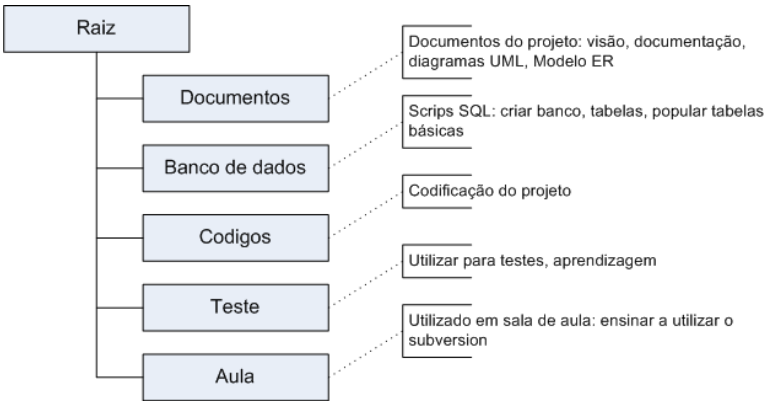

Figura 7 - Exemplo de estrutura de diretório no repositório de versões

3.4 Desenvolvimento dos projetos

Os projetos foram desenvolvidos cumprindo um calendário com data limite. Porém diferente do que acontece nas empresas, os alunos aqui envolvidos ainda estão em fase de aprendizagem. Desta forma, para reduzir o risco, foi necessário reduzir o tamanho do projeto, portanto cada grupo desenvolveu um pequeno projeto de software, porém completo, com documentações, versões e testes.

Havia reunião semanalmente do gerente com o professor para acompanhamento das atividades. Nesta reunião, o gerente informava o estado do projeto, os problemas encontrados e os problemas resolvidos. Todas as disciplinas da graduação tinham um ponto crítico no projeto. O ponto crítico era o momento na qual o projeto precisava do conhecimento de determinada disciplina. A tabela 3 exemplifica os pontos críticos em algumas disciplinas. A primeira linha desta tabela indica a porcentagem do projeto concluída. Nem sempre nos pontos críticos os alunos já tinham abordado na disciplina o conceito que iriam empregar. O resultado foi que os alunos buscaram soluções e em muitos casos, os professores relataram que os alunos aprenderam com muito mais facilidade quando o tópico era abordado em sala de aula.

Tabela 3 - Pontos críticos das disciplinas nos projetos

Projeto	25% (1 mês)	50% (2 meses)	75% (1 mês)	100% (1 mês)
Arquitetura de software	Elaborar uma arquitetura de acordo com o perfil dos integrantes e projeto	Elaborar uma arquitetura de acordo com o perfil dos integrantes e projeto	Reavaliar a arquitetura	Reavaliar a arquitetura
Projeto integrador		Especificar requisitos	Especificar/Codificar	Codificar
Banco de dados		Elaborar modelo ER	Elaborar store-procedure, scripts, etc.	
Engenharia de software 2	Elaborar um processo coerente com o time e projeto	Elaborar um processo coerente com o time e projeto		
Tópicos em ciência da computação		Especificar /Codificar	Especificar /Codificar	Especificar /Codificar
Programação para web		Codificar	Codificar	Codificar

Os primeiros meses, até o projeto alcançar 50% de implementação houve muitos impasses, principalmente nas disciplinas de arquitetura de software e engenharia de software 2. Estas contavam com uma maturidade para desenvolver um *guideline* de desenvolvimento que seria posteriormente utilizado por todos. Porém os alunos não tinham ainda uma maturidade para desenvolver tal atividade. Houve uma interação intensa entre o gerente, o arquiteto e os professores a fim de minimizar impasses futuros. Ao final do projeto, cada gerente/arquiteto apresentou quais impasses o projeto teve e quais soluções foram adotadas.

3.5 Critérios de avaliação

Foi elaborado um modelo para avaliação discente (figura 8). As avaliações constituíram basicamente da análise dos artefatos gerados em função do processo do time. Neste projeto foram considerados os seguintes escopos para avaliação:

- Avaliação do time: o time é avaliado como um todo. Seu produto final é avaliado, pois representa a unidade de produção do time virtual.
- Avaliação do site: Os grupos de uma instituição foram avaliados de acordo com suas produtividades
- Avaliação do grupo: grupo de estudantes que cursam a mesma disciplina foi avaliado
- Avaliação do indivíduo: avaliação do que o indivíduo produziu

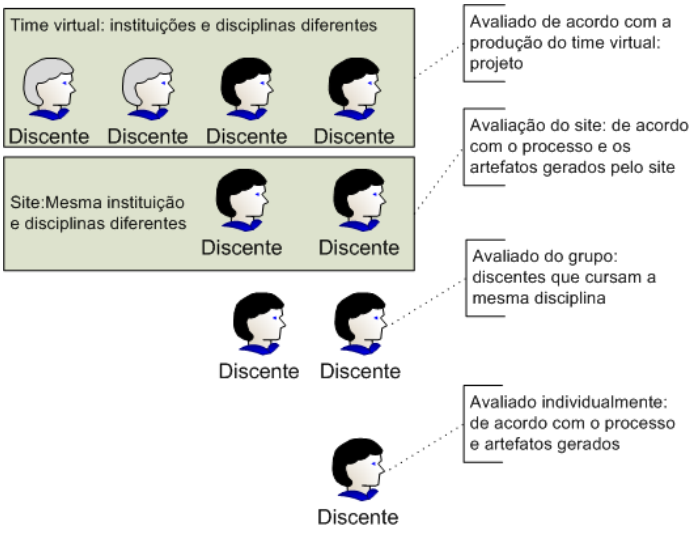

Figura 8 - Modelo de avaliação

Para avaliar o individuo e o site foi considerado as disciplinas nas quais estavam matriculados. Por exemplo, se o individuo estava matriculado na disciplina de programação web, seu papel no processo do time obrigatoriamente teria que ser de programador, o mesmo ocorre com que está matriculado na disciplina de arquitetura de software, tendo por obrigatoriedade que ser um arquiteto de software. Cada papel foi definido de acordo com a ementa da disciplina, desta maneira foi possível avaliar cada individuo e site na disciplina.

Todo gerente de projeto foi condicionado a entregar semanalmente um relatório de estado do projeto. Neste relatório o gerente descrevia: a fase do projeto, o estado do projeto (em

andamento ou estagnado), os principais problemas pendentes, os problemas resolvidos, como o grupo resolveu estes problemas e a perspectiva de realização das novas fases.

4 CONSIDERAÇÕES FINAIS E CONCLUSOES

Um dos principais pontos observados neste projeto foi a aprendizagem antecipada dos alunos. Uma conseqüência deste projeto foi expor os alunos a problemas que eles não haviam se deparado antes. Por este motivo, estes, precisaram aprender o conteúdo, antes de ser ministrado em sala de aula. Os professores das disciplinas na qual este fato ocorreu relataram uma aprendizagem e entrosamento dos alunos com uma eficácia muito maior, se comparado aos anos anteriores quando os trabalhos eram desenvolvidos de acordo com a disciplina e não havia um projeto central.

A troca de experiência entre alunos que estavam concluindo o curso, por exemplo, alunos da disciplina de arquitetura de software, com alunos que estavam iniciando o curso, (disciplina de linguagens de programação), juntamente unidos em um projeto, cursos de diferentes perfis (bacharelado e tecnologia), revelaram uma aprendizagem mais interativa à medida que os problemas eram revelados no decorrer do projeto. Além dos problemas técnicos, os alunos precisaram aprender a lidar com o gerenciamento de times. Neste projeto, os principais problemas encontrados foram a diferença de conhecimento entre alunos e a diferença de experiência (alunos que já haviam ingressado no mercado de trabalho e alunos que ainda não atuavam no mercado).

O gerenciamento de times heterogêneos e de formações distintas trouxe para o aluno uma convivência real e prática do que acontece em organizações que trabalham com desenvolvimento distribuído de software. A distância física fez surgir impasses de ambigüidades que não haviam sido discutidas na primeira versão do processo. O gerente coordenou a produção e precisou realocar atividades algumas vezes.

A interação entre os membros do time virtual nem sempre foi coerente. Em muitos casos, problemas precisaram ser discutidos de forma a amenizar ambigüidade e impasses. Este fato aconteceu principalmente pela diferença entre o conhecimento técnico de cada time/aluno. Um exemplo foi o uso de frameworks para desenvolvimento web. Embora adotando *struts* em seu projeto, uma das equipes, achou desnecessário a documentação de como iniciar a implementação de novos requisitos, pois a maioria de seus membros tinha uma experiência neste framework. Este time em particular foi composto em sua maioria por discentes que já atuavam no mercado e já tinham uma experiência com banco de dados e *struts*. Mas, experiência não foi suficiente para minimizar os impasses gerados em função da distância. A falta de detalhes na definição de processos de como os produtos de trabalho deveriam ser construídos, fez com que este mesmo grupo recodificasse muito com o objetivo de compatibilizar artefatos de trabalhos não coerentes.

Um ambiente de controle de versões foi disponibilizado para todos os grupos. Por meio deste ambiente foi possível extrair dados, como por exemplo, o volume de código gerado pelo grupo, a evolução do trabalho, etc. Este ambiente foi crucial e além dele, um ambiente de desenvolvimento homogêneo foi disseminado pelos grupos a partir do arquiteto de software. Esta atividade fez com que todos os integrantes do mesmo grupo utilizassem o mesmo conjunto de ferramentas e tecnologias de desenvolvimento. Para os grupos que despenderam um trabalho mais refinado nesta tarefa, não houve muitos impasses posteriormente no desenvolvimento.

Além de um ambiente homogêneo, ou seja, todos os integrantes do time utilizaram as mesmas ferramentas IDE, mesmo banco de dados, mesma base de controle de versões e mesmo processo, os alunos aprenderam que é necessário também um contexto onde os

componentes são gerados. Este contexto é composto por bibliotecas utilizadas no projeto, bibliotecas utilizadas durante os testes, esquemas de acesso a banco de dados e outras configurações particulares. Foi necessário documentar como utilizar o contexto para que os artefatos de trabalho fluíssem com mais agilidade entre os grupos. Para este projeto o contexto não foi estático, ou seja, cada time definiu seu próprio contexto. O contexto evoluiu gradativamente a medida que foram adicionadas ou atualizadas bibliotecas que os grupos utilizaram no desenvolvimento do software.

Um dos grandes desafios deste projeto foi apresentar a dissentes como as organizações estão desenvolvendo software de maneira distribuída. O grau de realismo perante a ambientes reais foi próximo. Todos os alunos enfrentaram as dificuldades oriundas da distancia física, diferença de conhecimento e cultura.

REFERÊNCIAS BIBLIOGRÁFICAS

ALSTYNE, Marshall Van. The state of network organization: a survey in three frameworks. in Journal of Organizational Computing & Electronic Commerce, 7(3); pp.83-151, 1997.

ANTUNES, P. **Groupware: Conceitos Fundamentais e Caracterização dos Principais Blocos Construtivos.** Relatório Técnico. Disponível em: <http://www.di.fc.ul.pt/tech-reports>. Acesso em: julho 2005.

BECKER, Simon et al. A Delegation Based Model for Distributed Software Process Management. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, 8., 2001, Tókio. **Proceedings...** Tókio: ACM, 2001. p. 130 - 144.

BORGHOFF, U., SCHLICHTER, J., **Computer-Supported Cooperative Work – Introduction to Distributed Applications.** 1. ed. Berlin: Springer, 2000. 529 p.

CARMEL, Erran; AGARWAL, Ritu. Tactical Approaches for Alleviating Distance in Global Software Development. **IEEE Software**, v. 18, n. 2. março/abril 2001. p. 22-29.

GRUDIN, Jonathan. CSCW: History and Focus. **IEEE Computer**, New York, v. 27, n. 5, p.19-26, Maio 1994.

HERBSLEB, James D.; GRINTER, Rebecca E.. Architectures, Coordination, and Distance: Conway. **IEEE Software**, New York, v. 16, n. 1, p.63-70, 1999.

HERBSLEB, James D.; MOCKUS, Audris.. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. **IEEE Transactions on Software Engineering**, v. 29, n. 3, p.1-14, 2003.

L'ERARIO, Alexandre; PESSÔA, Marcelo Schneck de Paula. An Analysis of the Dynamics and Properties of the Distributed Development of Software Environments: A Case Study. In: SOFTWARE ENGINEERING RESEARCH AND PRACTICE, 2007, Los Angeles. p. 471 - 477.

MARTIN, Patrick. A Management Information Repository for Distributed Applications Management. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS, 2., 1996, Tókio. **Conferência.** Tókio: SBC, 1996.

MINTZBERG, Henry. **Criando organizações eficazes:** Estruturas em cinco configurações. 2. ed. São Paulo: Atlas, 2003. 334 p.

O'Day, V. L.; Bobrow, D. G.; Shirley, M. The social-technical design circle. Ackerman, M. S. ed. CSCW 96; Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work; 1996 November 16-20; Boston, MA. New York: ACM; 1996; 160-169.

SA, Jin; MASLOVA, Elena. A unified Process Support Framework for Global Software Development. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 26., 2002, Oxford. **Proceedings...** Oxford: IEEE, 2002. p. 1 - 2.

SHAMI, N. Sadat et al. An experimental simulation of multi-site software development. In: CONFERENCE OF THE CENTRE FOR ADVANCED STUDIES ON COLLABORATIVE RESEARCH, 10., 2004. **CASCON 2004.** 2004: ACM, 2004. p. 255 - 266.

SUZUKI, Junichi; YAMAMOTO, Yoshikazu. Leveraging Distributed Software Development. **IEEE Computer**, New York, v. 32, n. 9, p.59-65, setembro 1999.

A METHOD FOR TEACHING AND PRACTICIES OF DISTRIBUTED DEVELOPMENT OF SOFTWARE FOR GRADUATION COURSES

Abstract: *This article presents a study accomplished collaboratively among universities, whose objective was to incorporate concepts of distributed development of software in their graduation courses. The students developed a software project in a distributed way, like as the organizations develop. This environment was created considering cases studies accomplished previously by L'Erario (2007). Was necessary involves several graduation disciplines for success of this project.*

Key-words: *Distributed development of software, DDS teaching*