Desenvolvimento de Ambientes de Aprendizagem Cooperativa na Web

Cristiane Guimarães, Mônica D. Fernandes, Neide Santos

Depto de Informática e Ciência da Computação

Universidade do Estado do Rio de Janeiro

Rio de Janeiro – RJ – Brasil – CEP 20550-013

neide@ime.uerj.br

Resumo: Um problema a ser enfrentado pelo projetista de *software* refere-se às etapas a serem seguidas para o desenvolvimento de aplicações para a Web, em especial as aplicações para aprendizagem cooperativa. Nosso objetivo é apresentar diretrizes de apoio ao processo de desenvolvimento, a partir das fases de um ciclo de vida clássico de desenvolvimento de software: Análise, Projeto, Implementação, Avaliação e Testes, e Manutenção. Verificamos que as fases de Análise e de Implementação parecem ser as que sofrem maior impacto com a consolidação da Web como plataforma de desenvolvimento.

Palavras Chave: Ambientes e Metodologias de Autoria de Atividades de Aprendizagem. Aprendizagem Colaborativa Apoiada por Computadores

1. Introdução

O crescimento da Web criou novas oportunidades para os projetistas de software, tanto dentro quanto fora de corporações. Uma nova categoria de aplicações está surgindo e elas são centradas em rede, ricas em mídia, direcionadas para o usuário final e possibilitam significativa cooperação entre pessoas. Desenvolver ambientes cooperativos é cada vez mais encontrar soluções baseadas na Web. Em paralelo, surgem novos enfoques de desenvolvimento de *software*, como tecnologia de componentes, objetos distribuídos, *design patterns* e *frameworks*. A especificidade da Internet com sua arquitetura cliente-servidor, seus protocolos, padrões e suas novas aplicações, requer um novo processo de desenvolvimento? Pode-se utilizar um dos processos existentes?

No desenvolvimento de aplicações para CSCL (*Computer-Supported Cooperative Learning*), um problema a ser enfrentado pelo projetista de *software* refere-se a que diretrizes de desenvolvimento seguir. A partir da especificidade destas aplicações, o objetivo deste trabalho é definir diretrizes para o desenvolvimento deste tipo de aplicação. Com este propósito, identificamos posições teóricas e exemplos práticos sobre desenvolvimento de ambientes CSCL e utilizamos como guia as fases de um ciclo de desenvolvimento de software. Elegemos, por sua simplicidade, o ciclo de vida clássico, com suas cinco fases: Análise, Projeto, Implementação, Avaliação e Testes, e Manutenção.

2. Diretrizes de Apoio ao Processo de Desenvolvimento de Aplicações CSCL

A fase de **análise** requer a definição das tarefas a serem desenvolvidas no *software*, subdividindo-as em pequenas partes ou descrevendo o sistema de maneira a facilitar seu entendimento. Em ambientes de aprendizagem cooperativa, o projetista pode adotar as seguintes diretrizes:

a) Descrever o domínio da aprendizagem cooperativa

O desenvolver deve descrever o domínio de aprendizagem cooperativa e o tipo de tarefa (por exemplo: aprendizagem de conceitos, solução de problemas, desenvolvimento de projetos, construção de conhecimento, fórum de discussões) que o software vai atender. O projetista precisa analisar qual é o seu propósito com a construção de um ambiente de cooperação, pois cada tipo de tarefa requer um suporte de software diferente.

b) Verificar se há soluções a serem reutilizadas

A reutilização de soluções conceituais adotadas em desenvolvimentos anteriores e de componentes de software é cada vez mais uma necessidade para otimizar a construção de software. É fundamental que o processo de desenvolvimento seja documentado e posto disponível para a comunidade de projetistas. A documentação pode ser realizada se adotando os padrões de processo, de projeto e de documentação. Santoro, Borges e Santos (2001) construíram um sistema de padrões de processo para ambientes CSCL, onde os elementos envolvidos na cooperação estão descritos a partir de problemas e soluções identificados. O modelo é baseado no Objetivo que se deseja atingir com a proposta cooperativa, e no Processo que representa as atividades cooperativas a serem desenvolvidas e em seus relacionamentos. Dentre os componentes do sistema de padrões, o projetista deve eleger aqueles que irão compor os requisitos do ambiente a ser desenvolvido.

c) Definir os requisitos do sistema

- Genericamente, os ambientes CSCL devem atender aos seguintes requisitos:
- Dar suporte à edição cooperativa, provendo whiteboard para a edição de documentos;
- Dar suporte à comunicação síncrona e assíncrona entre os membros do grupo de trabalho;
- prover interface multi-usuário;

Dar suporte a diferentes modos de cooperação: modo fortemente acoplado (as interfaces são obrigadas a observar a ação de um usuário individual), e modo fracamente acoplado (cada interface pode agir independentemente);

Prover suporte para awareness, ou seja, para a percepção do que os outros estão fazendo agora ou fizeram anteriormente;

- Ter instrumentos que possibilitem ações conjuntas sobre o objeto de estudo;
- Prover protocolos de acesso para lidar com prioridades e restrições de uso das informações compartilhadas;
- Fornecer formas de representar opiniões conflituosas e o processo de discussão:
- Fornecer meios que permitam o trabalho individual e garantam o espaço de cada participante;
- Dar suporte à decisão em grupo através de brainstorm, votação, negociação e resolução de conflito;
- Prover controle de versões;
- Dar suporte à reutilização de textos inteiros ou partes de textos desenvolvidos pelo grupo;
- Permitir a construção ou atualização de versões dos trabalhos; e
- Oferecer possibilidade de recuperação direta, permitindo o downloading de versões do último trabalho desenvolvido.

d) Avaliar e documentar a etapa

A fase de análise deve gerar a especificação informal do sistema. Esta fase pode ser avaliada através de lista de verificação, comparando-se os objetivos do ambiente com os componentes selecionados do sistema de padrões de processo e os requisitos definidos para o sistema. Para futura reutilização, a especificação informal, na forma de uma lista de requisitos, deve constituir parte da documentação do processo de desenvolvimento.

A fase de **projeto** visa definir técnicas que descrevam o sistema a ser desenvolvido. Os requisitos definidos na fase de análise devem ser traduzidos em uma representação do software, sendo avaliada sua qualidade antes do início da codificação. Desta forma, o projetista deve:

a) Escolher um modelo para a especificação do sistema que represente o conceito de orientação a objetos

Os sistemas distribuídos, como os ambientes CSCL, podem ser modelados a partir de objetos, representados como classes do sistema. Pode ser útil expressar formalmente os requisitos do sistema em uma notação que represente o conceito de orientação a objetos, como UML.

b) Sempre, que possível, adotar componentes de software

Objetos distribuídos e os padrões CORBA e COM não são componentes físicos de software, estando entre partes especificadas do sistema e sua codificação. Mas, uma vez que eles se tornem disponíveis, sua utilização pode orientar a modelagem e a especificação do software. Se disponível, o projetista pode utilizar também framework de componentes. Um dos produtos da documentação é uma biblioteca de classes modeladas para posterior reuso.

Na fase de **implementação**, deve-se considerar que a arquitetura mais simples das aplicações Web é uma extensão do servidor Web, executando um programa ou uma extensão do servidor baseada em API ou em um script do lado do servidor. As aplicações Web são sistemas cliente/servidor. Para o servidor Web, muitas aplicações também usam servidores de aplicação, de tal forma que os usuários utilizam o browser para acessar o servidor, e o servidor Web chama o servidor de aplicação para atender o processamento do pedido e envia de volta a resposta para o usuário (browser). Isto envolve certo grau de complexidade, pois é preciso um protocolo a mais entre o servidor Web e o servidor da aplicação. Outra arquitetura é aquela em que o software roda como um servidor standalone, que tem um componente HTTP que fornece apenas um conjunto de URLs ad-hoc.os objetos distribuídos, padrões (como CORBA e COM) e frameworks de componentes podem otimizar o processo de desenvolvimento, desde que se tornem disponíveis para os projetistas. Os frameworks de classes e os toolkits e groupkits, podem diminuir a complexidade da codificação do sistema. Neste sentido, o projetista deve:

a) Analisar os frameworks disponíveis à luz dos requisitos do ambiente CSCL

Os frameworks dão ao projetista um conjunto de classes codificadas, o diminui o tempo de desenvolvimento do software e garante melhores níveis de qualidade no produto final. Habanero, desenvolvido pelo National Center for Supercomputing Applications (NCSA), é um framework colaborativo e um ambiente contendo um conjunto de aplicativos. Ele é uma API (Application Programming Interfaces) cooperativa e um conjunto de aplicações que permitem aos usuários compartilhar objetos Java remotamente pela Internet. O framework também possibilita a construção de groupware com boa redução de tempo, disponibilizando bibliotecas necessárias para criação de novas aplicações, ou conversão de aplicações convencionais já existentes em aplicações colaborativas. Em educação, está sendo investigado o uso do Habanero para facilitar a implementação do conceito de classes virtuais. No entanto, no Habanero, há somente algumas ferramentas implementadas e não é simples reutilizar a biblioteca de classes em Java. Para maiores detalhes, ver Guimarães & Dutra (2001).

b) Analisar os toolkits e groupkits disponíveis à luz dos requisitos de seu ambiente CSCL

Há alguns toolkits e groupkits disponíveis. No entanto, pode ocorrer que nenhum deles atenda aos requisitos do ambiente CSCL a ser desenvolvido, pois eles não foram projetados para ambientes CSCL.

TeamWave é um toolkit que dá suporte a comunicações virtuais síncronas e assíncronas, apresentando rica variedade de ferramentas virtuais interativas, incluindo um escritório virtual no qual podem ser armazenados dados, que são utilizadas em conferências e colaborações Web. As aplicações incluem apresentações, serviços ao cliente, conferências, oferecendo ao usuário suporte à comunicação textual através de recursos de áudio e vídeoconferência. Outros toolkits disponíveis são Groupkit, Egret e Suite (Guimarães e Dutra, 2001).

c) Analisar os ambientes de programação disponíveis

Caso os frameworks e os groupkits disponíveis não dêem suporte à implementação do ambiente CSCL, o projetista deve analisar os ambientes de programação e verificar qual deles a equipe de desenvolvimento domina ou tem os pré-requisitos necessários para dominar a curto prazo. Grande parte das aplicações Web funciona como extensões do servidor Web. Entre as primeiras decisões que o projetista deve tomar está a escolha do método de extensão a ser usado. Os mais comuns são: CGI, Script do lado do servidor, API nativa do servidor Web e API Servlet Java. Isto implica em um bom manuseio de linguagens de programação. Uma das vantagens do uso de uma linguagem para implementar um ambiente CSCL é a flexibilidade oferecida, o que permite incorporar características importantes no ambiente. Mas o domínio destas linguagens pode não ser tarefa trivial. A linguagem Java está se firmando como um padrão de linguagem para desenvolvimento de aplicações para a Web. O fato de ser orientada a objetos ajusta-se à plataforma Web. No entanto, no Brasil, a comunidade de usuários de Java é pequena e se pode considerar outras opções de implementação, como ASP e JavaScript.

d) Avaliar e documentar a fase

Deve-se construir critérios para avaliar a qualidade da especificação e da modelagem usando técnicas de avaliação. Para esta fase, Schneiderman (1998) assinala que os testes de usabilidade em laboratório podem ser proveitosos. Seria interessante gerar padrões de programação para posterior reuso.

e) Implementar um protótipo da interface e do software

Cada vez mais é recomendado que o desenvolvimento de software adote um enfoque incremental, e que a interface com o usuário seja prototipada usando um ambiente de software disponível para este fim. Alguns critérios a serem seguidos no projeto da construção da interface são consistência, integridade estética, estilo de input, see-and-point, o uso de metáforas do mundo real, sempre que possível, e uso de feedback e diálogos.

Na fase de **testes e avaliação**, deve-se proceder aos testes. Schneiderman (1998) recomenda que sejam utilizados os testes de aceitação, avaliação durante o uso com usuários reais e as listas e os fóruns de discussão.

Na fase de **manutenção**, deve ser definida a política de manutenção do ambiente CSCL. Esta etapa costuma ser negligenciada em alguns processos de desenvolvimento. Devem ser previstos melhoramentos no ambiente, para futura inclusão de novas funcionalidades e ferramentas. O modelo da Internet - desenvolvimento distribuído em diferentes servidores de aplicações - pode tornar a manutenção mais ágil e fácil.

3. Conclusões

Definimos algumas diretrizes para o desenvolvimento de aplicações para CSCL. Verificamos que as fases de Análise e de Implementação parecem ser aquelas que sofrem maior impacto com a consolidação da Web como plataforma de desenvolvimento. A fase de Análise, no desenvolvimento de uma aplicação cooperativa, mostra que determinados requisitos devem ser contemplados, e que tais requisitos somente serão implementados em uma plataforma distribuída como a Web. A fase de Implementação pode beneficiar-se dos *frameworks* de classes, construídos para aplicações na Web, e dos pacotes de *software*, como os *groupkits*, tornando o processo mais ágil, mas eventualmente diminuindo o nível de personalização do ambiente cooperativo.

As conclusões do trabalho sugerem que os conceitos básicos da Engenharia de Software, norteadores do processo de desenvolvimento, atendem à situação particular do desenvolvimento para a Web. Talvez a popularização, com a conseqüente disponibilização, de padrões como CORBA e COM, venha diminuir as dificuldades da passagem entre a fase de Projeto e a de Implementação.

4. Bibliografia

Applications and Standards. In http://biz.inter.nl.net/abraham/INTRANET/what-are/FUNCTION/

Distributed Groupware. In http://www.laas.research.ec. org/ cabernet/sota/report/ node76.html

Guimarães, C. & Fernandes, M. (2001). *Diretrizes para o Desenvolvimento de Ambientes de Aprendizagem Cooperativa na Web*. Projeto de Final de Curso. Depto de Informática e Ciência da Computação/UERJ, Rio de Janeiro, Maio.

Santoro, F.M., Borges, M.R.S., Santos, N. (2001). Modelo de Cooperação para Aprendizagem Baseada em Projetos: Uma Linguagem de Padrões. In: *1ª Conferência Latino Americana em Linguagens de Padrão para Programação – SugarLoaf PloP*. Outubro. Rio de Janeiro, Brasil.

Schneiderman, B. (1998). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley - 3rd edition.