PDS-E: Em direção a um processo para desenvolvimento de Software Educacional

Lucia Giraffa, Sabrina Marczak, Rafael Prikladnicki

Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Faculdade de Informática, Prédio 30 – Bloco IV
CEP 90.619-900 – Porto Alegre – RS – Brasil

{giraffa, smarczak, rafael}@inf.pucrs.br

***Abstract**: Educational software can be explored in different ways according teachers goals and methodologies. The design of such systems needs to support a great variety of requirements related to educational functionalities. This paper presents PDS-E, a set of Software Engineering practices applied to design educational applications. PDS-E was written to help teachers and others educators to organize and to formalize their ideas concerning the system design, including pedagogical features. It was tested in real situation with teachers from High Schools with different backgrounds.*

***Resumo:** Os programas educacionais, conforme a metodologia utilizada pelo professor, podem ser explorados de diferentes formas. Neste sentido, estes ambientes requerem um conjunto de funcionalidades para atender os aspectos pedagógicos, identificados pelos professores especialistas. Como esta tarefa não é de fácil execução dado o perfil diferenciado da equipe interdisciplinar, faz-se necessário que se utilize uma metodologia para guiar e organizar todo o projeto de desenvolvimento do software. Neste artigo apresenta-se um relato de experiência em uma disciplina da graduação da PUCRS no intuito de formalizar as práticas de engenharia de software adotadas para o projeto e desenvolvimento de software educacional, através do projeto PDS-E.*

1 Introdução

Os atuais programas educacionais utilizam recursos hipermídia, permitem o trabalho individual ou coletivo, utilizam técnicas de Inteligência Artificial em diferentes escalas e, conforme a metodologia utilizada pelo professor, podem ser explorados de diferentes formas. Estas possibilidades fazem com que o projeto destes sistemas seja bastante complexo. Hoje não é mais possível desenvolver um bom software educacional sem a ajuda de uma equipe interdisciplinar composta por especialistas da área de domínio da aplicação, especialistas em Informática na Educação (principalmente na área de projeto e desenvolvimento desta modalidade de programas) e em Engenharia de Software. Os novos ambientes requerem um grande conjunto de funcionalidades para atender os aspectos pedagógicos, com metodologias cada vez mais apoiadas em processos interativos, monitoramento e auxílio do trabalho do aluno, *feedback* para o professor e muitas outras. A fim de obter-se de forma clara o conjunto de funcionalidades necessárias para o *design* do sistema, deve-se fazer um levantamento criterioso dos requisitos identificados pelos professores especialistas. No entanto, esta não é uma tarefa trivial ou de fácil execução dado o perfil e conhecimento tão diferenciado da equipe interdisciplinar. Logo, faz-se necessário que se utilize uma metodologia para guiar e organizar todo o projeto de desenvolvimento do software educacional.

Existe uma série de metodologias para desenvolvimento de software, como, por exemplo, MSF (*Microsoft Solutions Framework*) e RUP (*Rational Unified Process*). Todas podem, em tese, atender as necessidades de um projeto desta natureza. No entanto, elas falham no que concerne em muitos aspectos e acabam não sendo utilizadas. Pode-se citar como exemplo o conjunto de documentos, diagramas até produtos de software para auxiliar a modelagem dos sistemas educacionais. Dentro deste contexto surge o projeto PDS-E - Processo para Desenvolvimento de Software Educacional.

O artigo está organizado em 5 seções. Na seção 2 apresentam-se conceitos relacionados à engenharia de software, as principais abordagens e ferramentas, bem como os principais problemas e desafios encontrados na área. Na seção 3 apresenta-se o relato de experiência da aplicação de alguns destes conceitos em uma disciplina de graduação. Na seção 4 apresentam-se as considerações finais. Na seção 5 têm-se as referências bibliográficas.

2 Referencial Teórico

2.1 Engenharia de Software

De um modo geral, pode-se entender a Engenharia de Software como sendo um conjunto de disciplinas. A literatura apresenta diversas definições, e algumas delas são apresentadas a seguir:

> "Engenharia de Software pode ser definida pelo estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais [PRE 01]".

> "Engenharia de Software é uma disciplina que reúne metodologias, métodos e ferramentas a serem utilizados, desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolver problemas inerentes ao processo de desenvolvimento e ao produto de software [CAR 01]".

Ainda que muitas definições abrangentes tenham sido propostas, todas elas convergem no sentido de apontar para necessidades de maior rigor no desenvolvimento de software. Sendo assim, a partir da revisão bibliográfica e baseando-se nas definições anteriormente citadas, encontrou-se na definição de [IEE 93] uma forma mais abrangente para definir a Engenharia de Software. Esta definição diz que:

> "Engenharia de Software é a aplicação de um ambiente sistemático, disciplinado e quantificável para o desenvolvimento, operacionalização e manutenção do software; ou seja, a aplicação da engenharia ao software [IEE 93]".

Segundo [PRE 01], a Engenharia de Software pode ser entendida através de camadas. Estas camadas abrangem três elementos fundamentais: ferramentas, métodos e processo. De acordo com a Figura 1, cada um destes elementos corresponde a uma camada, sendo que a camada base representa o foco na qualidade. Isto significa que os elementos representados nas três primeiras camadas devem ser capazes de possibilitar ao gerente de software o controle do processo de desenvolvimento de software e oferecer ao desenvolvedor uma base para a construção de software de alta qualidade.

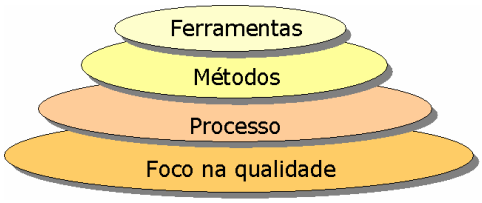

Figura 1 - As camadas da Engenharia de Software

Fonte: [PRE 01]

Os métodos de engenharia de software proporcionam os detalhes de "como fazer" para construir o software. Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste, manutenção, etc. As ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos. Atualmente, existem ferramentas para sustentar cada um dos métodos citados anteriormente. Quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser usada em outra, é estabelecido um sistema de suporte ao desenvolvimento de software chamado Engenharia de Software Auxiliada por Computador (CASE - *Computer-Aided Software Engineering*).

O processo é a camada mais importante da engenharia de software [PRE 01]. Esta camada se constitui no elo que mantém juntos as ferramentas e os métodos, além de possibilitar um desenvolvimento racional do software. Um processo define a seqüência em que os métodos serão aplicados, como os produtos serão entregues, os controles que ajudam a assegurar a qualidade e a coordenar as mudanças, e os marcos de referência que possibilitam aos gerentes de software avaliar o progresso do desenvolvimento. Um processo de desenvolvimento de software é representado por um modelo, enquanto que o modelo é operacionalizado por meio de uma metodologia. Existem diversos modelos de processo de desenvolvimento de software, e cada modelo pode ter mais de uma metodologia que o operacionaliza. A metodologia estabelece basicamente a seqüência das atividades e como elas se relacionam entre si, identificando o momento em que os métodos e as ferramentas serão utilizados [PRE 01].

Sendo assim, um processo de desenvolvimento de software deve ser implementado de acordo com um modelo previamente definido, seguindo uma metodologia que se satisfaça às necessidades e objetivos existentes, e tudo isto deve servir de guia para a correta utilização dos métodos e das ferramentas, tendo sempre em mente que a camada básica é o foco na qualidade.

2.2 Problemas e Desafios do Desenvolvimento de Software

Os problemas que afetam o desenvolvimento de software podem ser caracterizados a partir de uma série de perspectivas diferentes. Entre os principais problemas, pode-se citar a especificação de requisitos, capacitação de pessoal, qualidade e teste de software, planejamento e gerência de projeto, cumprimento dos prazos, trabalho em equipe, e muitos outros. Buscando sintetizar todos os problemas existentes, pode-se agrupá-los em algumas categorias mais amplas (Figura 2).

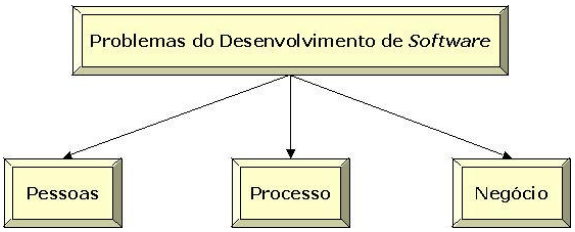

Figura 2 – Categorias de Problemas do desenvolvimento de software

Desta forma, pode-se dizer que os problemas identificados abrangem as dimensões das pessoas, dos processos e dos negócios. Sendo assim, os problemas citados na literatura são classificados da seguinte forma (vide Tabela 1):

Tabela 1 – Os problemas e autores relacionados

Categoria	Problemas	Autores relacionados
Pessoas	Capacitação de Pessoal	[McC 96], [PRE 01], [SOM 03]
	Motivação	[McC96], [SCH 00]
	Produtividade	[McC96]
	Trabalho em Equipe	[SCH 00], [SOM 03]
Processo	Especificação de Requisitos	[PET 01], [PRE 01]
	Qualidade e teste de software	[COR 01], [PRE 01], [SCH 99]
	Manutenibilidade	[PRE 01], [SOM 03]
Negócio	Custo	[PET 01], [PMI 00]
	Gerência de Projeto	[PMI 00], [ROY 98]
	Planejamento	[AUD 01], [PRE 01], [REP 98]
	Prazo	[McC 96], [PRE 01]

Além dos problemas, existem inúmeros desafios do desenvolvimento de software. Os principais desafios de hoje estão fortemente ligados com a maioria dos problemas relatados na literatura. Entre os principais, podem-se citar os novos ambientes de desenvolvimento (desenvolvimento de software com equipes geograficamente distantes), gerência de riscos, a existência de padrões no desenvolvimento de software, melhoria de processos de software, reutilização e gestão do conhecimento.

Em resumo, o processo de desenvolvimento de software tem avançado muito nos últimos anos, mas muitos desafios devem ser vencidos e muitas barreiras precisam ser quebradas. A evolução dos ambientes geograficamente distribuídos mostra uma

nova tendência em desenvolvimento de software no âmbito mundial. Além disso, um dos fatores determinantes para o sucesso do desenvolvimento de software em uma empresa é a capacidade de implantar um processo e possibilitar formas de criar, armazenar e disseminar o conhecimento gerado neste contexto. Ao longo das últimas décadas a engenharia de software passou por uma grande evolução, uma mudança radical em termos de metodologias/abordagens/processos de desenvolvimento de software. Diversos autores deram contribuições importantes para a engenharia de software chegar aos dias de hoje de forma organizada e definida. A ordem cronológica de como toda esta evolução ocorreu ao longo do tempo pode ser vista na Figura 3.

Figura 3 - Evolução das abordagens de desenvolvimento de software

2.3 Engenharia de Software e Software Educacional

Com o aumento da complexidade dos softwares educacionais em relação aos sistemas produzidos ano atrás, da diversidade de tecnologias adotadas e do número de pessoas envolvidas, tornou-se inadequado projetar um programa educacional sem utilizar-se um processo bem definido para orientar o seu desenvolvimento. As equipes interdisciplinares integrantes dos projetos de software educacional agora necessitam de especialistas de Engenharia de Software (ES). Estes contribuem para organização e definição de todos os aspectos relacionados à produção do software. Existem três aspectos fundamentais envolvidos na ES: *métodos*, que proporcionam os detalhes de "como fazer" para construir o software através da definição de um conjunto de tarefas; *ferramentas*, que proporcionam apoio automatizado ou semi-automatizado aos métodos; e *processos*, que constituem o elo que mantém juntos as ferramentas e os métodos [PRE 01]. O último aspecto é considerado o mais importante da ES.

Em relação às ferramentas, existe no mercado um conjunto de ferramentas que dão suporte à modelagem de software usando a UML (*Unified Modeling Language*), o que facilita a adoção desta técnica e auxilia na manutenção da modelagem e na troca de informação entre os envolvidos no projeto. Uma das ferramentas mais conhecidas no mercado de Informática é a *Rational Rose*, da IBM. Esta ferramenta é bastante completa, oferecendo suporte não só a fase de modelagem do software, mas também de todo o seu ciclo de desenvolvimento (ex. é possível modelar, gerar o "esqueleto" do código-fonte, testar o sistema, entre outras atividades, de forma integrada). *Visio* da Microsoft é outra opção que oferece um conjunto de diagramas para modelagem. A grande desvantagem destas duas ferramentas é o seu custo de aquisição, pois se faz necessário adquirir licenças para usufruir os seus recursos. Em geral, profissionais envolvidos com o desenvolvimento de software educacional, sejam eles da área da Ciência da Computação, Educação, Pedagogia ou afins, não possuem verbas disponíveis

para investir neste tipo de recurso. Para minimizar esta dificuldade, tem-se disponível algumas ferramentas para modelagem de software gratuitamente.

Segundo [FER 01], estas ferramentas possuem funcionalidades bastante reduzidas se comparadas às ferramentas comerciais, mas que podem suprir boa parte das suas necessidades. Dentre as de distribuição gratuita, a que merece destaque é a *ArgoUML*, totalmente escrita em Java e de código-fonte aberto. Sua versão mais recente já dá suporte a UML 1.3, mas permanece restrita a um subconjunto de diagramas: de casos de uso, classe, estado, atividade, colaboração e seqüência, este último com algumas restrições [GOM 04]. Alternativas de distribuição gratuita são as ferramentas *TCM*, *Dia-UML*, *Umbrello*, *DOME*, *Proxy Designer*, entre outras. Destas, nem todas foram utilizadas, apenas estudadas através de seus materiais de referência com a intenção de elencar as alternativas disponíveis para uma comunidade com restrição de verba para este tipo de recurso.

3 A experiência de aplicação da metodologia nas disciplinas de graduação em Pedagogia Multimeios

O Projeto PDS-E foi criado para suprir as lacunas enfrentadas pelos pesquisadores e professores que atuam em disciplinas de projeto de software educacional no curso de Pedagogia Multimeios da PUCRS. E, não somente no contexto destas disciplinas, bem como, para organizar, definir e padronizar a forma como são elaborados e projetados os softwares educacionais dentro do grupo de pesquisa de Informática na Educação da Faculdade de Informática da PUCRS.

Os resultados do projeto foram organizados e geraram uma metodologia, testada em dois semestres consecutivos com uma ótima avaliação por parte dos alunos. O conteúdo programático da disciplina engloba os seguintes tópicos:

- Conceitos Básicos de Engenharia de Software: Ciclos de Vida no Desenvolvimento de Software, Modelagem Conceitual de Software, Especificação de Requisitos, Modelo de Casos de Uso, Diagrama de Atividades e Interface com o Usuário;
- Projeto de Software Educacional: Enfoques pedagógico, computacional e ergonômico, Estudos de Caso com Diversas Modalidades de Software Educacionais e Validação de Software Educacional;
- Desenvolvimento de um Projeto de Software Educacional: Descrição Pedagógica, Descrição Software/Hardware, Especificação de Requisitos, Elaboração do Modelo Conceitual, Elaboração da interface gráfica com o usuário e requisitos ergonômicos.

Para um melhor entendimento das fases e necessidades envolvidas num projeto de software educacional, os alunos recebem no início do semestre um tema (ou escopo, que define a área de abrangência do projeto) que deverá ser observado quando da escolha do tipo de aplicação que os alunos irão trabalhar. Dado o porte do trabalho, grupos são criados. O grupo deve ter no mínimo três e no máximo seis membros. Um exemplo de tema/escopo utilizado é que o sistema deve ser na Web. Partindo-se desta especificação os alunos devem chegar a um consenso e escolher uma área de aplicação (Matemática, Português, entre outras). Depois de escolher a área devem escolher o assunto e, depois o conteúdo a ser trabalhado.

Cada grupo recebe um *template* contendo a estrutura do documento que o grupo deve preparar ao longo do semestre e entregar no encerramento da disciplina. O *template* contém as seguintes seções:

- Controle de versão do documento: como o trabalho será elaborado ao longo do semestre, em diferentes etapas, faz-se necessário que todas as etapas fiquem bem documentadas. Um controle de versão do documento é monitorado para os alunos puderem dimensionar, no final, todo o esforço dispendido;
- Índice: o documento é volumoso e vai exigir organização das ações e respectiva numeração para facilitar o acesso às informações nele contidas;
- Sumário executivo: nesta seção o grupo vai construindo como percebe e entende o contexto onde o projeto será desenvolvido. Como é a infraestrutura necessária em termos de software e hardware para desenvolvimento e uso do futuro sistema, as políticas da escola para projetos e programas educacionais, se existe e como funciona o projeto pedagógico da escola no que tange ao uso e desenvolvimento de softawre educacionais, característica da equipe de projeto e desenvolvimento e seus membros, qual o aporte teórico, em termos de Educação, que o sistema se insere e como se pretende que os profesosres e alunos o utilizem;
- Visão: Nesta parte o grupo deve ter levantado os requistos do sistema e os apresentar. Bem como, o conjunto de funcionalidades que pretende desenvolver e porque. Uma visão geral de como o sistema deve funcionar deve ser apresentada. Como os alunos e os professores irão se beneficiar do sistema;
- Definição do problema: o problema deve ser escrito de forma clara, tendo em vista o contexto. É uma redação mais objetiva;
- Objetivos do trabalho: estabelecer de forma clara e objetiva o que o sistema se propõem a fazer;
- Escopo: especificar todas as funcionalidades e o que elas fazem;
- Produtos gerados/esperados: aqui devem ser especificados os tipos de produtos que o grupo espera disponibilizar com o sistema, tais como relatórios do projeto;
- Pré-requisitos/restrições: diagramas, avaliações de desempenho do aluno e outras;
- Ítens não contemplados no escopo: importante deixar claro o que os sitema não vai fazer. Exemplo: avaliar o aluno de forma automática, sistema de ajuda sensível ao contexto e outros;
- Estrutura do projeto: aqui são apresentados os diagramas de caso de uso e de interação (fluxo da informação quando o usuário integra com o sistema);
- Papéis e assinaturas: a organização do grupo deve seguir o formato: coordenador, responsável pela documentação, responsável pela tecnologia, responsável pela interação com o especialista e redatores da documentação. Em função do tamanho do grupo alguns membros podem assumir vários papéis. A restrição é que cada membro não pode assumir mais de 2 papéis, pois prejudicaria a performance do aluno. Daí a restrição ao número mínimo de elementos no grupo;
- Referências: dados da bibliografia usada pelo grupo na elaboração do documento.

Desta forma o trabalho a ser desenvolvido na disciplina possui todas as etapas da metodologia. Ele começa de forma embrionária e, à medida que o grupo aprende os conceitos teóricos da disciplina, os vai aplicando na modelagem do sistema. Por conseqüência, o volume vai crescendo em número de páginas. A cada etapa uma versão é enviada e vai compondo a nota da disciplina.

4 Considerações Finais

Os resultados obtidos com a aplicação da disciplina em três semestres consecutivos foram muito promissores. Os alunos, em seus depoimentos colhidos através da

avaliação formal da disciplina e no decorrer da mesma, afirmaram que desta maneira conseguem dimensionar melhor o trabalho a ser feito e conseguem conversar melhor com os desenvolvedores (programadores). Os programadores, por sua vez, entendem melhor o que devem fazer. Enfim, todos ganham com a formalização e padronização das idéias. Neste contexto, tem-se buscado formalizar a metodologia de ensino relatada e a uso dos recursos através da proposta do PDS-E.

Um problema identificado se refere ao ferramental para suporte a modelagem. Os alunos têm dificuldades de usar as ferramentas tradicionais de ES. As interfaces estão em língua inglesa e, além disto, ocupam muito espaço em disco. A solução para este problema é a criação de uma ferramenta que possui as funcionalidades associadas ao tipo de trabalho desenvolvido na disciplina, isto é, que forneça suporte de forma adequada à metodologia, e seja contextualizada dentro das necessidades do público-alvo. A fim de superar esta limitação, está em fase final de desenvolvimento uma ferramenta para atender esta necessidade.

O interessante deste novo projeto é a participação dos ex-alunos da disciplina como consultores. Eles estão fornecendo subsídios para revisar e aprimorar o conjunto de funcionalidades a serem modeladas e implementados na ferramenta.

Enfatiza-se a necessidade de se utilizar documentação adequada de maneira a permitir o reuso de resultados envolvendo o projeto e desenvolvimento de softwares educacionais. Sem isto se corre o risco de se gastar tempo e recursos e não conseguir reaproveitar código e experiência da equipe. Um ponto importante a ser lembrado é que os projetos do grupo de pesquisa, na sua maioria, estão vinculados a bolsistas com tempo de permanência do projeto limitado e variável em função dos seus objetivos. Bolsistas de iniciação científica, alunos de mestrado e doutorado tem menos ou mais tempo. Porém, após a conclusão dos seus trabalhos os projetos continuam ou se estendem. A má qualidade da documentação gera um retrabalho e uma perda de informação importante e dispendiosa.

A forma de trabalhar que se propõe visa auxiliar na discussão da necessidade de se buscar uma formação mais ampla e rigorosa no que concerne ao desenvolvimento de aplicações educacionais. Pretende-se estender o projeto com a inclusão de aspectos envolvendo qualidade de software e gerência de projetos, aspectos ainda incipientes na fase atual. Porém, cruciais para se obter resultados melhores. Maiores detalhes podem ser obtidos em http://www.inf.pucrs.br/~giraffa/swedu/swedu2004.htm. Neste *link* é possível fazer *download* de exemplo de projetos construídos com a metodologia.

Referências Bibliográficas

[AUD 01] AUDY, Jorge Luis N. Modelo de Planejamento Estratégico de Sistemas de Informação: Contribuições do processo decisório e da aprendizagem organizacional. 195 f. 2001. Tese (Doutorado), PPGA – UFRGS, Porto Alegre, Brasil, 2001.

[CAR 01] CARVALHO, Ariadne M. B. R., CHIOSSI, Thelma. C. S. Introdução à Engenharia de Software. São Paulo: Editora da Unicamp, 2001, 148 p.

[COR 01] CORTES, Mario. L., et al. Modelos de Qualidade de Software., São Paulo: Editora da Unicamp, 2001 148 p.

[FER 01] FERREIRA, Alex. Análise de ferramentas de modelagem UML gratuitas. PPGCC – UFRGS, Porto Alegre/RS, 2001. Disponível em: http://www.inf.ufrgs.br/procpar/disc/cmp167/ trabalhos/sem2001-1/T1/alex/. Acessado em Junho de 2004.

[GOM 04] GOMES, Handerson Ferreira. ArgoUML. Javafree.com.br, 2004. Disponível em: http://www.javafree.com.br. Acessado em: Junho de 2004.

[IEE 93] IEEE Standards Collection: Software Engineering. IEEE Standard 610.12 – 1990, IEEE, 1993.

[McC 96] McCONNEL, Steve. Rapid Development: Taming Wild Software Schedules. EUA, Redmond: Microsoft Press, 1996. 660 p.

[PET 01] PETERS, J. F.; PEDRYCZ, W. Engenharia de Software, Teoria e Prática. Rio de Janeiro: Editora Campus, Brasil, 2001. 601 p.

[PMI 00] Project Management Institute. A guide to the project management body of knowledge (PMBOK guide). Project Management Institute. Pensylvania, EUA, 2000. 216 p.

[PRE 01] PRESSMAN, Roger S. **Software Engineering: a practitioner's approach**. EUA: McGraw Hill, 2001. 860 p.

[REP 98] REPONEN, Tapio. The Role of Learning in Information System Planning and Implementation. In: GALLIERS, H. e BAETS, R. IT and Organizational Transformation, 1998, Chichester. Proceedings... England, 1998.

[ROY 98] ROYCE, Winston. Software Project Management – a Unified Framework. EUA: Addison-Wesley, 1998. 448 p.

[SCH 00] SCHWALBE, Kathy. Information Technology Project Management. Cambridge, England: Course Technology, 2000. 561 p.

[SOM 03] SOMMERVILLE, Ian. Engenharia de Software. São Paulo: Addison Wesley, 2003. 592 p.