

METODOLOGIA PARTICIPATIVA PARA DESENVOLVIMENTO DE INTERFACE DE SOFTWARE EDUCACIONAL

Gabriela Trindade Perry M.Sc

Fernando Gonçalves Amaral Dr.

Programa de Pós graduação em Informática na Educação – PPGIE/UFRGS - Av. Paulo Gama, 110 - prédio 12105 - 3º andar sala 332 - 90040-060 - Porto Alegre (RS) – Brasil - gabrielaperry@hotmail.com

Palavras-chave: Processos de design, design de software educacional, metodologia em IHC, design participativo.

Resumo

A elaboração de *softwares* educacionais representa uma grande preocupação entre os conceitores. No entanto, os modelos existentes na literatura pouco ou nada consideram a complexidade do gerenciamento das equipes. Além disso, não apresentam maiores detalhes no que concerne sua elaboração. Este artigo tem como objetivo propor uma metodologia de desenvolvimento de *software* educacional, que permita viabilizar a estruturação e controle da integração entre os membros da equipe de desenvolvimento, considerando as características específicas deste tipo de produto. Esta metodologia é de caráter iterativo e participativo obedecendo a princípios da prototipagem.

Abstract

The development of educational softwares is a main concern between designers. However, a great part of models in literature do not consider the complexity of team management. Besides, the details concerning their elaboration are not clear. This paper aims to propose an educational software development methodology, which allows the control of integration between the members of the development team, considering the specific characteristics of this kind of product. This methodology has an iterative and participative character following the principles of prototyping.

1. INTRODUÇÃO

Ao propor uma metodologia de desenvolvimento de *software*, busca-se agilizar e sistematizar processos. Assim, normalmente, são estabelecidas metas e etapas baseadas em: métodos, formalismos, documentação e ferramentas diversas; esperando-se com isto, que os produtos alcancem um determinado patamar de qualidade. Como exemplos, pode-se citar os modelos de “Waterfall”, de Royce (1970), de caráter linear, composto de fases que vão sendo executadas sequencialmente, gerando produtos; e o modelo “Espiral”, de Boehm (1988), orientado pelos riscos (*risk-driven*), ao invés de orientado por documentação e código, como o Waterfall. Inspirados nestes, são identificadas diferentes metodologias, por exemplo: as inspiradas nas noções de prototipagem e desenvolvimento incremental que se reúnem sob a denominação de “Processos Ágeis” (Beck et al., 2001); metodologias específicas para produção de *software* educacional (Crosier et al., 2002; Burd, 1999; Pernin, 1996 e de Van der Mast, 1995) e ainda as que se designam por *HCI-enriched models*. Além destas, também existem metodologias orientadas para produção de *software* livre como a “Bazaar”, citada por Raymond (1999), na qual o código é produzido e disponibilizado na internet, à vista de todos.

A escolha de uma metodologia reflete as crenças dos líderes da equipe e o contexto de produção. Além disso, no que concerne à produção de *softwares* educacionais deve-se considerar as necessidades específicas deste tipo de produto, por exemplo: equipes multidisciplinares de produção, necessidade de avaliação de aprendizagem, adequação aos currículos e fidelidade a modelos (quando simulações).

Todavia, as metodologias existentes para produção de *softwares* educacionais não dão conta de todos estes itens, pois não consideram a complexidade do gerenciamento das equipes. Elas são também pouco elaboradas, não indicando claramente as etapas do desenvolvimento nem critérios para passar de uma fase à outra. Exemplos

podem ser encontrados em Dorrego (1994), Zambrano et al. (1995), Campos e Rocha (1996), Castro e Aguiar (2000) e Batista (1997).

No que diz respeito às bases da metodologia proposta neste artigo, dois modelos tiveram especial importância: o MARS de Pernin (1996) e o de Van der Mast (1995), ambos específicos para *softwares* educacionais e de treinamento. Sua importância se deve à defesa do estabelecimento de um ambiente que promova a interação dos profissionais envolvidos, inserido em ciclos de prototipagem.

A qualidade da integração da equipe é um aspecto determinante do sucesso do produto. Esta pode ser alcançada através de técnicas de design participativo – durante as etapas de desenvolvimento e design – e de coleta conjunta de dados (grupos focados, por exemplo) para levantamento de requisitos. A participação dos usuários, por sua vez, costuma ser em momentos de testes, sendo os métodos mais comuns as entrevistas e as observações diretas. Assim, a metodologia de desenvolvimento de *software* educacional proposta tem por objetivo viabilizar a estruturação e controle da integração entre os membros da equipe de desenvolvimento (cujos principais membros são: professores, especialistas do conteúdo, designers e programadores), considerando as características específicas deste tipo de produto.

2. SUJEITOS E MÉTODOS

A proposta metodológica para desenvolvimento de *softwares* de caráter educacional estrutura-se nas etapas descritas a seguir. A figura 1 mostra a metodologia graficamente.

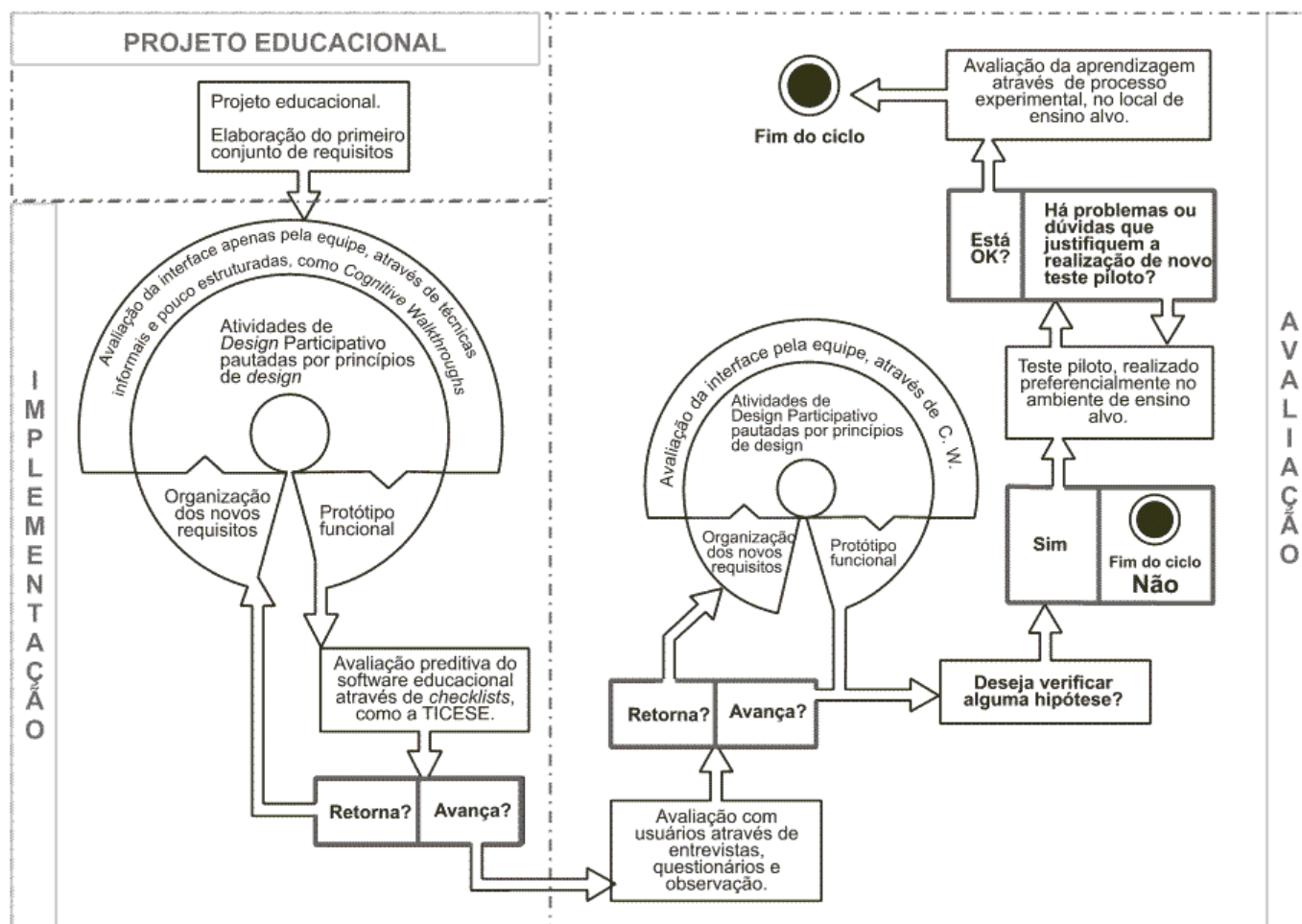


Figura 1. Representação das etapas da metodologia proposta.

2.1 Formação da equipe de desenvolvimento

Diversas áreas do conhecimento contribuem para o projeto de uma interface: educação e psicologia; ergonomia e interação homem-computador e engenharia de *software*. No entanto, somente reunir estas habilidades na equipe de produção não é suficiente; é necessária a participação dos usuários, o que torna esta tarefa bastante complexa. Sugere-se então que a equipe de desenvolvimento seja composta por: especialistas em educação, professores do domínio, designers e programadores. Também é recomendado que especialistas do domínio e consultores em estatística possam, eventualmente, prestar colaboração (figura 2). Estes devem, antes de iniciar os trabalhos, firmar o compromisso de tentar compreender os problemas colocados pelos colegas. O cumprimento deste acordo está relacionado à motivação e à personalidade de cada um. Os membros da equipe precisam dar-se conta que, caso o respeito pela opinião, pelos problemas alheios e o interesse sobre o trabalho dos colegas prevaleça, todos enriquecerão sua prática, pois estarão desenvolvendo novas habilidades. Isto é especialmente valioso caso a equipe planeje trabalhar novamente junta no futuro. O último item a ser destacado diz respeito à gerência do projeto. Caso seja considerado necessário indicar alguém para este posto, recomenda-se que seja um especialista em educação.

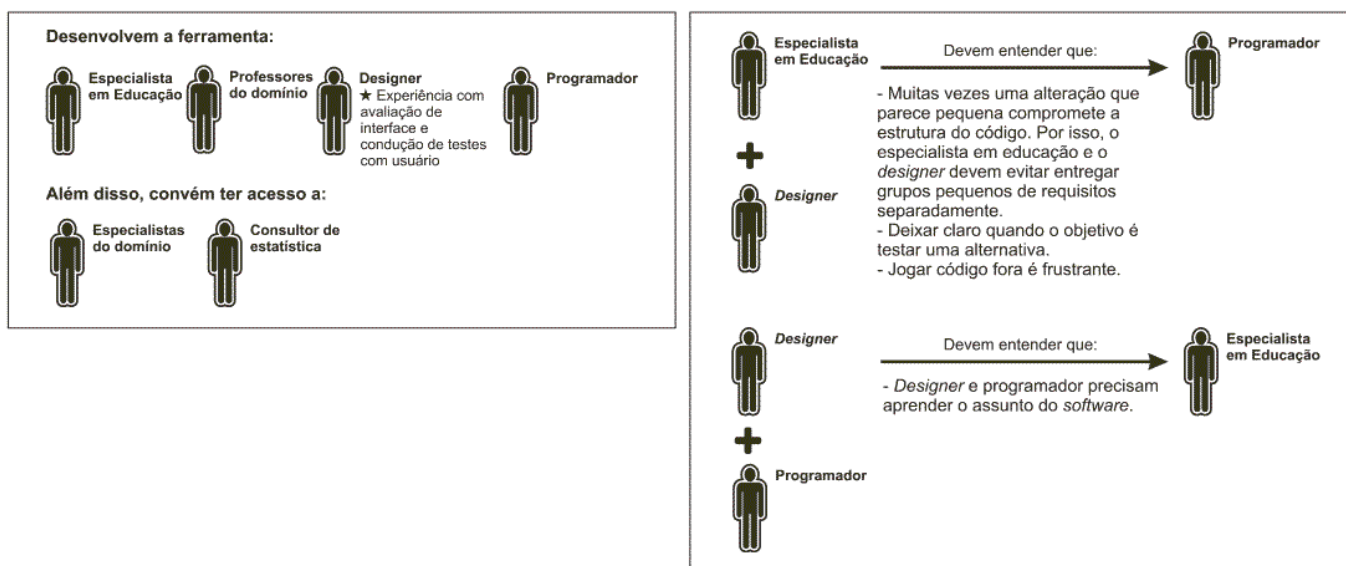


Figura 2. Sujeitos requeridos para o processo e relacionamento entre os membros da equipe.

2.2 O projeto educacional

Devido à especificidade do produto, as questões colocadas à equipe de desenvolvimento são muito peculiares. Além das conhecidas dificuldades da produção de *softwares* (incorporação de novos requisitos aos protótipos, manutenção de prazos e orçamento, comprometimento real com as etapas de teste com usuários e design da interface), somando-se uma nova dimensão: a educacional. Na verdade, esta última é o maior desafio destas equipes, por dois motivos principais. Primeiro, mesmo que uma estratégia tenha sido adequada para um determinado produto, não é garantia de que o sucesso será repetido. O segundo motivo é que muito difícil – se não impossível – simplesmente transpor uma dada teoria que oriente a prática pedagógica para o *software* a ser criado, haja visto que algumas das mais celebradas não são teorias de aprendizagem. A teoria de Jean Piaget, por exemplo, que tem auxiliado a prática docente no Brasil desde a década de 80, é uma teoria de caráter psicológico sobre a gênese do conhecimento, não sobre o ensino ou a aprendizagem. Por outro lado, as teorias focadas no ensino e na aprendizagem, como a de Ausubel (1968), foram concebidas antes do uso educacional do microcomputador. A teoria construcionista de Papert (1982), por sua vez, oferece validação teórica apenas para projetos de *software* de simulação ou de modelagem, mas desamparando outras estratégias de interação. Sustenta-se que o fato de haver uma estratégia didática não significa que o programa seja construtivista, sócio-interacionista, behaviorista, etc. Acredita-se que o significado do *software* se realiza e concretiza durante seu uso, que é definido pelo professor.

O projeto educacional é a única etapa em que não estão indicadas iterações, por ser a etapa que mais consome tempo de preparação e recursos humanos especializados. Acredita-se também que ela deva ser pouco formalizada, por causa da grande especificidade de cada projeto. A confiança nesta estratégia se deve à crença em que, se o projeto for bem feito e estruturado, o *software* terá aumentado suas chances de ser eficiente, ou seja, de facilitar e servir à aprendizagem.

Para passar à etapa seguinte, qualquer que tenha sido a forma escolhida para criar o projeto educacional, é importante que o seguinte conjunto de perguntas tenha sido respondido.

- Qual a delimitação do assunto?
- Quais os problemas dos estudantes para compreender este assunto?
- Qual a estratégia para abordar estas dificuldades?

A partir das respostas destes questionamentos, podem ser traçadas estratégias para a forma de interação com o usuário e escolha de mídias, por exemplo.

2.3 A implementação

Uma vez concluído o projeto educacional, deve-se avançar na direção da construção do primeiro protótipo. Esta tarefa acontece dentro do ciclo iterativo de implementação, no qual se realizam três tipos de atividades: especificação de requisitos, design participativo e construção do protótipo.

O ciclo tem início com as especificações geradas pelo projeto educacional. Esta atividade deve ser coordenada pelo designer e deve contar pelo menos com a participação do programador. O objetivo é projetar a aparência da interface e a forma de interação que o usuário irá experimentar. É especialmente recomendado a adoção de técnicas de design participativo, tais como CARD, PICTIVE e construção de *mock-ups*, conforme descrito em Bodker e Iversen (2002), Muller (2001) e Spinuzzi (2002). Estas técnicas fornecem estratégias para criar e para decidir entre diversas idéias e propostas de interfaces; além de permitirem que todos expressem suas opiniões. Balizar a construção destes protótipos por *guidelines*, como por exemplo, as de Lewis e Rieman (1994) e por heurísticas, como por exemplo, as de Molich e Nielsen (1990) e Bastien e Scapin (1993) também é especialmente indicado. Caso seja possível, pode-se desenvolver diversos protótipos funcionais para testar módulos separados do programa.

De qualquer forma, os protótipos gerados, sejam eles funcionais ou não, devem ser submetidos à avaliação de todos os membros da equipe, de preferência em uma sessão conjunta do tipo *cognitive walkthrough*. Além disso, é conveniente mostrar tais protótipos para professores do domínio específico, pois eles podem enriquecer com sugestões pertinentes. Também é importante, principalmente em *softwares* de modelagem e simulação, que um especialista do domínio acompanhe o desenvolvimento da interface e dos módulos que estão sendo criados. Provavelmente, esta será a etapa mais longa do processo. No momento em que nenhum dos membros da equipe tiver sugestões para alterações, deve-se passar para a próxima fase do ciclo de iteração: a construção de um protótipo funcional.

No caso da existência de um protótipo funcional, este deve ser construído pelo programador. A ele cabe a responsabilidade de escolher as ferramentas julgadas mais adequadas. Os demais membros da equipe são acionados apenas se o programador entender necessário.

Depois de concluída esta fase, passa-se para a avaliação preditiva com o uso de *checklists*, recomendando-se para tal a TICESE de Gamez (1998). Por serem ferramentas simples, podem ser aplicados mesmo por pessoas sem experiência ou poucos conhecimentos em design e ergonomia. Esta aplicação trará um resultado, que a equipe deve avaliar. Se forem encontrados problemas, deve-se voltar para o ciclo de iteração imediatamente superior. A lista de problemas identificados será o novo conjunto de requisitos que o próximo protótipo deve então atender. Por outro lado, caso o desempenho seja considerado satisfatório e se considere que não há necessidade de alterações no protótipo, passa-se à etapa seguinte: a avaliação com usuários.

2.4 Avaliação com os usuários

O objetivo desta etapa é fazer um levantamento de novos requisitos e submeter o protótipo à crítica de professores expertos no domínio. Recomenda-se, que os professores participantes desta etapa não tenham conhecimento prévio do *software*. Os membros da equipe que participam desta etapa são: o designer e o especialista em educação; este último orientando o primeiro na confecção do material de apoio. Sugere-se que o levantamento de sugestões e de novos requisitos seja feito através de entrevistas. Estas podem ser conduzidas por qualquer um dos membros da equipe que se sinta em condições de ouvir os professores especialistas, pois, nestas

reuniões, podem surgir sugestões que necessitam de uma compreensão mais acurada. Deve-se considerar que, durante a entrevista, o profissional pode não se comportar como um professor, e, portanto, não considerar o entrevistador como aluno. Em seguida, deve-se tabular estas sugestões e críticas, o que provocará modificações no protótipo, iniciando-se mais uma vez o ciclo de iteração da etapa de implementação.

Em alguns casos, pode haver a necessidade de realização de experimento para teste de hipótese. Estas podem estar relacionadas com o design da interface; a interação com o usuário; o projeto educacional, etc. É imprescindível que a equipe tenha então consciência de que projetar, executar e interpretar dados em um experimento é uma tarefa extremamente cara, trabalhosa e demorada; necessitando do acompanhamento de sujeitos com habilidades muito especializadas. Para este fim, conta-se com um consultor em estatística, auxiliado pelo especialista nas variáveis em estudo (por exemplo, o designer, caso o experimento seja sobre a forma de interação da interface). Entre os custos de realização de um experimento cita-se (a) tempo despendido no projeto e na elaboração de materiais para coleta de dados e do manual para análise dos resultados; (b) necessidade de condução de teste piloto, para testar o *software* e o material de coleta; (c) dificuldade de conseguir um professor de uma classe com número suficiente de estudantes, que disponha de tempo e que incentive os estudantes a participar do teste e, finalmente, (d) tempo gasto na análise de dados. Deve-se ainda ter o cuidado de documentar amplamente esta atividade, tendo em vista inspeções pelo Comitê de Ética do órgão ao qual se está filiado ou que financia a pesquisa.

3. CONCLUSÃO

A necessidade de estruturar uma metodologia para desenvolvimento de software educacional surgiu da constatação a partir de revisão bibliográfica, que nenhuma das metodologias existentes contempla o conjunto de aspectos considerados relevantes pelos autores: definição clara das etapas; ordem das atividades; marcos para finalização de cada etapa e passagem para a seguinte e distribuição de tarefas entre os membros da equipe (Campos e Rocha, 1996 e Dorrego, 1994). Outro critério importante foi o fato de que a maioria das metodologias pesquisadas, talvez por terem sido elaboradas por profissionais da informática, davam ênfase muito maior às etapas de produção de código e produção de conteúdo (o projeto instrucional), em relação às atividades de design da interface, não contemplando de forma satisfatória a ergonomia (Batista, 1997 e Castro e Aguiar, 2000).

Pela análise desses trabalhos, percebe-se o designer colocado em uma função de produção estética, sendo o responsável pela boa aparência da aplicação. Na opinião dos autores, esta abordagem não possibilita o pleno exercício das capacidades dos profissionais de design. Fala-se em designers acumulando as funções de produção estética e análise ergonômica da interface, porque a figura do especialista em ergonomia sequer foi citada. Considerando-se a incipiente produção de softwares educacionais no Brasil, este cenário justifica-se. Contudo, a necessidade de atender às demandas relacionadas ao design da interface precisam ser atendidas, sendo que estas iniciativas estão concentradas, no Brasil, especialmente nos programas de Engenharia de Produção.

No cenário atual, certamente diversos aspectos positivos foram encontrados e o conjunto de metodologias pesquisado forneceu material para permitir a elaboração da proposta deste artigo. O modelo proposto por Zambrano et al. (1995), por exemplo, traz a etapa de avaliação do software educacional extremamente bem detalhada, constituindo-se um bom guia de inspeção. Em Crosier et al. (2002), encontra-se uma metodologia muito detalhada, mas com pouca ênfase aos aspectos ergonômicos. As maiores inspirações foram os benchmarks destas propostas – o MARS de Pernin (1996) e o modelo de van der Mars (1995) – cujos aspectos principais foram combinados para atender ao conjunto de requisitos exposto anteriormente, e para viabilizar a estruturação e controle da integração entre os membros da equipe de desenvolvimento, considerando as características específicas dos softwares educacionais, objetivos plenamente alcançados.

Além disso, esta metodologia foi validada pela construção de um *software* educacional e subsequente realização de uma avaliação experimental da aprendizagem, envolvendo o uso do *software* em um ambiente de ensino presencial.

Como indicação para futuros estudos sugere-se que seja determinado, de forma objetiva, o impacto de cada etapa da metodologia e a importância da presença de cada tipo de profissional, em relação ao processo de desenvolvimento e à qualidade educacional do *software*.

REFERÊNCIAS

- AUSUBEL, D. P. **Educational psychology : a Cognitive View**. Holt, Rinehart and Winston: New York, 1968.
- BASTIEN, J. M. C.; SCAPIN, D. L. **Ergonomic Criteria for the Evaluation of Human-Computer Interfaces**. Relatório de Pesquisa n° 0156. INRIA. Institut National de Recherche en Informatique et en Automatique. Rocquencourt, França, 1993. Disponível em: <<http://www.inria.fr/rrrt/rt-0156.html>>. Acesso em: 15 março 2004.
- BATISTA, J.; FIGUEIREDO, A. D.: **Desenvolvimento de programas educativos por prototipagem continuamente evolutiva**. In: 2° Simpósio Investigação e Desenvolvimento de *Software* Educativo, 1997, Coimbra. Disponível em: <<http://lsm.dei.uc.pt/simposio/pdfs/c07.pdf>>. Acesso em 10 novembro de 2004.
- BODKER S.; IVERSEN, O. S. Staging a Professional Participatory *Design* Practice – Moving PD Beyond the Initial Fascination of User Involvement. In: **NordiCHI'02**, Arhus: ACM, p. 11-18, 2002.
- BORGHOLM, T.; MADSEN, K. H. Cooperative usability practices. **Communications of the ACM**, v. 42 n. 5, p. 91-97, 1999.
- BURD, L. **Desenvolvimento de Softwares para Atividades Educacionais**. Campinas: UNICAMP, 1999. 241 p. Dissertação (mestrado). NIED. Núcleo de Informática na Educação, Universidade de Campinas, 1999.
- CAMPOS, F. C.; ROCHA, A. R. C. **Dez Etapas para o Desenvolvimento de Software Educacional Multimídia**. In: 3° Congresso Iberoamericano de Informática Educativa. RIBIE, 1996, Barranquilha.
- CASTRO, G. C. M.; AGUIAR, T. C. **Engenharia de Software no Processo de Desenvolvimento de Software Educacional Multimídia**. In: XXV Conferência Latino-Americana de Informática. CLEI'00, 2000, Assunção.
- CROSIER, J. K.; COBB, S.; WILSON, J. R. Key Lessons for the Design and Integration of Virtual Environments in Secondary Science. **Computers & Education**, v. 38, p. 77-94, 2002.
- DORREGO, E. **Modelo Para la Producción y Evaluación Formativa de Medios Instruccionales, Aplicado al Video y al Software**. In: RIBIE'94 – 2° Congresso Iberoamericano de Informática Educativa, Lisboa. Disponível em <<http://www.c5.cl/ieinvestiga/ribie94.htm>>. Acesso em 04 agosto de 2004.
- GAMEZ, L. **TICESE: Técnica de Inspeção de Conformidades Ergonômicas em Software Educacional**. Dissertação (mestrado). Universidade de Minho, 1998.
- LEWIS, C.; RIEMAN, J. Task-Centered User Interface Design: A Practical Introduction, 1994. Disponível em <<http://hcibib.org/tcuid/>> Acesso em 02 fevereiro de 2004.
- MOLICH, R.; NIELSEN, J. Improving a Human-Computer Dialogue: What designers Know About Traditional Interface Design. **Communications of the ACM**, v. 33, p. 338-342, 1990.
- MULLER, M. J.: Layered Participatory Analysis: New Developments in the CARD Technique. In: **CHI'01 - Conference on Human Factors in Computer Systems**. New York: ACM, 2001, pp. 90-97.
- MYERS, B. A. Challenges of HCI Design and Implementation. **Interactions**, p. 73-83, jan. 1994.
- PAPERT, S. **Mindstorms: Children, Computers, and Powerful Ideas**. Basic Books: New York, 1982.
- PERNIN, J. **M.A.R.S.: Un Modèle Opérationnel de Conception de Simulations Pédagogiques**. Grenoble : Université Joseph Fourier, 1996. 276 p. Tese (Doutorado). Laboratoire CLIPS. IMAG, Université Joseph Fourier, 1996.
- RAYMOND, E. S. **The Cathedral & the Bazaar**. O'Reilly, 1999.
- SPINUZZI, C. A Scandinavian Challenge, a US Response: Methodological Assumptions in Scandinavians and US Prototyping Approaches. In: **SIG'02**. Ontario. Canada, p. 208-215.
- VAN DER MAST, C. A. P. G.: **Developing Educational Software: Integrating Disciplines and Media**. Delft: Technische Universiteit Delft. 276 p. Tese (Doutorado). Technische Universiteit Delft, 1995.
- ZAMBRANO, J. **Enseñanza Asistida por Computador y Producción de Software Educativo (PROSDOS)**. Caracas, Venezuela: Imprenta Universitaria Universidad Central de Venezuela, 1995.