em teia
Revista de Educação Matemática e Tecnológica Iberoamericana
Universidade Federal de Pernambuco
ISSN 2177-9309
Pós-Graduação em Educação Matemática e Tecnológica
EDUMATEC

CONTRIBUTOS PARA O DESENVOLVIMENTO DE *SOFTWARE* EDUCATIVO TENDO POR BASE PROCESSOS CENTRADOS NO UTILIZADOR

António Pedro Costa
Centro de Investigação em Didática e Tecnologia na Formação de Formadores (CIDTFF) da Universidade de Aveiro e Ludomedia – Aveiro – Portugal
apcosta@ua.pt/pcosta@ludomedia.pt

Estela Barreto da Costa
Ludomedia – Aveiro – Portugal
ebarreto@ludomedia.pt

Resumo

O desenvolvimento de *software* educativo deve ter por base as teorias de aprendizagem, o tipo de recursos e as metodologias de desenvolvimento. Estes princípios têm como intuito garantir a qualidade necessária que este tipo de recurso educativo necessita quando explorado no processo de ensino e aprendizagem. Por outro lado, o desenvolvimento de *software* educativo exige um investimento na constituição de equipas com competências diversificadas capazes de dar resposta aos requisitos pedagógicos e técnicos. O uso de *software* educativo para a Matemática desperta e estimula os alunos para a aprendizagem da disciplina, fazendo, por vezes, com que aqueles que antes a detestavam passem a sentir mais prazer e motivação ao estudá-la. No entanto, existem no mercado muitos pacotes de *software* educativo para a Matemática de qualidade duvidosa e como tal torna-se peremptório discutir como estes recursos podem ser desenvolvidos. Neste âmbito, apresentamos um processo de desenvolvimento simples, iterativo e incremental tendo como "alicerces" princípios do design centrado no utilizador, bem como práticas e valores dos métodos ágeis de desenvolvimento de software. Este processo será extremamente útil para garantir a qualidade dos pacotes de *software* educativo para a Matemática.

Palavras-Chave: Metodologias de Desenvolvimento de *Software* Educativo. Engenharia de *Software*. Métodos Ágeis. Metodologia Híbrida de Desenvolvimento Centrado no Utilizador. Design Centrado no Utilizador.

CONTRIBUTIONS TO EDUCATIONAL SOFTWARE DEVELOPMENT METHODOLOGY BASED ON USER CENTERED DESIGN

Abstract

The development of educational software should be based on learning theories, the kind of resources and the development methodologies. These principles have the intention to ensure the necessary quality that this type of educational resource needs when exploited in the process of teaching and

learning. On the other hand, the development of educational software requires an investment in building teams with diversified skills able to meet the educational and technical requirements. The use of educational software for mathematics awakens and stimulates students to learn this discipline making, sometimes, those who hated the discipline before, to start feeling more pleasure and motivation to study it. However, there are many packages on the market of mathematics educational software with very dubious quality and, as such, it becomes imperative to discuss how these resources can be developed. In this context, we present a simple, iterative and incremental development process having as its basis principles of user centered design, as well as values and practices of agile software development. This process may be very valuable to ensure the quality of software packages for teaching mathematics.

Keywords: Educational Software Development Methodologies. Software Engineering. Agile Methods. Hybrid User Centered Development Methodology. User Centered Design.

INTRODUÇÃO

Em muitos países, os governos têm investido continuamente em *hardware*, de forma a munir as instituições de ensino com as mais recentes tecnologias, partindo do princípio que estas serão, por si só, o "motor" do ensino e aprendizagem. A maioria destas decisões, por vezes, não tem por base qualquer estudo que caracterize e analise o contexto de utilização destas tecnologias, não passando de uma medida meramente política "ocultando" as medidas e estratégias que a educação realmente carece. Contudo, as tecnologias como os *tablets*, computadores, entre outras, são artefactos potenciadores de aprendizagens significativas, levando os profissionais da educação a procurar conteúdos generalistas ou específicos. Por diferentes fatores, entre eles, a falta de competências de análise e avaliação por parte dos profissionais da educação no que confere aos requisitos técnicos e pedagógicos de determinado *software* educativo, leva-nos a concordar com Ramos e colaboradores (2005) que a condicionante atrás referida leva, por um lado, à aquisição de conteúdos de qualidade duvidosa e por outro ao pouco cuidado dos editores na garantia da sua qualidade. Desta forma, o desenvolvimento de *software* educativo de qualidade implica uma avaliação formativa dos protótipos concebidos, pelas equipas, durante o processo de desenvolvimento (LOUREIRO, 2002; COUTINHO; CHAVES, 2001; GOMES, 2000). A importância de se proceder à avaliação do *software* educativo prende-se a várias razões, entre as quais: a inevitabilidade da sua utilização em contexto educativo; a "facilidade" que existe, atualmente, na sua produção; a capacidade económica das empresas produtoras; e a homogeneidade das equipas que os desenvolvem (RAMOS *et al.*, 2005). Independentemente do tipo de avaliação, dificilmente será mensurável se não forem envolvidos os utilizadores finais. É, na realidade, através da observação, análise e avaliação da reação dos destinatários que se processam as devidas correções ou adaptações aos pacotes de *software*.

Pretendemos neste ensaio efetuar uma breve abordagem aos processos de desenvolvimento de *software* educativo na produção e na garantia da qualidade dos conteúdos para a Matemática através da exploração dos dispositivos tecnológicos que "invadem" atualmente as instituições de ensino. A nossa pretensão é sensibilizar e discutir que os processos de desenvolvimento (aliados às teorias de aprendizagem e ao tipo de recursos) não resolverão todos os problemas, mas permitirão garantir a qualidade dos pacotes de *software* educativo que são necessários e disponibilizados ao serviço da educação.

METODOLOGIAS DE DESENVOLVIMENTO DE *SOFTWARE* EDUCATIVO

O desenvolvimento de *software* educativo é uma atividade de elevada complexidade, sendo que na maioria dos casos ocorre sem ser devidamente planeada, suportada por decisões de "curto prazo" (FOWLER, 2005) com a necessidade de obter resultados de forma rápida.

Shneiderman e Plaisant (2005) afirmam que 60% dos projetos de desenvolvimento de *software* falham na definição dos objetivos. Este problema surge nomeadamente porque, na maioria dos projetos, existe falha (ou simplesmente não existe) de comunicação entre os vários elementos da equipa e entre elementos da equipa e os utilizadores finais.

Os primeiros métodos/metodologias[1] (designados na literatura como disciplinados, tradicionais ou clássicos) derivaram do processo mais comum do desenvolvimento de *software*. O método em cascata de água ou ciclo de vida de desenvolvimento de *software* surgiu nos anos 70 e é-lhe atribuído o facto de servir de base teórica para outros métodos, sendo por vezes designado como um método genérico para o desenvolvimento de *software* (SOMMERVILLE, 2007). Porém, segundo Larman e Basili (2003), o método iterativo e o método incremental já remontam os anos 50, existindo exemplos concretos de projetos nos anos 70. Nos anos 80 surgiram, entre outros, os métodos em espiral e de prototipagem, e nos anos 90 os métodos ágeis, exemplos reais de integração das abordagens iterativas e incrementais (ABBAS; M. GRAVELL; WILLS, 2008; PAELKE; NEBE, 2008; SOMMERVILLE, 2007; BOEHM; TURNER, 2003; LARMAN; BASILI, 2003; MIGUEL, 2003; BOEHM, 2002). Os métodos ágeis, como o Extreme Programming (XP), o Scrum e o Crystal Clear, através do envolvimento dos utilizadores no processo de desenvolvimento, procuram fornecer e dar prioridade aos novos requisitos do *software* e avaliar as iterações do mesmo. Dão enfoque ao papel das pessoas, sendo que as competências da equipa de

[1] Segundo os autores Jiang e Eberlein (2008), métodos e metodologias são termos que têm sido utilizados no desenvolvimento de *software* com o mesmo significado, porém em contextos diferentes.

desenvolvimento devem ser reconhecidas e exploradas. Os elementos da equipa são livres para utilizar os seus próprios métodos de trabalho sem serem prescritos processos (PAELKE; NEBE, 2008; PETERSEN, 2008; SOMMERVILLE, 2007; BERGIN *et al.*, 2004; KEITH, 2002; BECK, 2000).

A necessidade de definir uma metodologia de desenvolvimento de *software* educativo tem como propósito minimizar/reduzir erros durante o processo de desenvolvimento e garantir a qualidade do recurso em si. Por outro lado, também permite ser um guia de orientação para todos os elementos da equipa envolvidos, possibilitando a perceção de como está a evoluir o projeto. Portanto, a escolha do método adequado para o desenvolvimento de um *software* educativo terá vantagens em termos de **qualidade** e **económicas**, mas caso seja selecionado um método menos adequado, o mais provável será o projeto ultrapassar os limites temporais, existindo falhas e consequentemente problemas económicos (TOTH, 2005).

Nos primeiros projetos de desenvolvimento de *software* educativo, partimos do princípio que constituindo uma equipa envolvendo apenas profissionais técnicos (designers e programadores) e professores da área a que se destinava o *software* educativo seria o essencial para se garantir a qualidade, acreditando ser possível antecipar a definição dos requisitos. Todavia, após o desenvolvimento de dois pacotes de *software* educativo, começamos a perceber que na educação a volatilidade dos requisitos, descrita por Toth (2005), por ser um fator difícil de quantificar, requer uma boa compreensão do domínio do problema, especificamente no que diz respeito à maturidade do utilizador e da equipa de desenvolvimento. Se o *software* não tiver precedentes ou se o problema é de uma área emergente, os stakeholders[2] não vão ter a certeza do que vai ser alterado enquanto não for colocado em prática. É neste contexto que o envolvimento dos utilizadores é importante na especificação e na definição dos requisitos, na validação dos requisitos, na verificação de protótipos (suporte descriminado do projeto e do processo de desenvolvimento), na revisão das especificações e na aceitação de produtos finalizados.

Pacotes de *software* educativo como o *Courseware* Sere (COSTA, 2012; GUERRA, 2007), o Univap Virtual (BICUDO *et al.*, 2007) e o Softvali (BENITTI; SCHLINDWEIN, 2005) foram desenvolvidos envolvendo utilizadores finais, um dos pressupostos do Design Centrado no Utilizador. Neste âmbito, será que um processo de desenvolvimento simples, iterativo e incremental tendo como "alicerces" princípios do Design Centrado no Utilizador,

[2] Indivíduos ou instituições que estão ativamente envolvidos num projeto ou cujos interesses possam ser positiva ou negativamente afetados com o resultado da execução ou da conclusão do projeto.

bem como práticas e valores dos métodos ágeis de desenvolvimento de *software* poderá garantir a qualidade dos pacotes de *software* educativo para a Matemática?

O PAPEL DO DESIGN CENTRADO NO UTILIZADOR

Atualmente, muitos pacotes de *software* educativo são desenvolvidos, tomando decisões relativamente às características e funcionalidades que deverão ser implementadas, sem envolver o utilizador (HAUSER, 2007). Assume-se, então, que é possível antecipar a definição dos requisitos tal como defendem as metodologias tradicionais/clássicas (ABBAS; M. GRAVELL; WILLS, 2008).

O Design Centrado no Utilizador serve para descrever os processos de um projeto em que os utilizadores finais têm influência na forma como este é conduzido. Alguns métodos do Design Centrado no Utilizador sondam os utilizadores sobre as necessidades que estes possuem em determinada área educacional, envolvendo-os em partes específicas do processo de desenvolvimento. Por outro lado, existem métodos em que os utilizadores têm uma maior presença, integrando a equipa, isto é, são envolvidos como elementos durante todo o processo (ABRAS; MALONEY-KRICHMAR; PREECE, 2004).

O Design Centrado no Utilizador é descrito na 9241-210 (2010) – *Ergonomics of Human-System Interaction* (210: *Human-centred design for interactive systems*). Esta norma descreve uma situação ideal onde não existem quaisquer obstáculos à utilização dos pressupostos do Design Centrado no Utilizador, excetuando a possível **falta de competências** por parte da equipa de desenvolvimento (SVANAES; GULLIKSEN, 2008). Autores como Facer e Williamson (2004), entre outros, reforçam que o Design Centrado no Utilizador é uma "metodologia" que combina, entre outros aspetos, a participação do utilizador e avaliação formativa de protótipos. Os projetos Design Centrado Utilizador são regidos por seis princípios: i) entendimento explícito de utilizadores, tarefas e ambientes; ii) constituição de uma equipa multidisciplinar; iii) a interação entre o utilizador e o sistema; iv) o envolvimento ativo dos utilizadores e; v) experiência do utilizador; vi) a iteração de soluções de projecto.

O desenvolvimento dos pacotes de *software* supracitados alicerçou-se em pressupostos do Design Centrado no Utilizador, entre os quais a constituição de equipas multidisciplinares, organizadas por profissionais da área da educação (investigadores de psicologia e pedagogia), profissionais da área da informática, especificamente da área de engenharia de *software* e programadores, designers com conhecimentos de usabilidade e por fim os professores e os alunos. É de esperar que o envolvimento alargado de equipas, com intuito de garantir a maior

qualidade, leve a que recursos educativos que têm por base o Design Centrado no Utilizador demorem mais tempo a serem desenvolvidos.

Maguire (2001) descreve que determinados métodos do Design Centrado no Utilizador poderão ser apresentados em diferentes fases (planeamento, contexto de utilização muitas vezes designado por análise, requisitos, design e avaliação) do processo de desenvolvimento de um *software* (ver tabela 1). Para cada método a autora apresenta um resumo, objetivos, benefícios, quando aplicar, a duração média e a abordagem ao método.

Tabela 1 – Métodos Design Centrado no Utilizador, adaptado de Maguire (2001)

Planeamento	**Contexto de Utilização**	**Requisitos**	**Design**	**Avaliação**
•Planeamento e propósito da usabilidade •Análise do custo/benefício da usabilidade	•Identificação dos *stakeholders* •Análise do contexto de uso •Levantamento dos utilizadores existentes •Campo de estudo/observação do utilizador •Diário de manutenção (*diary keeping*) •Análise de Tarefas	•Análise do *Stakeholder* •Análise do custo/benefício do utilizador •Entrevista de levantamento de requisitos do utilizador. •*Focus groups* •Cenários de utilização •Personas •Análise da concorrencial do atual *software* •Mapeamento de tarefas/funções •Alocação de funções •Utilizador, requisitos organizacionais de usabilidade	•Chuva de ideias •*Parallel design* •Orientações e normas •*Storyboarding* •*Affinity diagram* •*Card sorting* •Protótipos em papel •Protótipos de *Software* •*Wizard-of-Oz prototyping* •*Organizational prototyping*	•Avaliação Heurística (*Heuristic Evaluation*) •*Controlled User Testing* •Questionários de Satisfação (*Satisfaction questionnaires*) •Avaliação da Carga Cognitiva (*assessing cognitive workload*) •Incidentes críticos (*critical incidents*) •Experiência pós-entrevistas (*post-experience interviews*)

A nossa proposta não passa por aplicar todos os métodos apresentados na tabela anterior, mas selecionar aqueles que se coadunem com a equipa. Vredenburg, Mao, Smith e Carey (2002) afirmam que a correta aplicabilidade dos pressupostos e métodos do Design Centrado no Utilizador depende da experiência dos elementos da equipa. Estes investigadores, no estudo "A Survey of User-Centered Design Practice", identificaram que os indivíduos que participaram no mesmo tinham em média 7,6 anos de experiência em Design Centrado no

Utilizador, facilitando assim a sua integração. Normalmente, as equipas são pequenas, constituídas no máximo por dezasseis elementos, sendo dois elementos os responsáveis principais pela implementação do(s) método(s). Reconhecemos que as atividades do Design Centrado no Utilizador, através da aplicabilidade dos métodos, têm um impacto significativo no desenvolvimento de um *software* educativo, melhorando não só a **usabilidade** do mesmo mas também a sua **utilidade**.

PROPOSTA DE DESENVOLVIMENTO DE SOFTWARE EDUCATIVO PASSÍVEL DE SER EXPLORADA

Quando somos confrontados com uma nova tecnologia e nos "exigem" a disponibilização de conteúdos de forma rápida e com qualidade reconhecida, devemos constituir uma equipa de desenvolvimento com experiência que consiga reduzir o tempo e consequentemente o custo de desenvolvimento, sendo estas duas das desvantagens apontadas ao Design Centrado no Utilizador (ABRAS; MALONEY-KRICHMAR; PREECE, 2004). Poderá optar-se por envolver o utilizador final (professores e alunos) apenas nas tarefas de avaliação do recurso e também submeter o recurso à avaliação por parte de peritos exteriores à equipa (COSTA, 2012; COSTA; LOUREIRO; REIS, 2010; COSTA; LOUREIRO; REIS, 2009; GUERRA, 2007), o que se considera incontornável, independentemente da metodologia adotada.

Tendo por base o que foi acima descrito, a figura 1 apresenta a Metodologia Híbrida de Desenvolvimento Centrado no Utilizador (COSTA, 2012), passível de ser utilizada no desenvolvimento de *software* educativo para a Matemática.

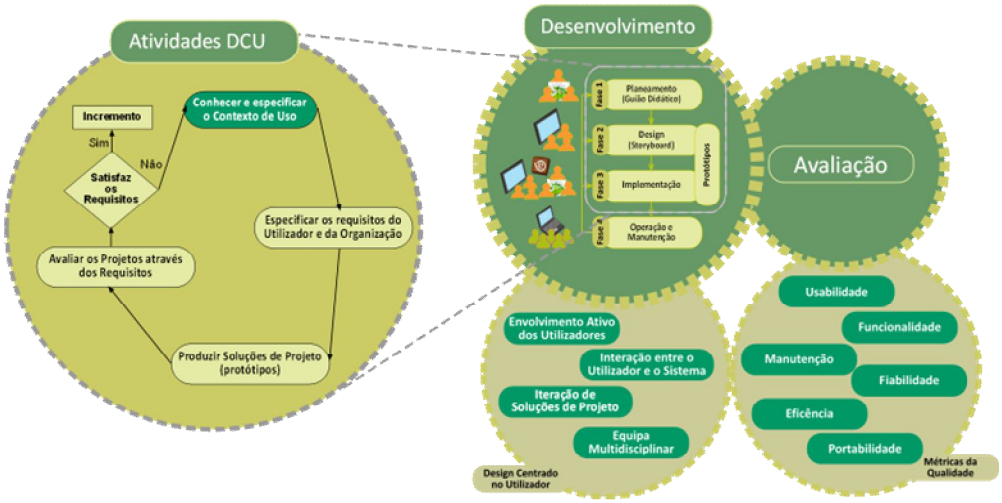

Figura 1 - Metodologia Híbrida de Desenvolvimento Centrado no Utilizador

Atendendo à figura 1, relativamente às atividades do Design Centrado no Utilizador, é importante compreender através do desenho de cenários o contexto de utilização do *software* educativo, representando algumas situações do mesmo (BASSANI *et al.*, 2006; CARVALHO, 2003). Neste âmbito, o desenvolvimento de protótipos realizado de forma colaborativa[3], envolvendo todos os membros da equipa, permite, entre outros, identificar aspetos na interface que tenham implicações na arquitetura do *software.* A prototipagem do *software* também é potenciada/usada, no processo de desenvolvimento, de forma a explorar algumas soluções de *software* em particular.

Dependendo da fase e dos requisitos, a exploração dos seguintes tipos de protótipos permitirá dar suporte à tomada de decisão por parte da equipa:

- protótipos em papel (*early paper prototypes*);
- ecrãs chave (*key screens*);
- protótipos programados (*running prototypes*).

Pretendendo-se avaliar tanto o recurso como o seu processo de desenvolvimento, a avaliação deve ser transversal a todas as fases de um processo de desenvolvimento. Mesmo neste campo, existem métodos do Design Centrado no Utilizador que permitem realizar uma avaliação equilibrando as variáveis custo-benefício. Uma boa prática são os *workshops* de avaliação com professores, em que através de observação e da aplicação de questionários se deve avaliar tanto a dimensão técnica como a proposta didática. Os professores, em grupos de dois a três elementos, exploram o *software*, por parte de um grupo heterogéneo de potenciais utilizadores do recurso (COSTA; LOUREIRO; REIS, 2009; GUERRA, 2007). A avaliação também deve ser efetuada pelos alunos que irão usar a aplicação. Esta avaliação deve sempre que possível ser efetuada em contexto de sala de aula (por exemplo, os alunos em grupos de

[3] Em projetos em que o desenvolvimento de *software* é feito envolvendo equipas multidisciplinares, o seu sucesso depende do desempenho dos elementos da equipa, como sucede em qualquer projeto que envolva interação entre pessoas (MOE; DINGSØYR; DYBÅ, 2010). As equipas que trabalham colaborativamente aumentam a possibilidade de obterem melhores resultados do que se os seus elementos atuassem de forma individual, uma vez que: i) é possível rentabilizar o mesmo trabalho com base no esforço e competências de cada elemento (FUKS; RAPOSO; GEROSA, 2002); e ii) os elementos podem identificar antecipadamente inconsistências e falhas que decorrem durante o processo de desenvolvimento. Colaborativamente, a equipa pode debater ideias e resolver problemas detetados, além de facilitar o processo criativo, surgindo mais soluções de projeto para os requisitos do *software*, analisando desta forma e em conjunto as vantagens e desvantagens, antes do incremento de novas soluções de projeto (TUROFF; HILTZ, 1982). O facto de um elemento de uma equipa receber *feedback* dos outros elementos quando da disponibilização, por exemplo, de uma versão de um documento, permite que o mesmo seja mais reflexivo e pode levar ao seu melhoramento (BENBUNAN-FICH, 1999) incrementando assim a qualidade do *software* educativo.

três a quatro elementos exploram as atividades do *software*), tratando-se de uma avaliação controlada (COSTA; LOUREIRO; REIS, 2010a). É possível melhorar o *software* e implementar novas funcionalidades através de novos requisitos que são detetados durante este processo (MIGUEL, 2003; SOMMERVILLE, 2007).

PROCEDIMENTOS E TÉCNICAS

Para agilizar o processo de desenvolvimento e partindo do princípio que o trabalho colaborativo decorre simultaneamente em dois estados, presencial e *online*, o Procedimento de Verificação e Validação é essencial, como representa o *workflow* da figura 2. Este procedimento surge integrado numa das atividades do Design Centrado no Utilizador, Produção de Soluções de Projeto (protótipos), antecedendo a fase de avaliação destas soluções, com o utilizador final ou peritos. De suporte a estas atividades pode ser utilizado um *software* colaborativo (*groupware*) que permita a equipa gerir o processo de desenvolvimento, agilizando a comunicação entre os elementos da equipa multidisciplinar, para disponibilizar documentos, debater ideias, entre outros.

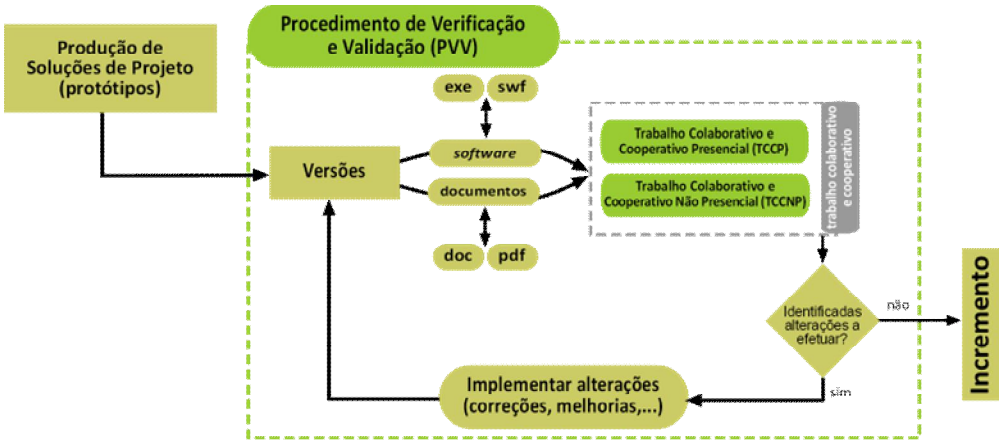

Figura 2 – Procedimento de Verificação e Validação

No Procedimento de Verificação e Validação, compete aos elementos da equipa multidisciplinar efetuar a verificação e validação das versões do *software*, como das versões dos respetivos documentos. Sendo identificadas alterações a efetuar, é disponibilizada no *groupware* uma nova versão, para verificação e validação. Estas iterações apenas terminam quando não se identificam alterações a efetuar.

No Trabalho Colaborativo e Cooperativo Presencial, comummente, deve ser o gestor de projeto que efetua um primeiro levantamento dos pontos a serem discutidos na reunião

presencial. Estes pontos devem ser ordenados por importância ou áreas de atuação, sendo enviados previamente para os elementos da equipa multidisciplinar (ver figura 3).

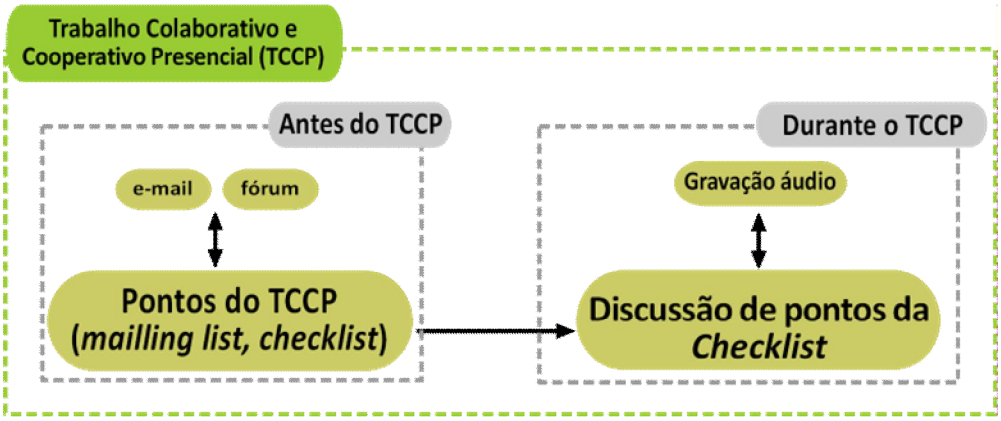

Figura 3 – Trabalho Colaborativo e Cooperativo Presencial

Do Trabalho Colaborativo e Cooperativo Presencial, ao serem identificadas alterações a efetuar, as mesmas são disponibilizadas no *groupware*. Neste contexto, as ferramentas (recursos, módulos de atividades e blocos) utilizadas (ver figura 4) no Trabalho Colaborativo e Cooperativo Não Presencial permitem promover e agilizar uma maior interação entre os elementos da equipa multidisciplinar.

Figura 4 – Ferramentas de Comunicação, Coordenação e Colaboração e Cooperação

As ferramentas apresentadas na figura 4 devem ser disponibilizadas evolutivamente, com o surgimento da necessidade de tornar o trabalho colaborativo e cooperativo mais fácil.

CONSIDERAÇÕES FINAIS

Embora a seleção do método dependa do ambiente em que se insere o projeto e de um conjunto de variáveis que, por vezes, são de difícil definição, os métodos existem para tentar auxiliar no desenvolvimento de *software*, minimizando a incerteza, de modo a permitir a obtenção do resultado esperado da forma mais eficiente possível. Tal como Toth (2005) e Sommerville (2007), consideramos que a adoção do mesmo método para todos os projetos de desenvolvimento de *software* educativo dificilmente será uma boa escolha, se tivermos em conta a diversidade de utilizadores, o objetivo da utilização do *software* e as alterações constantes da tecnologia. Também concordamos com Abbas, Gravell e Wills (2008), que designam o desenvolvimento de *software* como uma atividade imprevisível, sendo necessário um método adaptável para controlar esta imprevisibilidade.

Neste contexto, relativamente ao desenvolvimento de *software* educativo, os processos iterativos e incrementais associados a procedimentos de prototipagem, incluindo ferramentas de avaliação e monitorização nas diferentes fases, são uma forma eficiente de um processo se adaptar à mudança constante de requisitos e da tecnologia (COSTA, 2012; COSTA; LOUREIRO; REIS, 2010b; COSTA; LOUREIRO; REIS, 2010c; COSTA *et al.*, 2009). Paralelamente, também revemos nas palavras de Abras, Maloney e Preece (2004), que duas das desvantagens do Design Centrado no Utilizador são a dos projetos necessitarem de mais tempo para serem desenvolvidos e consequentemente serem mais dispendiosos. Contudo, e segundo Shneiderman e Plaisant, os métodos do Design Centrado no Utilizador permitem que o *software* gere menos problemas durante o desenvolvimento e que se reduzam os custos na fase de manutenção (2005, p. 118), fase esta que é considerada a mais dispendiosa do ciclo de vida de um *software* (DUIM; ANDERSSON; SINNEMA, 2007).

Referências

ABBAS, N.; M. GRAVELL, A.; WILLS, G. B. *Historical Roots of Agile Methods: Where did "Agile Thinking" Come from?* (A. processes and eXtreme programming in S. Engineering, Org.). Limerick, Irlanda: [s.n.], 2008

ABRAS, C.; MALONEY-KRICHMAR, D.; PREECE, J. *User-Centered Design.* In Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, p. 1–14, 2004.

BASSANI, P. S. *et al.* Em busca de uma proposta metodológica para o desenvolvimento de software educativo colaborativo. *CINTED-UFRGS. Novas Tecnologias na Educação*, Porto Alegre, v. 4(1), p. 1–10, 2006.

BECK, K. *Extreme Programming Explained: Embrace Change*. [S.l.]: Addison-Wesley, 2000. Disponível em: <http://books.google.com/books?id=G8EL4H4vf7UC&dq extreme+programming+explained+embrace+change&printsec=frontcover&source=bn&hl=pt-PT&ei=d24BS8mnHMa04QbhidWBDA&sa=X&oi=book_result&ct=result&resnum=4&vd=0CBcQ6AEwAw#v=onepage&q=&f=false>.

BENBUNAN-FICH, R. Impacts of Asynchronous Learning Networks on Individual and Group Problem Solving. A Field Experiment. Group Decision and Negotiation, p. 409–426, 1999.

BENITTI, F. B. V.; SCHLINDWEIN, L. M. Processo de Desenvolvimento de Software Educacional: proposta e experimentação. *CINTED-UFRGS. Novas Tecnologias na Educação*, Impresso, Porto Alegre, v. 3(1), p. 1–10, 2005.

BERGIN, J. *et al. Teaching Software Development Methods: The Case of Extreme Programming*. (ACM, Org.) *SIGCSE '04*. Norfolk, Virgínia: [s.n.], 2004

BICUDO, S. F. *et al.* Projecto e Desenvolvimento de Jogos Educativos em 3 Dimensões: a experiência da Univap Virtual. 2007, [S.l: s.n.], 2007.

BOEHM, B. Get ready for agile methods, with care. *Computer*, v. 35, n. 1, p. 64-69, 2002. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=976920>.

BOEHM, B.; TURNER, R. Observations on balancing discipline and agility. *Proceedings of the Agile Development Conference, 2003. ADC 2003*, p. 32–39, 2003. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231450>.

CARVALHO, C. V. *Conceitos básicos para o desenvolvimento de cursos multimédia - Manual do Formador (1ª Edição)*. Porto: Sociedade Portuguesa de Inovação, 2003. p. 64

COSTA, A. P. *et al. Courseware Sere : Technical and Didactic Evaluation*. (R. and I. in I. I. in E. (Ed. . In A. e. a. e. R. In Méndez-Vilas, Org.) *V Conferência Internacional de Multimédia e TIC na Educação (m-ICTE2009)*. Lisboa: [s.n.], 2009

COSTA, A. P. *Metodologia Híbrida de Desenvolvimento Centrado no Utilizador*. Universidade de Aveiro, Aveiro, 2012.

COSTA, A. P.; LOUREIRO, M. J.; REIS, L. P. *Courseware Sere: Avaliação Técnica e Didáctica efectuada por Alunos*. *5ª Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI2010)*. Santiago de Compostela, Espanha: Associação Ibérica de Sistemas e Tecnologias de Informação, 2010a

COSTA, A. P.; LOUREIRO, M. J.; REIS, L. P. Development Methodologies for Educational Software: the practical case of Courseware Sere. 2009, Barcelona, Espanha: International Association of Technology, Education and Development (IATED), 2009. p. 5816–5825.

COSTA, A. P.; LOUREIRO, M. J.; REIS, L. P. Metodologia Híbrida de Desenvolvimento Centrado no Utilizador aplicada ao Software Educativo António. *Revista Ibérica de Sistemas e Tecnologias de Informação (RISTI)*, p. 1–15, 2010b. Disponível em: <http://medcontent.metapress.com/index/A65RM03P4874243N.pdf>. Acesso em: 2 nov. 2013.

COSTA, A. P.; LOUREIRO, M. J.; REIS, L. P. *Metodologia Híbrida de Desenvolvimento Centrado no Utilizador: o caso prático do Courseware Sere*. *5ª Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI2010)*. Santiago de Compostela, Espanha: Associação Ibérica de Sistemas e Tecnologias de Informação, 2010c

COUTINHO, C.; CHAVES, J. *Desafios à Investigação em TIC na Educação*: As metodologias de desenvolvimento. Braga: Universidade do Minho, 2001.

DUIM, L. VAN DER; ANDERSSON, J.; SINNEMA, M. *Good Practices for Educational Software Engineering Projects*. *Proceedings of the 29th international conference on Software Engineering*. Minneapolis, USA: IEEE Computer Society, 2007

FACER, K.; WILLIAMSON, B. *Designing educational technologies with users - A handbook from Futurelab*. Disponível em: <http://www.futurelab.org.uk/resources/documents/handbooks/designing_with_users.pdf>.

FOWLER, M. *The New Methodology*. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>.

FUKS, H.; RAPOSO, A. B.; GEROSA, M. A. Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas. 2002, [S.l: s.n.], 2002. p. 89–128.

GOMES, M. C. A. *Avaliação e Ciclo de Vidas das Aplicações Educativas*: uma proposta com base na análise no desempenho do aluno. Universidade de Coimbra, Coimbra, 2000.

GUERRA, C. *Avaliação do Storyboard e da Metodologia de Desenvolvimento do Courseware Sere*. Universidade de Aveiro, Aveiro, 2007.

HAUSER, A. UCD Collaboration with Product Management and Development. *interactions*, Impresso e Referenciado na Tese, p. 34–35, 2007.

JIANG, L.; EBERLEIN, A. Towards a framework for understanding the relationships between classical software engineering and agile methodologies. *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral - APOS '08*, p. 9-14, 2008. Disponível em: <http://portal.acm.org/citation.cfm?doid=1370143.1370146>.

KEITH, E. R. *Agile Software Development Processes* - A Different Aprroach to Software Design. Impresso, 2002.

LARMAN, C.; BASILI, V. R. Iterative and Incremental Development: A Brief History. *Computer*, Impresso, v. 36, n. 6, p. 47–56, 2003.

LOUREIRO, M. J. *Un environnement d'apprentissage informatise développe base sur des conceptions alternatives des élèves: Une application à l'enseignement de l'électricité*. Université de Mons-Hainaut, 2002.

MAGUIRE, M. Methods to support human-centred design. *International Journal of Human-Computer Studies*, v. 55, n. 4, p. 587–634, out. 2001. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1071581901905038>. Acesso em: 1 nov. 2013.

MIGUEL, A. *Gestão de Projectos de Software*. [S.l.]: FCA - Editora de Informática, 2003. p. 498

MOE, N. B.; DINGSØYR, T.; DYBÅ, T. A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, Elsevier, v. 52, p. 480–491, 2010.

PAELKE, V.; NEBE, K. Integrating agile methods for mixed reality design space exploration. *Proceedings of the 7th ACM conference on Designing interactive systems - DIS '08*, p. 240–249, 2008. Disponível em: <http://portal.acm.org/citation.cfm?doid=1394445.1394471>.

PETERSEN, R. R. *ASAP*: A Planning Tool for Agile Software Development. Pittsburgh, Pennsylvania. ACM, p. 27–31, 2008.

RAMOS, J. L. *et al. Sistema de Avaliação, Certificação e Apoio à Utilização de Software para a Educação e Formação. Cadernos SACAUSEF – Sistema de Avaliação, Certificação e Apoio à Utilização de Software para a Educação e a Formação - Utilização e Avaliação de Software Educativo*. Lisboa: Ministério de Educação, 2005.

SHNEIDERMAN, B.; PLAISANT, C. *Designing the User Interface- Strategies for Effective Human-Computer Interaction*. Fourth ed. [S.l.]: Pearson Education, 2005. p. 652

SOMMERVILLE. *Software Engineering*. Eighth Edi ed. [S.l.]: Addison Wesley, 2007. p. 840

SVANAES, D.; GULLIKSEN, J. Understanding the Context of Design - Towards Tactical User Centered Design. 2008, Lund, Sweden: ACM, 2008.

TOTH, K. *Which is the Right Software Process for Your Problem*? Impresso, 2005.

TUROFF, M.; HILTZ, S. Computer Support for Group Versus Individual Decisions. *IEEE Transactions on Communications*, v. 30, n. 1, p. 82-91, jan. 1982. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1095370>.

VREDENBURG, K. *et al.* A survey of user-centered design practice. *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, n. 1, p. 471, 2002. Disponível em: <http://portal.acm.org/citation.cfm?doid=503376.503460>.