Uma Linha de Produto de Software para Módulos de Aprendizagem Interativa

Danilo L. Dalmon[1], Leônidas O. Brandão[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Departamento de Ciência da Computação – São Paulo, SP – Brasil

{ddalmon,leo}@ime.usp.br

Abstract. *Problems during development can impact the benefits offered by educational software, such as in cases of system unexpected behaviors and long delays for fixing a defect. Interactive Learning Modules (iLM) are a family of systems that can be used embedded in Learning Management Systems and face these difficulties. This work proposes a Software Product Line (SPL) for iLM as a systematic development method, aiming to reduce the problems above. Two iLM were implemented in order to evaluate the LPS influence, which has shown improvements on perceived productivity, code quality and the programmers satisfaction.*

Resumo. *Benefícios oferecidos por aplicativos educacionais podem ser reduzidos por problemas durante seu desenvolvimento, como em casos de comportamentos inesperados do sistema e longos períodos de espera pela correção de um defeito. Módulos de Aprendizagem Interativa (iMA) são uma família de aplicativos educacionais que podem ser usados integrados em Sistemas de Gerenciamento de Cursos e enfrentam essas dificuldades. Este trabalho propõe uma Linha de Produto de Software (LPS) para iMA, como método sistemático de desenvolvimento, com o objetivo de amenizar os problemas citados. Dois iMA foram implementados para avaliar a influência da LPS, o que mostrou melhorias na percepção de produtividade, na qualidade do código produzido e na satisfação dos programadores.*

1. Introdução

Aplicativos educacionais oferecem diversas contribuições para os processos de ensino e aprendizagem. Há aplicativos que objetivam melhorar o desempenho dos alunos em exames de avaliação padronizados [Bloom 1984], influenciar pelo uso do computador a motivação e o interesse [Raines e Clark 2011], ou promover experiências pedagógicas que são muito difíceis, ou impossíveis de realizar sem a ajuda da informática [Tang 2005]. Alguns resultados de uso apresentados nesses trabalhos mostram sua efetividade em aprimorar o trabalho dos professores e as experiências dos alunos.

Desenvolver aplicativos educacionais é uma tarefa sofisticada e complexa. Além de envolver aspectos de computação e de educação, inclui outros específicos dessa interdisciplinaridade [Mor e Winters 2007]. Problemas como falta de organização e comunicação na equipe multidisciplinar [Spalter e van Dam 2003], levantamento de requisitos educacionais mal formulados [McAndrew et al. 2006] e de dificuldade de utilização por professores [Hinostroza et al. 2000], são citados na literatura.

Grande parte dos grupos de pesquisa que desenvolvem aplicativos desse tipo não utilizam métodos sistemáticos de desenvolvimento [Dalmon et al. 2012c]. Essa situação é enfrentada por muitos programadores e influencia negativamente o andamento e as contribuições dos projetos. Relatos incluem dificuldade e insatisfação do programador durante a realização das tarefas e a frequente "perda de tempo" para tratar problemas provocados por má qualidade de código e falta de documentação no desenvolvimento de aplicativos educacionais.

Ao amenizar os problemas enfrentados durante esse processo, um grupo de pesquisa pode passar a experimentar vários benefícios. Exemplos são: a redução dos tempos de desenvolvimento, manutenção e evolução, aumento da satisfação dos programadores e maior qualidade do código desses aplicativos.

O objetivo deste trabalho é aprimorar o desenvolvimento de uma família de aplicativos similares chamados Módulos de Aprendizagem Interativa (iMA) pela utilização de uma Linha de Produto de Software (LPS). iMA são aplicativos que promovem atividades interativas de maneira integrada a páginas de Sistemas de Gerenciamento de Curso (SGC) [Dalmon et al. 2011]. O desenvolvimento de iMA enfrenta diversas das dificuldades relacionadas acima. A LPS se apresenta como um método sistemático de desenvolvimento proveniente da Engenharia de Software, que pode organizar esse processo de forma a amenizar os problemas enfrentados.

Essa técnica descreve o desenvolvimento de uma família de aplicativos similares em duas etapas: (i) a engenharia de domínio, em que são criados os elementos comuns; e (ii) a engenharia de aplicação, em que os elementos comuns são reutilizados e a eles são adicionados elementos específicos para a criação dos aplicativos membros da família [Clements e Northrop 2001]. Uma LPS foi construída a partir da análise dos iMA existentes, formada por um arcabouço de aplicação que tem o papel do núcleo a ser reutilizado, um método de desenvolvimento e de documentação de código e processo.

Como contribuição para a área de Informática na Educação, este trabalho apresenta a utilização de um método sistemático para o desenvolvimento de aplicativos educacionais, aprimorando esse processo. Consequentemente, as contribuições do grupo de pesquisa e de seus aplicativos são também aprimorados. Além disso, este trabalho propõe um método de desenvolvimento, documentação e um arcabouço de aplicação que compõem a LPS.

Para avaliar a influência direta da LPS criada no desenvolvimento de iMA, essa foi utilizada para a criação de dois aplicativos: (i) o *iTangran*, que simula o brinquedo/jogo Tangran [Pedrosa e dos Santos 2004]; e (ii) a segunda versão do *iVProg* [Kamiya e Brandão 2009]. O processo de implementação desses aplicativos foi analisado de forma a explicitar as contribuições da LPS. Todos os sistemas desenvolvidos e documentação estão disponíveis como código livre[1].

Este artigo segue da seguinte maneira: a próxima seção apresenta a fundamentação teórica, com a bibliografia sobre os problemas descritos e soluções utilizadas por outros autores. A seção 3 descreve a LPS criada, cujos resultados de influência no desenvolvimento de iMA são discutidos na seção posterior. O artigo termina com algumas conclusões e sugestões de trabalhos futuros.

[1] http://ccsl.ime.usp.br/redmine/projects/ima

2. Fundamentação Teórica

O desenvolvimento de aplicativos educacionais pode ser dividido em duas grandes etapas: projeto instrucional e projeto e implementação de *software*. O projeto instrucional exige uma equipe multidisciplinar que envolve, em geral, professores, projetistas instrucionais, programadores, administradores de ambientes educacionais, responsáveis pelos currículos, pedagogos, coordenadores, entre outros [Winters e Mor 2008]. Essa variedade de profissionais pode dificultar a especificação de requisitos para a etapa de projeto de *software*. Além disso, em alguns casos esses profissionais não possuem experiência em computação, o que pode também prejudicar essa especificação e provocar mudanças frequentes nos requisitos [Spalter e van Dam 2003].

Quando o aplicativo é desenvolvido em ambiente acadêmico, há fatores adicionais influentes no projeto. Um deles é a relação do aplicativo com projetos de pesquisa, o que pode descontinuar o desenvolvimento ou o suporte uma vez que o projeto de pesquisa se encerrar. No caso em que os programadores são alunos, sua experiência em programação é geralmente pequena, e caso alcancem maturidade durante o andamento do projeto, se desligam quando terminam seus estudos, prejudicando o futuro do aplicativo. Existem processos de desenvolvimento que consideram tanto a etapa instrucional quanto a de implementação [Braga et al. 2012].

A qualidade de código também possui uma influência sobre as experiências de aprendizagem [Schleyer e Johnson 2003]. Por exemplo, um aplicativo com defeitos pode desmotivar alunos e influenciar negativamente a aprendizagem. No caso, baixa qualidade de código pode provocar dificuldades durante a manutenção, como a lentidão na atualização e correção de defeitos pelos desenvolvedores. Esses e outros problemas de qualidade computacional podem reduzir o número de usuários do aplicativo e consequentemente sua contribuição à comunidade [Mor e Winters 2007].

Um levantamento de trabalhos relacionados que apresentam aplicações de técnicas de Engenharia de Software para o desenvolvimento de aplicativos educacionais indica a existência de tendências. Antes da massificação do uso de SGC, como o Moodle e o Sakai no início dos anos 2000, encontramos mais esforços no desenvolvimento de atividades interativas usando técnicas de reúso de código, como bibliotecas e arquiteturas de componentes. A partir do uso generalizado dos sistemas de gerenciamento, os esforços se voltaram mais a elementos de aprendizagem como documentos e apresentações. Mais recentemente vemos o uso de técnicas de Engenharia de Software com reúso de código associado também ao reúso de arquitetura e de processo, tanto para o desenvolvimento de documentos quanto de atividades interativas. Um exemplo é o trabalho apresentado por Bittencourt et al. (2012). Este trabalho segue a tendência de usar uma técnica com reúso de código, arquitetura e processo, a LPS, para a criação de aplicativos de atividades interativas, os iMA.

3. Uma LPS para iMA

A LPS para iMA foi desenvolvida com as seguintes etapas: (i) análise de domínio; (ii) definição de um modelo de sistema; (iii) desenvolvimento de um arcabouço de aplicação; que formam a Engenharia de Domínio e (iv) definição do método de desenvolvimento, que corresponde à Engenharia de Aplicação. A análise de domínio define o escopo da LPS. O modelo de sistema descreve o funcionamento interno de um iMA de maneira

genérica. O arcabouço de aplicação fornece a arquitetura e o código para serem reutilizados nos iMA, e o método descreve como a LPS deve ser usada para criá-los.

3.1. Análise de Domínio

A análise de domínio foi realizada sobre os iMA existentes para a definição do escopo da LPS desenvolvida. Essa etapa resultou, basicamente, em três produtos: (i) uma lista de requisitos gerais para os iMA; (ii) a lista das funcionalidades para o usuário final, classificadas de acordo com a presença nos aplicativos existentes e possibilidade de reúso; e (iii) a modelagem das características da LPS, baseadas nesses requisitos e funcionalidades, representada por um diagrama de características.

Os requisitos gerais dos iMA existentes são: (a) funcionar em navegadores Web; (b) comunicar-se com SGC por um protocolo específico; (c) fornecer ferramenta de autoria de atividades para professores; e (d) corrigir as soluções dos alunos para atividades interativas automaticamente [Dalmon et al. 2011]. Esses requisitos, em conjunto com as funcionalidades pontuais, foram classificados com relação à presença nos aplicativos. Uma funcionalidade comum é aquela que está ou poderia estar presente em mais de um membro da família, enquanto que uma funcionalidade específica está presente em apenas um aplicativo e não pode ser incluída nos demais.

Esse mapeamento resultou na definição das características da LPS, apresentadas no diagrama da Figura 1. Existem basicamente três tipos de características: (i) as obrigatórias, presentes em todos os aplicativos, identificadas com traços simples na figura; (ii) as alternativas, que cada aplicativo possui apenas uma das possibilidades, identificadas com um arco sobre as opções; e (iii) as opcionais, em que um aplicativo pode ou não possuir, identificadas por um círculo ao fim do traço.

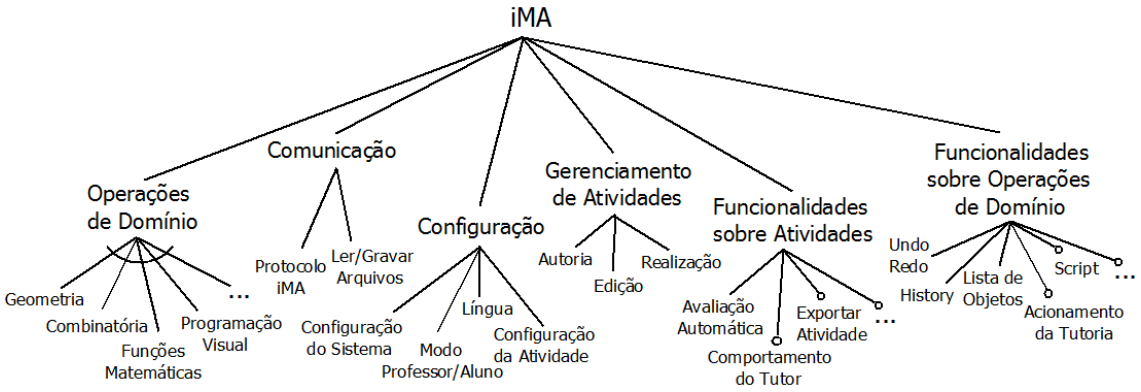

Figura 1. Diagrama de características da LPS para iMA.

No diagrama, um iMA é descrito pelas características obrigatórias: todas as de *Comunicação*, de *Configuração* e de *Gerenciamento de Atividades*, apenas uma das *Operações de Domínio*, que representam as funcionalidades específicas do domínio de cada iMA (geometria, combinatória, etc.), e quais opcionais possui das categorias *Funcionalidades sobre Atividades* e *Funcionalidades sobre Operações de Domínio*.

3.2. Modelo de Sistema

A partir das características, um modelo de sistema foi criado para definir o funcionamento interno de um iMA de maneira independente das funcionalidades específicas de cada aplicativo (as *Operações de Domínio*) e de forma a aceitar

características opcionais das categorias *Funcionalidades sobre Atividades* e *Operações de Domínio*. O resultado foi uma arquitetura genérica se baseando fortemente em Padrões de Projeto de Software [Gamma et al. 1994] como *Comando* e *Observador*.

Essa arquitetura é composta de quatro componentes: (i) *estrutural*, que oferece a base e as funcionalidades comuns; (ii) *atividades*, que define as atividades para os alunos, autoria por professores e correção automática; (iii) *domínio*, que modela genericamente os objetos e as operações de domínio; e (iv) *expansões*, que modela genericamente as funcionalidades sobre atividades ou operações de domínio [Dalmon et al. 2012b].

3.3. Arcabouço de Aplicação

Um arcabouço de aplicação foi escolhido como forma de implementação do núcleo da LPS por atender aos requisitos e características desejadas, que incluem promover reúso de código, reúso de arquitetura e não exigir conhecimentos avançados de programação para utilização por outros desenvolvedores [Guerra 2011]. Além de ser possível criar um método de desenvolvimento a partir dos pontos flexíveis do arcabouço.

O arcabouço para desenvolvimento de iMA foi derivado diretamente do modelo de sistema descrito acima. O diagrama de componentes da versão atual do arcabouço é mostrado na Figura 2. Os componentes com nome em **negrito** são parte do núcleo comum e fornecem as funcionalidades obrigatórias, aqueles em *itálico e sublinhado* têm o papel de implementar as operações de domínio, específicas para cada iMA, e os com nome em fonte `monoespaçada` representam as funcionalidades opcionais.

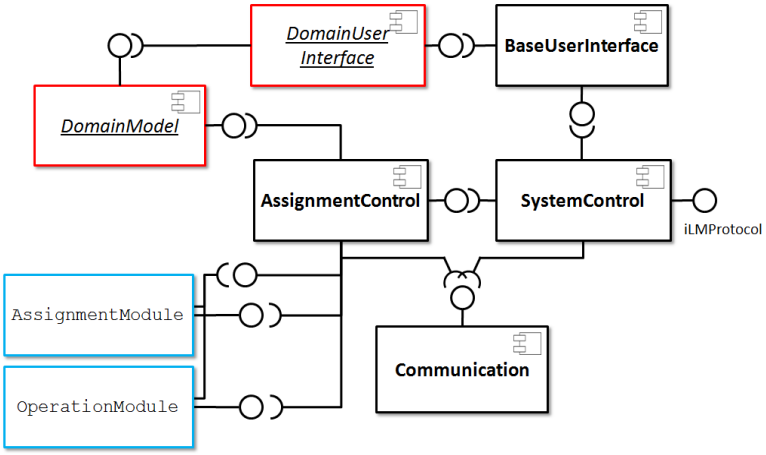

Figura 2. Diagrama de componentes do arcabouço para desenvolvimento de iMA.

Os componentes *DomainUserInterface* e *DomainModel* constituem a interface gráfica e o modelo das funcionalidades específicas de domínio, respectivamente. O componente **BaseUserInterface** é a interface gráfica básica e comum. Os componentes **SystemControl**, **AssignmentControl** e **Communication** fornecem a estrutura, o gerenciamento de atividades e o protocolo de comunicação com SGC. Por fim, o componente `AssignmentModule` modela as funcionalidades sobre atividades e o `OperationModule` aquelas sobre operações de domínio.

O processo de desenvolvimento do arcabouço foi iterativo e incremental. As primeiras iterações privilegiaram o projeto de software, enquanto que as finais o refinamento da implementação e testes. Essas características permitiram revisar as

necessidades e as soluções para o arcabouço ao longo do tempo, o que foi essencial para manter a qualidade do código e a simplicidade da arquitetura interna e do método de utilização [Dalmon et al. 2012c].

3.4. Método de Desenvolvimento – Engenharia de Aplicação

O método de desenvolvimento proposto pela LPS é o procedimento de utilização do arcabouço de aplicação apresentado acima. Esse método é documentado em uma série de manuais que o descrevem passo a passo por meio de exemplos, com o objetivo de facilitar o trabalho dos programadores. A definição desses passos evoluiu em paralelo à criação do arcabouço e vem sendo simplificada em cada iteração.

Em sua versão final, o método de desenvolvimento de iMA inclui duas etapas apenas, correspondendo à implementação dos componentes do arcabouço *DomainModel* e *DomainUserInterface*. *DomainModel* descreve os objetos de domínio e as ações que o usuário pode realizar para manipulá-los. *DomainUserInterface* é um painel no qual os objetos de domínio são representados graficamente e que dá ao usuário acesso às ações de domínio. Todas as outras funcionalidades, como as operações comuns, estrutura e comunicação são fornecidas pelo arcabouço.

Além do método de desenvolvimento de iMA, a LPS fornece um método de desenvolvimento de funcionalidades opcionais para esses aplicativos. Esse método descreve como um programador deve implementar os componentes `AssignmentModule` e `OperationModule`. Essa é a principal forma de expandir o núcleo da LPS, adicionando novas funcionalidades opcionais e comuns aos iMA [Dalmon et al. 2012c].

4. Avaliação da Influência da LPS no desenvolvimento de iMA

Para avaliar como a LPS criada pode contribuir para atingir os objetivos de aprimoramento do desenvolvimento de iMA, ela foi entregue a programadores para serem utilizadas na implementação de dois aplicativos, o *iTangran* e a segunda versão do *iVProg*. Os estudos de avaliação seguiram os métodos descritos como "prova de conceito" no caso do *iTangran* e "estudo de caso" no caso do *iVProg*.

Com uma prova de conceito é possível provar que um conceito é factível. Por exemplo, no caso deste trabalho, a prova de conceito é um iMA criado com a LPS. Ela prova que os recursos providos pela LPS permitem a criação de um iMA. Nas categorias de pesquisa descritas por Wazlawick (2009), a prova de conceito seria o resultado de uma pesquisa de apresentação de produto.

Um estudo de caso é um método de intensa análise de um fenômeno, o caso [Runeson et al. 2012]. Suas diretivas definem as etapas de planejamento, coleta e análise de dados. A coleta de dados deve incluir mais de um método de coleta, para que a análise não seja enviesada pelas características de um único. Podem ser usados métodos de coleta quantitativos ou qualitativos. Assim, como é descrito na área de Engenharia de Software, estudos de caso podem ser usados para descobrir efeitos e mecanismos causais em processos de desenvolvimento de software.

4.1. Novo iMA - *iTangran*

A prova de conceito realizada neste trabalho teve o objetivo de verificar a possibilidade de utilizar a LPS criada para o desenvolvimento de um novo iMA. Dessa forma, uma

versão em estágio avançado da LPS foi entregue a desenvolvedores em conjunto com uma série de requisitos que descreviam o aplicativo a ser criado, chamado *iTangran*. Ao longo do desenvolvimento, o autor intervia minimamente resolvendo dúvidas, corrigindo decisões realizadas com relação ao definido na LPS e eventualmente resolvendo os problemas encontrados no arcabouço.

Ao longo de dois meses, três alunos de pós-graduação trabalharam no desenvolvimento do *iTangran* durante as aulas de uma disciplina. A Figura 3 apresenta a interface gráfica principal do aplicativo criado. Essa interface tem duas partes principais: (i) a barra superior de botões; e (ii) a área central com as peças de Tangran. A barra de botões faz parte da interface gráfica básica, fornecida pelo arcabouço. As funcionalidades acessíveis por ela são independentes do domínio, como autoria de atividade, abrir e gravar arquivos, etc.

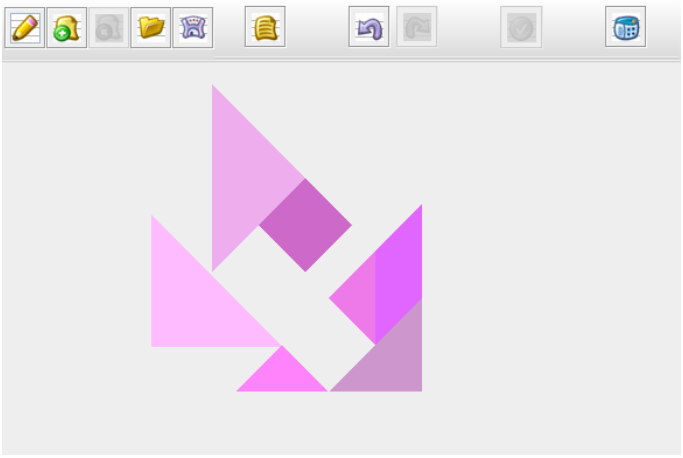

Figura 3. Interface gráfica do *iTangran* com algumas peças movimentadas.

As funcionalidades específicas de domínio são acessíveis pela área central, sendo translação e rotação das peças com o *mouse*. Uma atividade nesse iMA fornece ao aluno um desenho formado pelas peças de Tangran sem a identificação de cada uma. O aluno deve, a partir das peças na posição inicial, formar a figura indicada.

Como ao final do processo foi possível produzir um iMA utilizando a LPS criada, a prova de conceito é válida. Os relatos dos desenvolvedores sobre a influência da LPS nesse trabalho incluem a economia de tempo relacionada à adição de funcionalidades prontas, a organização do código imposta pelo arcabouço, a curva de aprendizado para utilizar a LPS e sugestões de melhoria para a arquitetura e para o método proposto pela LPS. Assim, podemos considerar que nesse caso a LPS contribuiu com a organização e o reúso de código, e a curva de aprendizado se mostrou relevante em um projeto com um tempo de execução curto.

4.2. Segunda versão do iMA *iVProg*

O *iVProg* é um aplicativo para ensino de introdução a programação [Kamiya e Brandão 2009]. Após seu principal desenvolvedor encerrar sua participação no grupo de pesquisa, um outro programador assumiu o projeto. Para esse processo de desenvolvimento, a avaliação consistiu em um estudo de caso que analisou o trabalho do programador antes e depois de adotar a LPS. Os dados coletados em cada um dos períodos consistiram principalmente de código fonte e relatos do programador sobre o andamento das tarefas.

Logo que assumiu o desenvolvimento do aplicativo, o programador do projeto foi incumbido de realizar correções de defeitos. Assim, passou a estudar o código para ser capaz de fazer as alterações. De acordo com ele, por causa de sua pouca experiência como programador e da qualidade do código aliada à falta de documentação, em cerca de 4 meses de trabalho apenas uma correção foi realizada, e apenas parte do código foi compreendido.

Ao passar a utilizar a LPS, em vez de continuar trabalhando com a versão inicial do *iVProg*, o programador começou a desenvolver outro aplicativo como uma segunda versão. Primeiro criou o núcleo com as regras de negócio e em seguida a interface gráfica. Esses passos consideraram a LPS apenas indiretamente, preparando para uma etapa posterior em que o código produzido foi integrado ao do arcabouço. Em cerca de quatro meses, a segunda versão do *iVProg* estava funcional, a Figura 4 apresenta um exemplo de atividade nesse aplicativo.

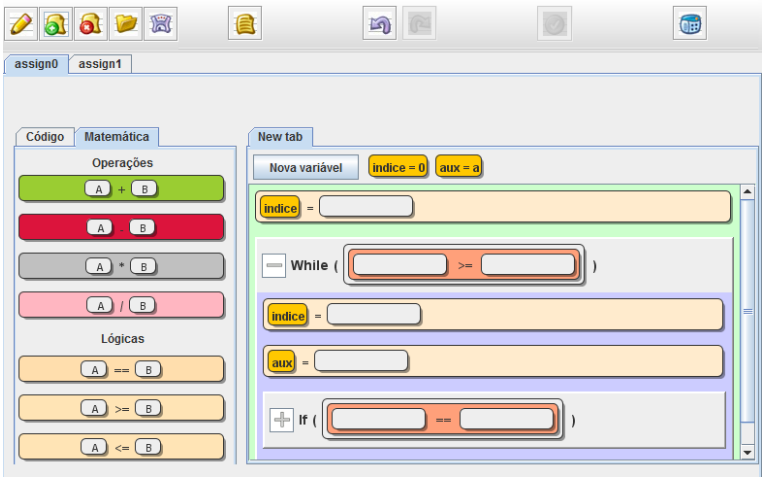

Figura 4. Interface gráfica da nova versão do *iVProg*, criada com o uso da LPS.

A interface gráfica inclui a barra de botões fornecida pelo arcabouço, já apresentada no *iTangran*, e dois painéis principais com objetos de domínio, trechos de código que podem ser arrastados do painel à esquerda (código disponível) para o painel à direita (programa sendo criado). Um aluno, nesse caso, recebe um enunciado com o programa que precisa criar arrastando esses elementos, o que eventualmente produz as saídas esperadas pelo professor.

Comparando os dados coletados antes e depois do uso da LPS, a qualidade do código produzido no segundo momento é significativamente maior, também o é a sensação de produtividade do programador. O programador relata que a arquitetura e o método fornecidos pela LPS contribuíram para seu trabalho ser organizado, para reduzir momentos de estagnação devido a dificuldades e, por fim, tornar mais agradável trabalhar sendo guiado e usando um código limpo.

O fato de passar a implementar uma nova versão do aplicativo contribuiu para o aumento da qualidade do código, possivelmente em maior grau do que a LPS. Os indícios de aprimoramento do desenvolvimento provocados pela LPS incluem a percepção de produtividade aumentada pela estrutura em passos sugerida pelo método de desenvolvimento da LPS, a qualidade do código fortalecida pela arquitetura do arcabouço e a satisfação do programador sobre o andamento de suas tarefas.

5. Conclusões

Relacionando os resultados obtidos neste trabalho com os objetivos definidos, temos indícios de que, nos casos analisados, a utilização da LPS no desenvolvimento provocou melhorias na produtividade percebida, na satisfação dos programadores e qualidade do código produzido. Essas melhorias podem permitir maior agilidade na produção de aplicativos e na sua utilização por professores e alunos, além de facilitar as tarefas de manutenção e evolução de software.

Os reúsos de código, arquitetura e processo promovido pela LPS contribuem significativamente ao desenvolvimento de iMA. O programador economiza tempo ao utilizar um código que teria de desenvolver ao não precisar pensar na estrutura de funcionamento do aplicativo. O processo facilita e guia o trabalho do programador, o que junto à qualidade do código e documentação aumenta sua satisfação.

A LPS produzida consiste em um arcabouço de aplicação, manuais de utilização, evolução e documentação. Como subprodutos da pesquisa realizada podemos citar a centralização do conhecimento anteriormente disperso nos programadores dos iMA, a divulgação do uso de método sistemático em contexto acadêmico de desenvolvimento de software educacional e a implementação de dois iMA.

Como trabalhos futuros sugerimos o aprimoramento e expansão da LPS, principalmente com relação ao seu funcionamento interno, método de utilização, número de funcionalidades oferecidas e quantidade de aplicativos que fazem parte da família de iMA. Além disso, este trabalho pode fornecer informações para investigar outras questões de pesquisa, como a relação entre qualidade de software e contribuição educacional dos aplicativos e as especificidades do contexto acadêmico para utilização de métodos sistemáticos de desenvolvimento.

Agradecimentos

Danilo Leite Dalmon foi financiado pela FAPESP, projeto 2010/06805-2. Este trabalho foi parcialmente financiado pela FAPESP (2011/10926-2) e CNPq (550449/2011-6).

Referências

Bloom, B. S. (1984) “The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring”. Educational Researcher, 13:4–16.

Bittencourt, I. I., Brito, P., Pedro, A., Isotani, S., Jaques, P. A., Rubira, C. (2012), “Desafios da Engenharia de Software na Educação: Variabilidade de Sistemas Educacionais Inteligentes e Instanciação em Larga Escala”. Anais do I Workshop de Desafios da Computação Aplicada à Educação - DesafIE.

Braga, J. C., Dotta, S., Pimentel, E., Stransky, B. (2012) “Desafios para o Desenvolvimento de Objetos de Aprendizagem Reutilizáveis e de Qualidade”. Anais do I Workshop de Desafios da Computação Aplicada à Educação - DesafIE.

Clements P. e Northrop L. (2001), “Software Product Lines: Practices and Patterns”. Addison-Wesley Professional. ISBN 0201703327.

Dalmon, D. L., Brandão, A. A. F., e Brandão, L. O. (2012a), “Uso de métodos e

técnicas para desenvolvimento de software educacional em universidades brasileiras". Anais do I Workshop de Desafios da Computação Aplicada à Educação - DesafIE.

Dalmon, D. L., Brandão, A. A. F., Isotani, S., Gomes, G. M., Brandão, L. O. (2012b), "Work in progress - a generic model for interactivity-intense intelligent tutor authoring tools". Proceedings of 42nd Frontiers in Education Conference.

Dalmon, D. L., Brandão, A. A. F., Isotani, S., Brandão, L. O. (2012c), "A Domain Engineering for Interactive Learning Modules". Journal of Research and Practice in Information Technology (aceito para publicação).

Dalmon, D. L., Tanbellini, M. J. G. S., Eisenmann, A., Nascimento, M. G., Rodrigues, P. A., Isotani, S., Brandão, A. A. F. e Brandão, L. O. (2011) "Interactive learning modules in engineering education and as a motivational tool for middle and high school students". Proceedings of International Symposium on Engineering Education.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. M. (1994), "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley Professional.

Guerra, E. M. (2010). "A Conceptual Model for Metadata-based Frameworks", Tese de Doutorado, Instituto Tecnológico de Aeronáutica.

Hinostroza, E., Rehbein, L. E., Mellar, H., Preston, C. (2000) "Developing educational software: a professional tool perspective". Education and Information Technologies, 5:103–117.

Kamiya, R. R, Brandão, L. O. (2009) "iVProg- um sistema para introdução à programação através de um modelo visual na internet". Anais do XX Simpósio Brasileiro de Informática na Educação.

McAndrew, P, Goodyear, P., Dalziel, J. (2006), "Patterns, designs and activities: unifying descriptions of learning structures". International Journal of Learning Technology, 2:126–242

Mor, Y., Winters, N. (2007). "Design approaches in technology-enhanced learning". Interactive Learning Environments, 15:61–75.

Pedrosa, C. P., dos Santos, M. H. B. P. (2004) "Reconstruindo a geometria com o tangram". Em Anais do VIII ENEM - Encontro Nacional de Ensino de Matemática.

Raines, J. M., Clark, L. M. (2011). "A brief overview on using technology to engage students in mathematics". Current Issues in Education, 14.

Runeson, P., Host, M., Rainer, A., Regnell, B., (2012). "Case Study Research in Software Engineering: Guidelines and Examples". Wiley, 1st edition.

Schleyer, T. K., Johnson, L. A. (2003), "Evaluation of educational software". Journal of Dental Education, 67:1221–1229.

Spalter, A. M., van Dam, A. (2003), "Problems with using components in educational software". Computers and Graphics, 27:329–337.

Wazlawick, R. S., (2009), "Metodologia de Pesquisa em Ciencia da Computaçao" Metodologia de Pesquisa em Ciência da Computação. Editora Campus.

Winters, N., e Mor Y., (2008), "Idr: A participatory methodology for interdisciplinary design in technology enhanced learning". Computers and Educaction, 50:579–600.