Revista Brasileira de Informática na Educação, Volume 21, Número 3, 2013

Sobre o Desenvolvimento de Software Educacional: proposta de uma Linha de Produto de Software para Módulos de Aprendizagem Interativa

On the development of Educational Software: proposal of a Software Product Line for Interactive Learning Modules

Danilo Leite Dalmon
Instituto de Matemática e Estatística
Universidade de São Paulo
ddalmon@ime.usp.br

Leônidas de Oliveira Brandão
Instituto de Matemática e Estatística
Universidade de São Paulo
leo@ime.usp.br

Resumo *O atual uso intensivo da Web para apoiar a educação tem demandado o desenvolvimento de novas soluções educacionais que proporcionem maior interatividade do aprendiz com o conteúdo e que aprimorem as atividades docentes. Isso tem implicado em sistemas mais complexos e na necessidade de uma maior integração deles com os ambientes Web. Esse é o caso dos Módulos de Aprendizagem Interativa (iMA), uma família de sistemas para atividades interativas que podem ser integrados a Sistemas de Gerenciamento de Cursos. Entretanto, problemas no processo de desenvolvimento do sistema educacional pode anular eventuais benefícios didáticos, por exemplo, quando o sistema apresenta uma falha inesperada ou quando se leva muito tempo para sanar esta falha. Para reduzir as dificuldades enfrentadas pelos programadores, este trabalho propõe uma Linha de Produto de Software (LPS) como método sistemático de desenvolvimento para iMA. Neste artigo serão apresentados o processo de construção da LPS, além de dois iMA que foram implementados para avaliar a influência da LPS. As avaliações mostraram melhorias na percepção de produtividade, na qualidade do código produzido e na satisfação dos programadores. Outro resultado é o arcabouço de aplicação, disponível na forma de software livre para alavancar a produção de novos iMA.*

Palavras-Chave: *Linha de Produto de Software, Módulo de Aprendizagem Interativa, Arcabouço de Aplicação, eLearning.*

Abstract *The current intense use of the Web to support education demands the development of new educational systems that promote better interactions between learners and contents, and also improve teachers' tasks. This has implied on more complex systems and on the need to better integrate them on Web environments. This is the case of Interactive Learning Modules (iLM), a family of educational systems that can be integrated to Learning Management Systems. However, the lack of quality on development process can disturb potential benefits of using these learning tools, such as when the system behaves unexpectedly or when it takes a long time to fix this flaw. In order to reduce these problems, this work proposes a Software Product Line (SPL) for iLM, as a systematic development method. We present the design process of SPL and the creation of two iLM used to evaluate the influence of the SPL on this process. The evaluation showed better perception of productivity, code quality and programmers' satisfaction. Another contribution is an application framework, available as free software to improve the creation of new iLM.*

Keywords: *Software Product Line, Interactive Learning Module, Application Framework, eLearning.*

Recebido: 25 de Junho de 2013 / Aceito: 28 Dezembro de 2013
DOI: 10.5753/RBIE.2013.21.03.113

1 Introdução

Aplicativos educacionais oferecem diversas contribuições para os processos de ensino e aprendizagem. Há aplicativos que objetivam melhorar o desempenho em exames de avaliação padronizados [1], influenciar pelo uso do computador a motivação e o interesse dos alunos [2], ou promover experiências pedagógicas que são muito difíceis ou impossíveis de realizar sem a ajuda da tecnologia [3]. Os resultados apresentados pelos trabalhos citados acima são exemplos da efetividade do uso da informática na educação em aprimorar tanto o trabalho dos professores quanto as experiências dos alunos.

Desenvolver aplicativos educacionais é uma tarefa complexa, além de envolver aspectos de computação e de educação, inclui outros que são específicos da interdisciplinaridade entre essas duas áreas [4]. Vários problemas relacionados a essa tarefa são citados na literatura, dentre os quais podemos destacar a falta de organização e comunicação na equipe multidisciplinar [5], levanta-mento de requisitos educacionais mal formulados [6] e dificuldade de utilização por professores [7].

Outro complicador identificado na área, este diretamente relacionado com o próprio desenvolvimento de sistemas, são as dificuldades associadas à degradação de código fonte e à baixa qualidade de software. Esses problemas devem ser enfrentados por meio de ferra-mentas e conhecimentos da Engenharia de Software [8]. Especificamente, algumas características de projetos de Informática na Educação, tornam essas dificuldades mais explícitas [9].

Programadores em projetos de desenvolvimento de aplicativos educacionais foram questionados sobre as dificuldades enfrentadas em seu trabalho cotidiano. Eles relataram mais frequentemente os seguintes problemas: a falta de documentação, tempo demais gasto para definir as próximas tarefas e, acima de tudo, dificuldades para compreender e reutilizar código legado. Grande parte dos problemas relatados é comumente tratada por técnicas e métodos da Engenharia de Software. A falta de sistematização no desenvolvimento facilita a ocorrência desses problemas [9]. Esses problemas afetam de forma negativa a influência que os aplicativos criados têm nos processos de ensino e aprendizagem [10]. Assim, aprimorar a etapa de desenvolvimento pode aumentar o impacto para o usuário final de projetos na área de Informática na Educação, influenciando os prazos de entregas, reduzindo os defeitos nos aplicativos e respondendo com maior agilidade na correção e adição de novas funcionalidades.

No artigo apresentado por Dalmon et al. [9], os autores constataram que, dentro dos grupos de pesquisa brasileiros envolvidos com desenvolvimento de sistemas educacionais, o nível de rigor no método de desenvolvimento é baixo. No referido trabalho, os autores pesquisaram artigos do Simpósio Brasileiro de Informática na Educação (SBIE) entre 2009 e 2011 e constataram que, para os trabalhos que desenvolveram software, se por um lado em 59% dos casos foram utilizadas técnicas *ad-hoc* para desenvolvimento, sem empregar métodos explícitos (o que foi considerado como nível 1), por outro lado apenas 14% dos trabalhos estavam fortemente amparados por técnicas de Engenharia de Software (nível 4). A Figura 1 apresenta esses dados e outros relevantes para compreender o contexto em que ocorrem esses desenvolvimentos. No gráfico b), está a distribuição do número de desenvolvedores participantes no projeto. No gráfico c) está a distribuição por anos de experiência dos desenvolvedores participantes nos projetos. Nota-se que 57% dos projetos tem até dois (2) programadores e 55% deles possui três (3) anos ou menos de experiência.

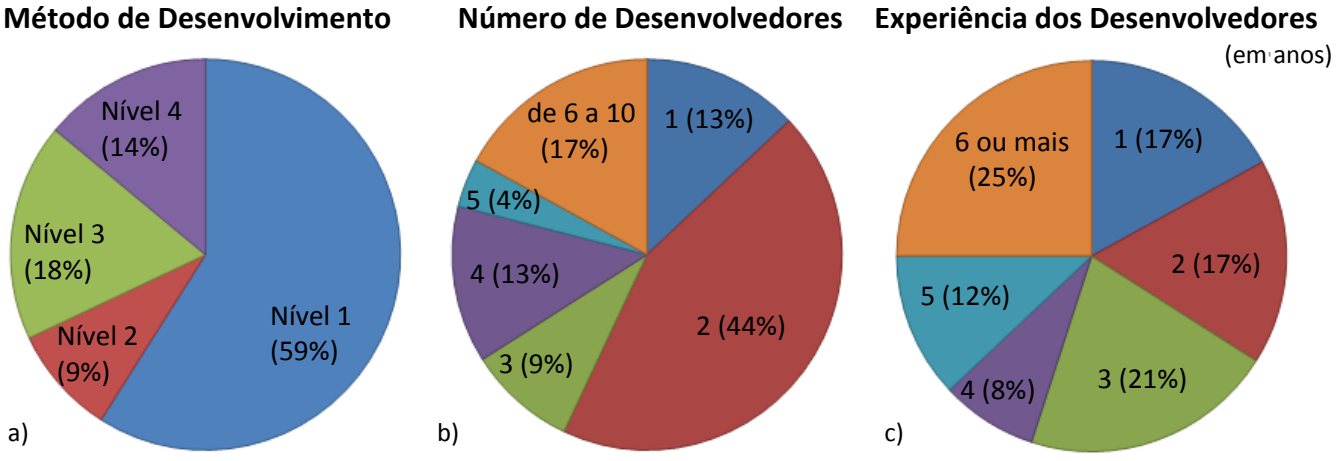

Figura 1: Características do contexto de desenvolvimento de grupos de pesquisa em Informática na Educação [9].

O contexto de desenvolvimento nos projetos relatados pelos programadores entrevistados é similar ao da maioria dos grupos de pesquisa estudados. Isso permite inferir que esses projetos podem também enfrentar problemas de desenvolvimento, de forma próxima aos relatados das entrevistas.

No grupo de pesquisa no qual este trabalho foi realizado (Laboratório de Informática na Educação - LInE) a situação de baixo uso de técnicas sistemáticas e problemas relacionados à qualidade de software também são frequentes. No LInE são desenvolvidos *Módulos de Aprendizagem Interativa* (iMA) [11] e ferramentas para integrar esses módulos a ambientes educacionais, como o Moodle [12]. Em geral, um iMA é um *applet* com ferramentas para autoria de atividades, que se comunica com ambientes Web e são específicos a alguma área do conhecimento. Os principais iMA disponibilizados em 2013 pelo LInE são: *iGeom*, um sistema de Geometria Interativa; *iGraf*, para estudo de gráficos e funções matemáticas; *iComb*, um sistema de ensino de combinatória; e *iVProg*, um sistema para ensino-aprendizagem de programação utilizando um modelo visual.

Deste modo, o objetivo deste trabalho é amenizar os problemas enfrentados durante o processo de desenvolvimento dos aplicativos educacionais da família iMA, no contexto do LInE. Para isso, utilizamos a *Linha de Produto de Software* (LPS) como principal ferramenta. A LPS é um conjunto de métodos, técnicas e ferramentas para o desenvolvimento sistemático de sistemas similares, que tenham um núcleo comum e compor-tamentos variantes [13]. Seguindo esse método proveniente da Engenharia de Software, é possível organizar o processo de desenvolvimento de iMA de forma a amenizar os problemas enfrentados.

Com isso, os resultados almejados dentro do grupo de pesquisa estão associados à redução do tempo de desenvolvimento, maior facilidade para a manutenção e evolução dos sistemas, aumento da satisfação dos programadores e uma maior qualidade do código desses aplicativos. Para outros grupos, o relato das experiências pode servir de exemplo para que iniciativas similares de aprimoramento do processo de desenvolvimento sejam implantadas.

A LPS 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 [13]. A partir de uma análise do domínio dos iMA existentes usando o método *Feature Oriented Domain Analysis (FODA)* [14], foi construída uma LPS, formada por um arcabouço de aplicação, que tem o papel do núcleo a ser reutilizado, um método de desenvolvimento para guiar o trabalho do programador e de documentação de código e processo [11].

Como contribuição para a área de Informática na Educação, este trabalho apresenta um exemplo de utilização de um método sistemático para o desen-volvimento de aplicativos educacionais, a LPS. Esse exemplo, além de sua influência direta no trabalho do grupo de pesquisa, pode contribuir com a sistematização do desenvolvimento de aplicativos realizado por outros grupos. Outras contribuições são os subprodutos do trabalho, artefatos que compõem a LPS, como o método de desenvolvimento para iMA, documentação e um arcabouço de aplicação.

Para avaliar a influência direta da LPS criada no desenvolvimento de iMA, essa foi utilizada para a criação de dois aplicativos: (i) o *iTangram*, que simula o brinquedo/jogo Tangram[1]; e (ii) a segunda versão do *iVProg*, cuja primeira versão foi criada por Brandão e Kamiya [15]. O processo de implementação desses aplicativos foi analisado como prova de conceito e estudo de caso, de forma a explicitar as contribuições da LPS para o trabalho dos programadores. Todos os sistemas desenvolvidos e documentação estão disponíveis como código livre[2].

O restante deste artigo está organizado da seguinte maneira: a próxima seção apresenta a fundamentação teórica, com trabalhos relacionados aos problemas descritos e soluções utilizadas por outros autores. A seção 3 descreve a LPS criada em quatro partes, a análise de domínio, a arquitetura, o arcabouço de aplicação e o método proposto para a engenharia de aplicação. Os resultados obtidos com relação à influência da LPS no desenvolvimento de iMA são discutidos na seção posterior. O artigo termina com algumas conclusões e sugestões de trabalhos futuros.

2. Fundamentação Teórica

Este é um trabalho de aplicação de conhecimentos da Engenharia de Software em um projeto de desenvolvimento de aplicativos educacionais. Dessa forma, a fundamentação aborda processos de criação desses aplicativos e outras iniciativas de utilização de técnicas da engenharia de software em projetos específicos para a educação. Por fim, apresenta alguns conceitos sobre LPS e os iMA.

1 O Tangram é um jogo milenar, criado na China, composto de sete (7) peças, sendo 4 triângulos e 3 paralelogramos.

2 http://ccsl.ime.usp.br/redmine/projects/ima

2.1 Processos de Desenvolvimento de Aplicativos Educacionais

O desenvolvimento de aplicativos educacionais pode ser dividido em duas grandes etapas: projeto instrucional e projeto e implementação de software. O projeto instrucional usualmente exige uma equipe multidis-ciplinar que envolve, em geral, professores, projetistas instrucionais, programadores, administradores de ambientes educacionais, responsáveis pelos currículos, pedagogos e coordenadores, dentre outros [10]. De acordo com os autores Winters e Mor, 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 [5].

Existem processos de desenvolvimento que consideram tanto a etapa instrucional quanto a de implementação, como o apresentado por Braga et al. [17]. Pela característica dos iMA possuírem ferramentas de autoria para os professores, o projeto instrucional ocorre predominantemente com o aplicativo pronto. Assim, este trabalho trata com mais profundidade a etapa de projeto e implementação de software.

Quando o aplicativo é desenvolvido em ambiente acadêmico, aparecem outros fatores que influenciam o projeto. Um deles é a relação do aplicativo com um projetos de pesquisa. Ao fim do projeto, é muito comum resultar em interrupção prematura do desenvolvimento dos aplicativos associados, deixando-se eventuais usuários sem suporte técnico. Outras dificuldades presentes no contexto acadêmico são relacionadas com os desenvolvedores, que são alunos. Em geral eles têm pouca experiência, como indicado em [9], e quando adquirem maior maturidade, finalizam seus estudos, deixando o projeto.

A qualidade de código também possui uma influência sobre as experiências de aprendizagem [10]. Qualidade de código pode ser entendida como a medida de características desejáveis em um código fonte, como lisibilidade e modularidade, que pode ser medida em métricas [18-19]. Por exemplo, um aplicativo com defeitos não detectados no desenvolvimento pode desmotivar alunos e influenciar negativamente a aprendizagem. Baixa qualidade de código pode provocar dificuldades durante sua manutenção e consequentemente lentidão na atualização do aplicativo e na correção de defeitos. Esses e outros problemas de qualidade de software podem reduzir o número de usuários do aplicativo e sua contribuição à educação [4].

2.2 Trabalhos Relacionados

Com o mesmo objetivo deste trabalho, aprimorar o desenvolvimento de aplicativos educacionais em situações específicas, vários pesquisadores utilizaram técnicas de engenharia de software [20]-[37]. Para fornecer uma visão geral desses trabalhos, os separamos de acordo com dois aspectos: o tipo de interação proporcionada pelo aplicativo criado e características da técnica de desenvolvimento utilizada. A distribuição dos artigos é apresentada na Tabela 1. Os critérios para a classificação dos artigos são apresentados a seguir.

A Tabela 1 apresenta uma síntese da revisão bibliográfica efetuada, destacando os trabalhos que tiveram maior influência nesta pesquisa. Nas linhas estão os trabalhos separados de acordo com o método de desenvolvimento, se promove apenas reúso de código ou não, e nas colunas pelo tipo de interação presente no sistema descrito no artigo, se usam mecanismos de interação genérica ou atividades com interatividade específica.

	Artigos que relatam a criação de aplicativos com mecanismos de interação genéricos (ex: botões, formulários e listas)	Artigos que apresentam aplicativos com outros mecanismos de interação (ex: animações interativas, manipulação de objetos)
Artigos que relatam uso de técnicas com apenas reúso de código	[20] Douglas(2001) [21] Boyle (2003) [22] Polsani (2003) [23] Pankratius (2005) [24] Ateyeh e Lockerman (2006) [25] Choquet e Corbière (2006) [26] Doderoet al. (2007)	[27] Nicholson e Scott (1986) [28] Roschelle et al. (1998) [29] Conlan et al (2002) [30] Bote-Lorenzo et al. (2004) [31] van Damet al. (2005)
Artigos com técnicas de reúso de código, arquitetura e processo	[32] Oberweiset al. (2007) [33] Reis (2007) [34] Gadelha et al. (2010) [35] Oliveira e Gerosa (2011)	[36] Stuiks e Damasevicius (2008) [37] Ahmed e Zualkernan (2011)

Tabela 1: Classificação de trabalhos que aplicam técnicas da Engenharia de Software em projetos de desenvolvimento de aplicativos educacionais.

Sobre os tipos de interação, os sistemas apresentados nos artigos possuíam mecanismos de interação genéricos,

como botões e caixas de opções (*combobox*), ou possuíam atividades interativas com outros mecanismos de interação, como animações e manipulação de objetos. Essa classificação segue os tipos de interatividade descritos por Tang [38]. A escolha dessa característica para agrupar os trabalhos relacionados é motivada pela natureza dos iMA. Os iMA fornecem atividades com mecanismos de interação específicos para cada área de conhecimento, o que não acontece nos casos do outro grupo de aplicativos. Isso influencia como o sistema é projetado e implementado [39].

Os métodos de desenvolvimentos empregados foram classificados quanto a existência ou não de mecanismos explícitos de reúso de arquitetura e processo, além do reúso de código. Na Tabela 1, nota-se clara predominância de métodos com apenas reúso de código. Cronologicamente, os artigos apresentavam mais técnicas como bibliotecas e componentes, o que posteriormente deu lugar a mais reúso de arquitetura e processo, como arcabouços de aplicação e LPS.

Uma análise da Tabela 1 indica a existência de tendências a partir da correlação entre os anos de publicação dos trabalhos e sua classificação quanto à técnica utilizada e os mecanismos de interação. Antes da massificação do uso de Sistemas de Gerenciamento de Curso (SGC), como o Moodle e o Sakai a partir do início dos anos 2000, encontra-se mais esforços no desen-volvimento de atividades interativas usando técnicas de reúso de código, como bibliotecas e arquiteturas de componentes. A partir do uso generalizado dos SGC, os esforços se voltaram mais para os 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) [40].

2.3 Linhas de Produto de Software (LPS)

A LPS visa a racionalização do processo de desenvolvimento e manutenção de software buscando agrupar sistemas com características comuns em uma única linha de produção de software. Como descreve Krueger em trabalho publicado na Web:

> Uma característica que distingue Linhas de Produto de Software de esforços anteriores quanto ao reúso é o *preditivo* versus o *oportunístico*. Em vez de colocar os componentes de software genéricos em uma biblioteca na esperança de surgir *oportunidades* de reutilização, as linhas de produtos de software apenas demandam pela criação de artefatos de software quando sua reutilização está *prevista* em um ou mais produtos em uma linha de produtos bem definidos. [sexto parágrafo de Krueger, 2013][3]

A LPS não diz respeito apenas ao reúso de código, sendo mais geral, visa também o reúso de processo e arquitetura. Reúso de código ocorre quando se pode aproveitar partes de código escritas anteriormente, como em bibliotecas da linguagem C. Reúso de arquitetura ocorre quando uma estrutura complexa é reutilizada para resolver um problema, como em arcabouços para a Web. Reúso de processo acontece quando atividades, etapas e papéis de pessoas e outros recursos são reutilizadas.

É possível criar uma LPS com diversas ferramentas, como arcabouços e bibliotecas de componentes. Essa técnica prescreve que o desenvolvimento seja dividido em duas partes, a *engenharia de domínio* e a *engenharia de aplicação*. Na primeira é criado o núcleo comum aos produtos finais da família, enquanto que na segunda o núcleo é reutilizado e expandido para a criação desses produtos.

Neste trabalho, a LPS foi escolhida pelo fato dos iMA serem um conjunto de aplicativos similares, que compartilham uma estrutura interna e um conjunto de funcionalidades para o usuário final. Esse é o campo de aplicação que motivou a criação dessa técnica e o mais indicado para sua utilização [13].

Os métodos de desenvolvimento da LPS aqui utilizados foram o FODA (*Feature Oriented Domain Analysis*) [14] e o PLUS (*Product Line UML-based Software Engineering*) [41]. O primeiro foi usado para a análise de domínio, cujo produto é um diagrama de características. O segundo serviu de base para o projeto de software. Nesse método, a linha de produto é criada com diagramas UML, seguindo as etapas de análise de domínio (requisitos e casos de uso), análises estática e dinâmica, e implementação.

Esses métodos foram utilizados para a criação de um arcabouço de aplicação. Eles foram escolhidos pela familiaridade que os autores possuíam com projeto de software orientado a objeto com UML. O arcabouço possui papel de núcleo da LPS e fornece os recursos a serem reutilizados durante o desenvolvimento dos iMA.

3 Frase original: "The characteristic that distinguishes software product lines from previous efforts is predictive versus opportunistic software reuse. Rather than put general software components into a library in hopes that opportunities for reuse will arise, software product lines only call for software artifacts to be created when reuse is predicted in one or more products in a well defined product line." em http://www.methodsandtools.com/archive/archive.php?id=45.

2.4 Módulos de Aprendizagem Interativa

A família iMA surgiu quando foram criados aplicativos derivados do *iGeom* para outras áreas, aproveitando conceitualmente algumas de suas funcio-nalidades, como poder ser usado em navegadores Web e possuir um protocolo de comunicação com o ambiente Web em que está integrado. Assim surgiram o *iGraf*, o *iComb* e o *iVProg*. Todos foram desenvolvidos para trabalhar tanto como aplicativo quanto como *applet*.

Outra característica comum dos iMA é a especificidade de seus mecanismos de interação com o usuário, que dependem da área de conhecimento que abordam: geometria, gráficos de funções, combinatória e programação visual. Esses conceitos podem ser expandidos para outras áreas com a criação de novos iMA. Deste modo, a existência de uma base comum a todos os iMA pode facilitar de modo significativo a manutenção e expansão da família.

Os iMA vêm sendo usados em cursos de extensão e por professores de ensino superior e fundamental para aprimorar o ensino de matemática e programação desde 2000. As principais vantagens relatadas pelos usuários são os mecanismos de interação com o aluno e a facilidade de utilizar os aplicativos integrados a ambientes Web [42]. A dificuldade enfrentada em desenvolver novas funcionalidades nos aplicativos existentes motivou este trabalho para sistematização do desenvolvimento com uma LPS para iMA.

3. Uma LPS para iMA

A LPS para iMA foi desenvolvida seguindo quatro (4) etapas: (i) análise de domínio; (ii) definição da arquitetura geral; (iii) desenvolvimento de um arcabouço de aplicação, que forma a Engenharia de Domínio; e (iv) definição do método de desenvolvimento, que corresponde à Engenharia de Aplicação.

A análise de domínio produziu o escopo da LPS, servindo de base para a arquitetura fornecer uma descrição do funcionamento interno de um iMA. O arcabouço de aplicação implementa essa arquitetura e fornece o código a ser reutilizado nos iMA, enquanto o método de desenvolvimento gera a descrição de como a LPS deverá ser usada para criar novos iMA.

3.1 Análise de Domínio

Esta primeira etapa serviu para capturar as funcionalidades comuns à família de iMA e definir seu escopo. Assim, a análise de domínio foi realizada estudando-se os iMA existentes, o que resultou basicamente em três produtos: (i) uma lista de requisitos gerais para os iMA; (ii) a lista detalhada das funcionalidades disponíveis para o usuário final classificadas de acordo com a presença nos aplicativos existentes e sua possibilidade de reúso; e (iii) a modelagem das características da LPS, baseadas nesses requisitos e funcionalidades, representada por um diagrama de características.

A análise constatou que os requisitos gerais mais importantes que identificam a família dos iMA são:

(i) funcionar em navegadores Web;

(ii) comunicar-se com o SGC pelo protocolo HTTP;

(iii) fornecer ferramenta de *autoria de atividades* para professores;

(iv) apresentar algum mecanismo de avaliação automática para as soluções dos alunos.

As funcionalidades presentes em cada iMA foram classificadas com relação à sua presença nos aplicativos. Uma *funcionalidade comum* é aquela que está ou poderia estar presente em mais de um membro da família de aplicativos, de acordo com sua compatibilidade com outros domínios. 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 2. As características existentes podem ser agrupadas em três tipos:

(i) *características obrigatórias*, aquelas presentes em todos os aplicativos, identificadas com na figura com arestas simples (sem detalhes de início ou fim). Essas características formam o núcleo comum da LPS;

(ii) *características alternativas*, são aquelas que cada aplicativo deve possuir apenas uma das possibilidades, na figura estão identificadas com um arco cortando as linhas sobre as opções;

(iii) *características opcionais*, são aquelas que um aplicativo pode ou não possuir, estão identificadas na figura por um círculo ao fim do traço.

No diagrama de características, um iMA específico pode ser descrito por meio de todas suas características obrigatórias (todas as de *Comunicação*, de *Configuração* e de *Gerenciamento de Atividades*), apenas uma das *Operações de Domínio*, que representará as funcio-nalidades particulares do seu domínio (geometria, combinatória, etc.) e por suas características opcionais, dentro das categorias *Funcionalidades sobre Atividades* e *Funcionalidades sobre Operações de Domínio*. As reticências no diagrama indicam o local em que novas características podem ser incluídas.

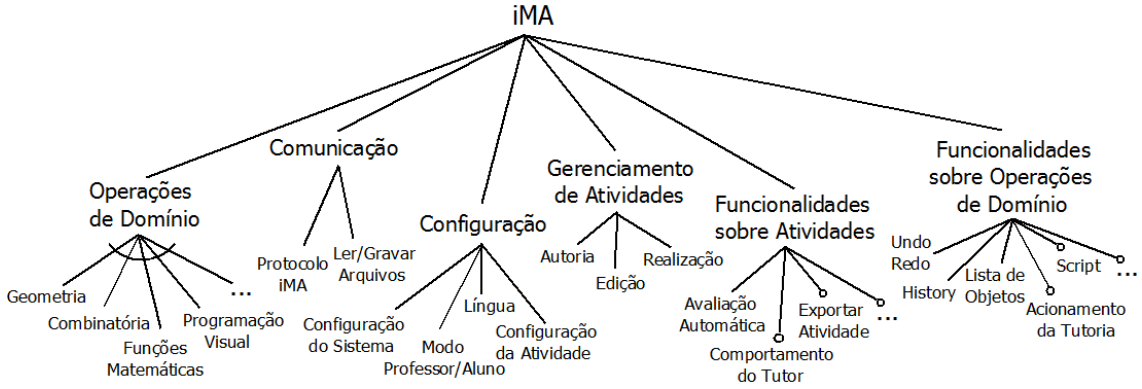

Figura 2: Diagrama de características da LPS para iMA

3.2 Arquitetura da LPS

A partir da identificação das características, uma arquitetura de sistema foi criada 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 duas categorias: *Funcio-nalidades sobre Atividades* e *Funcionalidades sobre Operações de Domínio* [43].

Essa tarefa intermediária entre o mapeamento das características e a modelagem das entidades que devem ser desenvolvidas teve objetivo de permitir que o núcleo da LPS fosse independente de domínio. Ou seja, para que o núcleo não estivesse preso às características de algum iMA específico.

O resultado foi a produção de uma arquitetura genérica, fortemente baseada em Padrões de Projeto de Software [44], tais como os padrões *Comando* e *Observador*. A arquitetura possui abstrações dos objetos de domínio e mecanismos de interação usados para manipulá-los, além disso a arquitetura descreve uma forma dos objetos e mecanismos serem compatíveis com as funcionalidades comuns a todos iMA. Essa arquitetura é composta por quatro componentes, conforme mostra a Figura 3, e descritos nas subseções a seguir:

(i) *componente estrutural*, que oferece a base e as funcionalidades comuns;

(ii) *componente de atividades*, que define as atividades para os alunos, a autoria para professores e a funcionalidade de avaliação automática;

(iii) *componente de domínio*, que modela de forma genérica os objetos e as operações de domínio;

(iv) *componente de extensão*, que modela genericamente as funcionalidades sobre atividades e as funcionalidades sobre operações de domínio.

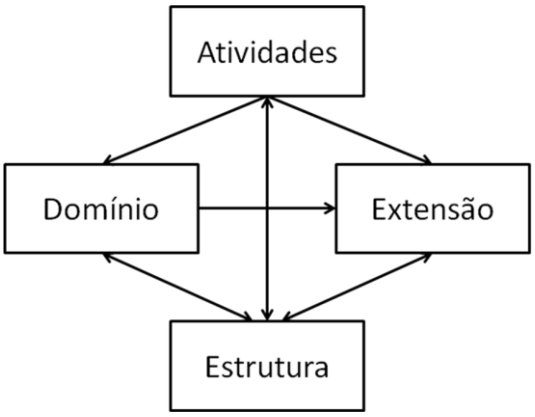

Figura 3: Arquitetura genérica da LPS para iMA.

3.2.1 Componente Estrutural

O *componente estrutural* possui três papéis principais: (i) fornecer a estrutura básica do sistema, (ii) fornecer funcionalidades independentes de domínio, como gerenciamento de arquivos, comunicação com SGC e opções de configuração; e (iii) iniciar e definir a comunicação entre os demais componentes. A estrutura básica define, entre outras características, com quais plataformas o conjunto de sistemas da linha será compatível. No caso dos iMA, define a estrutura básica de *applet*, porque os atuais aplicativos da família utilizam essa tecnologia. As funcionalidades comuns devem estar presente em qualquer dos possíveis domínios do sistema (dentre aqueles da família). Por definir a estrutura básica, esse componente é responsável pela carga do sistema e de cada um dos outros componentes.

Na Figura 3, as setas entre o componente estrutural e os demais indicam que o primeiro inicializa os demais componentes e que esses últimos fazem uso das funcionalidades independentes de domínio fornecidas pelo componente estrutural. No caso dos iMA, elas são as funcionalidades obrigatórias do diagrama de características da Figura 2.

3.2.2 Componente de Atividade

O *componente de atividade* possui também três funções: (i) definir e modelar as atividades; (ii) permitir que um professor crie e edite atividades; e (iii) permitir que alunos realizem as atividades. Todas essas funções devem ser fornecidas de forma independente de qualquer domínio. O modelo de atividades proposto é apresentado na Figura 4, contendo cinco (5) elementos principais. Essa estrutura foi derivada do uso cotidiano dos aplicativos iMA existentes.

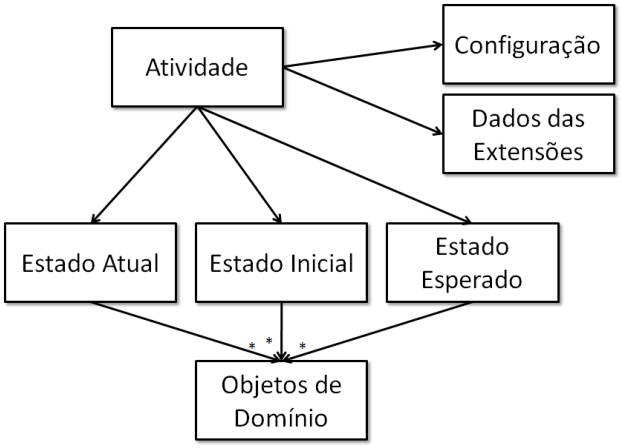

Figura 4: Representação da estrutura interna de uma Atividade.

(i) *Estado inicial*: é o conjunto de objetos de domínio que é recebido pelo aluno ao iniciar a realização de uma atividade. Os objetos de domínio são definidos no componente de domínio, apresentado no próximo item;

(ii) *Estado atual*: é o conjunto de objetos de domínio que está sendo manipulado durante a realização da atividade, geralmente por um aluno. Esse estado é inicializado com o valor do Estado Inicial quando o aluno começa a realizar a atividade.

(iii) *Estado esperado*: é o conjunto de objetos de domínio que o professor definiu como resposta, fazendo o papel de um *gabarito*. Ao final de uma atividade por parte do aluno, o estado esperado será comparado com o estado atual para possibilitar uma avaliação automática sobre a solução do aluno. Este recurso possibilita a implementação de *Tutoria Inteligente* em um iMA [41];

(iv) *Configuração*: é o conjunto de informações que definem comportamentos do sistema relacionados às atividades. Um dado de configuração comum é o texto enunciado da atividade. Outro exemplo, pode-se configurar determinada atividade para que o aluno tenha a sua disposição um subconjunto reduzido das funcionalidades disponíveis no iMA. Mais especificamente, no caso da Geometria, pode-se desabilitar um construtor de ponto médio sempre que o exercício tiver como objetivo que o aluno construa o ponto médio entre dois pontos dados;

(v) *Dados das extensões*: é o conjunto de informações específicas das extensões utilizadas pela atividade, como histórico de ações e dados para implementação de *Tutoria Inteligente*, quando essas extensões estiverem disponíveis.

A sequência de ações para o aluno é a seguinte: (i) o aluno abre uma atividade no iMA que exibe o Estado Atual com o valor definido pelo Estado Inicial, personalizado de acordo com a Configuração e os Dados das Extensões; (ii) o aluno manipula os objetos de domínio do Estado Atual usando operações de domínio até acreditar chegar na resposta correta; (iii) o aluno pede para o iMA comparar seu Estado Atual com o Estado Esperado para receber a informação se sua solução está correta ou não.

No caso do professor, a sequência de ações básica é a seguinte: (i) o professor abre o iMA no modo autoria; (ii) o professor cria e manipula os objetos de domínio com ações de domínio; (iii) define o Estado Inicial; (iv) define o Estado Final; (v) define a Configuração; e (vi) eventualmente define os Dados das Extensões. Essas sequências de ações de uma atividade são abstratas e devem ser concretizadas durante a criação de um iMA.

No diagrama da Figura 3, a seta que sai do componente de atividade em direção ao componente de domínio representa o acesso às ações de domínio e objetos de domínio, descritos abaixo. A seta que vai em direção ao componente de extensão indica o uso de alguma informação presente em alguma das extensões.

3.2.3 Componente de Domínio

Sob o ponto de vista dos desenvolvedores, o *componente de domínio* define conceitos abstratos que devem ser instanciados durante a criação do sistema, listados a seguir. Quando um programador cria um iMA, deve definir especializações desses conceitos para o domínio específico do iMA, por exemplo, geometria. Esses conceitos são utilizados pelos outros componentes também de forma abstrata, garantindo assim a independência de domínio de suas funcionalidades. Esse componente é composto de quatro conceitos:

(i) *Objeto de domínio*: é o item básico de um domínio, que é criado ou manipulado sempre que um usuário realiza uma operação, por intermédio de uma ação de domínio. Exemplos de objetos de domínio em Matemática incluem números, pontos, vetores, conjuntos e circunferências;

(ii) *Ação de domínio*: é uma operação sobre quaisquer objetos de domínio, que é disparada a partir de alguma intervenção realizada pelo usuário do sistema. Um

exemplo de ação de domínio em Matemática é a construção de uma circunferência a partir de dois pontos dados;

(iii) *Modelo de domínio*: é o conjunto de regras de domínio que deve ser utilizado por qualquer ação de domínio. É responsável pela execução das ações de maneira a respeitar a consistência do domínio;

(iv) Interface com o usuário de domínio: é a representação gráfica dos objetos e das ações de domínio por meio das quais o usuário manipula os objetos.

Esses quatro conceitos devem ser implementados durante o desenvolvimento de um sistema dentro da família de aplicativos. A partir dessa definição é derivado o processo de desenvolvimento dentro da linha de produtos, guiando o programador na instanciação de cada elemento. Na Figura 3, a seta que aponta do componente de extensão para o componente de domínio representa a utilização dos conceitos de domínio por parte de alguma das extensões.

3.2.4 Componente de Extensão

Por fim, o *componente de extensão* é responsável por definir, também de forma abstrata e independente de domínio, o conjunto de funcionalidades que podem ser realizadas sobre objetos ou ações de domínio por parte do sistema. Esse componente define a arquitetura das (i) funcionalidades sobre atividades e (ii) as funcionalidades sobre operações de domínio.

A arquitetura para as funcionalidades sobre atividades é definida na forma de uma extensão (*plug-in*). Essa extensão deve possuir uma interface gráfica na forma de um botão, sendo ela a responsável por realizar a funcionalidade. Para isso ela deve receber como parâmetro todos os dados de uma atividade. Para as funcionalidades sobre operações de domínio, a arquitetura é a de uma extensão que pode possuir uma interface gráfica na forma de um botão e que se comporta como um observador de ações de domínio. Sempre que uma ação de domínio for executada, essa extensão pode desencadear uma funcionalidade recebendo como parâmetro todos os dados dessa ação.

A arquitetura genérica descrita aqui define uma série de conceitos essenciais para a linha de produto para iMA. Ela serve de base para a implementação do arcabouço de aplicação que, por sua vez, serve de núcleo da LPS, descrito a seguir. Além disso, essa arquitetura pode ser usada fora da LPS para iMA, na modelagem de outras famílias de sistemas que tenham requisitos similares. Um exemplo é uma família de aplicativos educacionais para autoria de imagens de diferentes formatos (que caracteriza um domínio diferente), usando outras tecnologias que não seja *applet* (tecnologia diferente).

3.3 Arcabouço de Aplicação

Um arcabouço de aplicação foi escolhido como forma de implementação do núcleo da LPS por atender características desejadas ao contexto de desenvolvimento de iMA. Essas características incluem (i) não exigir conhecimentos dos programadores além das técnicas de orientação a objetos; (ii) permitir a criação de um manual passo-a-passo para sua utilização, representando o reúso de processo; e (iii) promover naturalmente reúso de código e arquitetura [45]. A exigência de conhecimento dos programadores apenas em orientação a objetos e a criação de manuais detalhados são importantes para facilitar o trabalho dos desenvolvedores que, no caso do grupo de pesquisa em questão, algumas vezes são professores de matemática com pouca experiência em desenvolvimento.

O arcabouço para desenvolvimento de iMA foi derivado diretamente da arquitetura de sistema descrita na seção anterior. O diagrama de componentes da versão atual do arcabouço é mostrado na Figura 5. Os componentes com nome em **negrito** são parte do núcleo comum e fornecem as funcionalidades obrigatórias, aqueles em *itálico* têm o papel de implementar as operações e interface gráfica de domínio, específicas para cada iMA, e os com nome sublinhado representam as funcionalidades opcionais.

Os componentes *InterfaceDeDomínio* e *ModeloDeDomínio* constituem a interface gráfica e o modelo das funcionalidades específicas de domínio, respectivamente. O componente **InterfaceBase** é a interface gráfica básica e comum. Os componentes **ControleDeSistema**, **ControleDeAtividade** e **Comunicação** fornecem, respectivamente, a estrutura, o gerenciamento de atividades e o protocolo de comunicação com Sistemas Gerenciadores de Cursos (SGC). Por fim, o componente MóduloDeAtividade modela as funcionalidades sobre atividades e o MóduloDeOperações aquelas sobre operações de domínio, ambos usando o conceito de complemento (*plug-in*).

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.

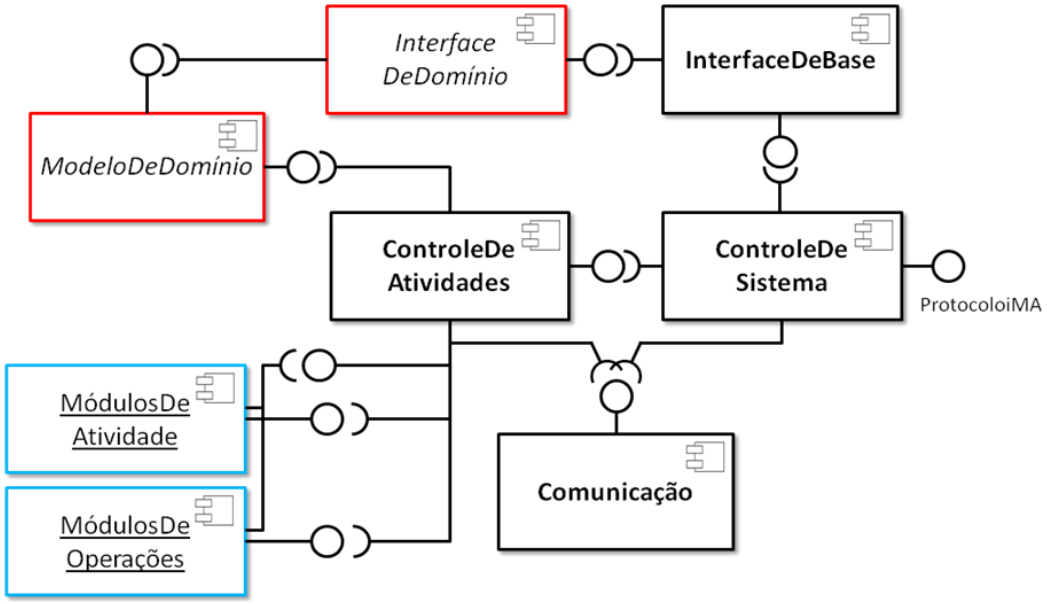

Figura 5: Diagrama de componentes do arcabouço para desenvolvimento de iMA.

3.4 Método para Engenharia de Aplicação

O método proposto pela LPS para a Engenharia de Aplicação, ou seja, para o desenvolvimento de iMA, 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 foi 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 *ModeloDeDominio* e *InterfaceDeDomínio*. O *ModeloDeDominio* descreve os objetos de domínio e as ações que o usuário pode realizar para manipulá-los. O componente *InterfaceDeDomínio* 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.Esses passos correspondem à instanciação do componente de domínio da arquitetura, definido na seção 3.2.3. Mais especificamente, *ModeloDeDominio* exige a implemen-tação dos objetos de domínio, das ações de domínio e do modelo de domínio. Complementando-o, o *InterfaceDe-Domínio* exige a implementação da interface de atividades, na qual os alunos realizam as atividades, e da interface de autoria, na qual os professores criam as atividades.

A LPS fornece adicionalmente um método para a criação de funcionalidades opcionais para os iMA. Esse método descreve como um programador deve implementar os componentes MóduloDeAtividade e MóduloDeOperações. Essa é a principal forma de expandir o núcleo da LPS, adicionando novas funcionalidades opcionais ou comuns aos iMA. Esse método corresponde à implementação de funcionalidades cuja arquitetura é definida no componente de extensões , descrito na seção 3.2.4. No arcabouço, o Módulo-DeAtividade implementa a arquitetura para as funcionalidades sobre atividades e o MóduloDe-Operações implementa as funcionalidades sobre operações de domínio. Atualmente, o arcabouço fornece algumas extensões, como histórico de ações e exportação de atividades. A Figura 6 apresenta um exemplo de interface com o usuário com acesso a módulos de atividade e de operações em uma variante atual do *iGeom* chamada *iGeom Kids*.

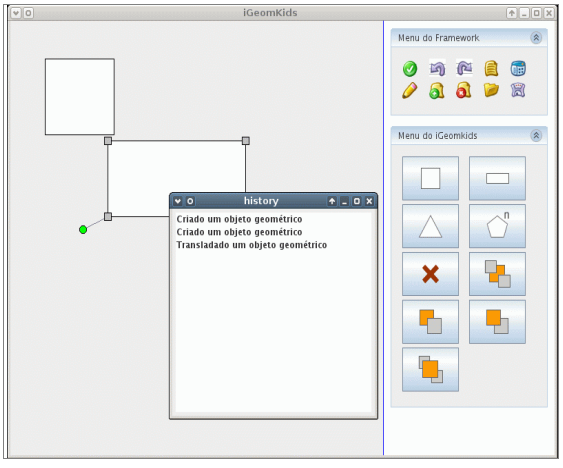

Figura 6: Tela do *iGeom Kids* com um exemplo de histórico de ações.

4. Avaliação da Influência da LPS no desenvolvimento de iMA

Para avaliar o impacto da LPS criada no desenvolvimento de iMA, foram realizadas avaliações da implementação de dois novos aplicativos, o *iTangram* e a segunda versão do *iVProg*. Os estudos de avaliação seguiram os métodos descritos como prova de conceito [46] no caso do *iTangram* e estudo de caso [47-48] no caso do *iVProg*. Essa diferença nos métodos deve-se ao contexto de cada desenvolvimento, *iTangram* foi desenvolvido como projeto semestral de uma disciplina de pós-graduação, enquanto que *iVProg* faz parte de um projeto de mestrado.

Com uma *prova de conceito* é possível mostrar que um conceito é factível. Por exemplo, no caso deste trabalho, a prova de conceito é um iMA criado com a LPS. Ela ilustra a possibilidade de criação de um iMA a partir do uso da LPS. Nas categorias de tipos de pesquisas descritas por Wazlawick (2009) [46], a *prova de conceito* seria o resultado de uma pesquisa de apresentação de produto. Por não possuir rigor, a prova de conceito não permite conclusões além da possibilidade de realizar uma proposta.

Já um *estudo de caso* é um método de análise profunda de um fenômeno [47-48]. Tal estudo deve ter as etapas de planejamento, de coleta e de análise de dados. A coleta de dados deve ser realizada de várias formas, para que a análise não seja enviesada por características de um único método. Podem ser usados métodos de coleta quantitativos ou qualitativos. Assim, como é entendido 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 O novo iMA: *iTangram*

A prova de conceito realizada neste trabalho teve o objetivo de verificar a possibilidade de uso da LPS para o desenvolvimento de um novo iMA. Dessa forma, uma primeira versão completa da LPS foi entregue aos desenvolvedores do *iTangram*, juntamente a um conjunto requisitos que descreviam o aplicativo a ser criado. Este sistema educacional alvo é baseado no tradicional jogo geométrico Tangram.

Durante o desenvolvimento do *iTangram*, o responsável pela LPS (primeiro autor deste artigo) intervia apenas para resolver dúvidas, corrigir decisões realizadas que eram incompatíveis com a LPS e eventualmente corrigir problemas encontrados no arcabouço. Essas intervenções eram feitas em número reduzido e com cuidado para manter a independência dos programadores. Sem as intervenções os programadores levariam mais tempo para entender os manuais e enxergar erros de utilização da LPS. O objetivo era aprimorar a LPS e possibilitar que no curto período de tempo destinado ao projeto (apenas 2 meses), fosse possível construir o *iTangram* utilizando apenas a LPS como suporte.

Ao longo de dois meses, três alunos de pós-graduação trabalharam no desenvolvimento do *iTangram* durante as aulas de uma disciplina de pós-graduação (ministrada pelo segundo autor deste artigo). A Figura 7 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 (área de trabalho) com as peças do Tangram. A barra de botões faz parte da interface gráfica básica, fornecida pelo arcabouço. As funcionalidades acessíveis por ela são independentes do domínio, como a autoria de atividade ou abrir ou gravar arquivos.

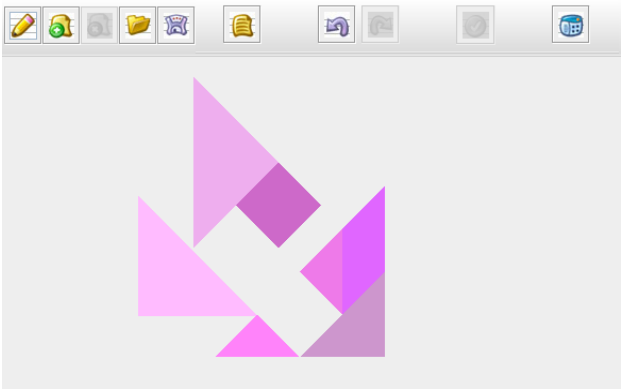

Figura 7: Interface gráfica do *iTangram* com peças movimentadas.

As funcionalidades específicas de domínio são acessíveis a partir da área de trabalho, como por exemplo, a translação e a rotação das peças com o uso do *mouse*. Em uma atividade típica deste iMA, é fornecido ao aluno um desenho composto pelas sete (7) peças do Tangram, sem a identificação de cada uma. O aluno deve, a partir das peças em uma posição inicial, formar a figura indicada. Por falta de tempo, o avaliador automático, que compararia a configuração proposta pelo aluno com uma configuração modelo não pode ser implementado pelos alunos durante a disciplina. A construção do *iTangram* foi realizada em aproximadamente dois (2) meses, em sessões semanais de no máximo quatro (4) horas de trabalho.

Como ao final do processo foi possível produzir um iMA utilizando a LPS criada, a prova de conceito validou a proposta. Ou seja, o *iTangran* é um exemplo de iMA cujo desenvolvimento usou como base a LPS desenvolvida neste trabalho.

4.2 Segunda versão do iMA *iVProg*

O *iVProg* é um aplicativo para ensino de Introdução a Programação empregando um modelo visual, no qual o aluno pode escolher comandos a partir de ícones [15]. O

contexto do estudo de caso é o seguinte. Após seu principal desenvolvedor encerrar o programa de mestrado e consequentemente sua participação no projeto, um outro estudante-programador assumiu a tarefa de continuar o *iVProg*. O novo programador trabalhou com o código legado por seis (6) meses. Após esse período, passou a desenvolver uma versão completamente nova baseada na LPS (i.e., nenhum código foi aproveitado, exceto aquele do arcabouço)

4.2.1 Planejamento e execução do estudo de caso

Nesse estudo de caso, a avaliação da influência da LPS consistiu em comparar o trabalho do programador nos períodos anterior e posterior à adoção da LPS. Os dados coletados em cada um dos períodos consistiram principalmente de código fonte do *iVProg* e relatos de entrevistas com o programador e o coordenador do projeto sobre o andamento das tarefas.

O estudo de caso realizado teve como objetivo investigar a influência da LPS no desenvolvimento do *iVProg*. Dessa forma, se caracteriza como uma pesquisa com caráter exploratório, de levantamento de fatores importantes no processo, visando a qualidade do produto final. Caso haja evidências de maior influência positiva do que negativa, é possível afirmar que a LPS aprimora o desenvolvimento de iMA [48].

O método de trabalho desse estudo foi planejado para comparar um mesmo fenômeno (o desenvolvimento do *iVProg*) sob duas circunstâncias distintas, uma antes da adoção da LPS e outra depois. Para caracterizar esses dois momentos, foram coletados dois tipos de dados: (i) relatos do desenvolvedor e do coordenador do projeto; e (ii) dados quantitativos de métricas do código fonte produzido. Os relatos gravados em entrevistas servem para levantar indícios sobre o método utilizado, da percepção de produtividade e de qualidade do produto, além de considerar a opinião dos envolvidos no projeto com relação aos resultados obtidos em cada momento. As métricas de código fonte servem para levantar indícios sobre a qualidade de código do sistema em desenvol-vimento [18-19]. Todos os dados coletados estão disponíveis no site do projeto[4].

Em um estudo de caso, idealmente todas as condições devem ser mantidas nas situações anterior e posterior à intervenção, exceto aquela que se deseja estudar. Na prática, muitas vezes é impossível mantê-las invariantes. Isso ocorreu em nosso estudo, por isso a análise dos dados precisou considerar alguns aspectos que diferenciavam o momento anterior e o posterior, além da variação na influência da LPS. Ao longo do período analisado, por ser o desenvolvedor um aluno iniciante na área da programação[5], ele continuou estudando e amadurecendo como programador. Uma distinção clara entre os dois momentos é que o código produzido no segundo momento é todo de autoria do desenvolvedor, enquanto que anteriormente havia código legado. Para contornar essas mudanças não desejadas nas condições do estudo de caso, as entrevistas conduzidas continham perguntas específicas, o que será comentado a seguir na seção de análise dos dados.

4.2.2 Contexto e histórico do *iVProg*

Logo que assumiu o desenvolvimento do *iVProg*, 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 baixa qualidade do código legado, aliado à falta de documentação, em cerca de seis (6) meses de trabalho apenas uma correção foi realizada com sucesso e apenas parte do código foi bem compreendido por ele.

Ao passar a utilizar a LPS, em vez de continuar trabalhando com a versão inicial do *iVProg*, o progra-mador começou a desenvolver uma nova versão do aplicativo. 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 Na Figura 8 são apresentandas suas interfaces em três versões distintas geradas durante esse processo de desenvol-vimento.

A interface gráfica inclui a barra de botões fornecida pelo arcabouço, já apresentada no *iTangram*, e dois painéis principais com objetos de domínio. Na parte esquerda das figuras, nas três versões, estão objetos disponíveis para a construção de programas e à direita está a área de trabalho, na qual o aluno constrói seu algoritmo. Uma atividade básica no *iVProg* consiste no aluno receber uma descrição de um problema que deve ser resolvido na forma de algoritmo, como determinar se um dado de entrada é ou não um número primo. Utilizando os comandos com interfaces visuais ele deve construir seu algoritmo. Na versão corrente do *iVProg*, de fevereiro de 2013, ainda não está implementado o avaliador automático, mas isso está no planejamento das próximas funcionalidades.

Os dados coletados serão apresentados em três partes: (i) relato do desenvolvedor e do coordenador para o período anterior à adoção da LPS; (ii) relatos para o período durante a utilização da LPS como método; e (iii) métricas de código fonte.

4 http://ccsl.ime.usp.br/redmine/projects/ima

5 O aluno fez graduação em Licenciatura em Matemática e, no início do projeto, estava em seu segundo semestre no mestrado.

4.2.3 Desenvolvimento anterior à adoção da LPS

Antes de receber o projeto, o programador possuía apenas experiência com programação em C, em disciplinas eletivas do Instituto em que fez licenciatura em matemática. Estudou a linguagem do *iVProg*, Java, com a ajuda de livros e páginas Web. Apenas pouco antes da adoção da LPS, cursou uma disciplina sobre programação orientada a objetos, na qual teve contato pela primeira vez com métodos sistemáticos e boas práticas de desenvolvimento.

Com relação ao método utilizado, em reuniões o coordenador informava as tarefas para o programador e indicava regiões do código que estariam relacionadas à solução. Assim, o programador lia o código para entender como o problema ocorre e solucioná-lo.

Após cerca de 6 meses, as soluções para os problemas selecionados não tinham sido implementadas. Apenas um defeito relacionado a diferentes versões de Java foi corrigido. O extrato da entrevista abaixo ilustra a opinião do programador com relação aos resultados obtidos.

Danilo - Então o que te impedia de produzir mais, de programar mais?

Programador - Acho que experiência. Se eu tivesse mais experiência, sei lá [sic], ou então se o código do iVProg fosse mais claro, estivesse documentado eu acho que iria perder muito menos tempo. Porque, assim, 90% do tempo era perdido para compreender, para entender, para buscar relação [sic] dentro do próprio código. Isso tomava muito tempo.

4.2.4 Desenvolvimento posterior à adoção da LPS

O desenvolvimento do novo *iVProg* foi feito em três (3) etapas: inicialmente foi desenhado e implementado o núcleo do *iVProg* (referente ao componente modelo de domínio); depois foi implementada uma interface com o usuário para este poder interagir com o núcleo *iVProg* (componente interface de domínio); e finalmente foi feita a integração com o arcabouço. O trecho da entrevista abaixo representa como o programador percebeu a influência do arcabouço em seu trabalho:

Danilo - E no final das contas, qual a relação do arcabouço com o método que você usou para fazer o novo iVProg?

Programador - Acho que foi mais na separação em objetos bem... assim por classes [sic]. Por exemplo, eu poderia reunir várias coisas em uma única classe mas eu acho que fica mais claro dado que existem lá [sic] os objetos de domínio, eu acho que ajudou colocar esses papéis [...]. Ele ajudou também a achar alguns pontos que estavam bem fracos no meu projeto. Quando eu comecei a integração eu enxerguei outras coisas, como por exemplo eu precisava de identificação única para cada objeto que eu ia [sic] adicionar no painel de domínio.

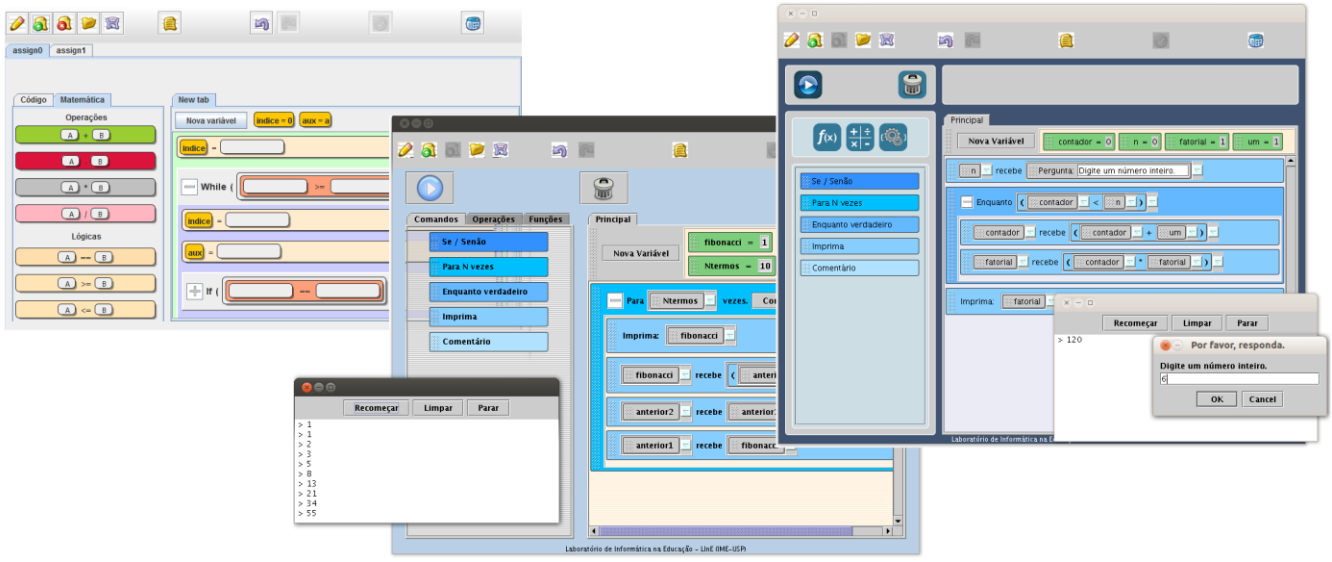

Figura 6 : Interface gráfica em diferentes fases de desenvolvimento da nova versão do *iVProg*, criada com o uso da LPS.

Para o programador, a LPS guiou seu trabalho, explicitando boas práticas e impedindo que a qualidade do código fosse reduzida ao longo do período. Também relatou que os limites definidos não afetaram sua produtividade. Como diz no trecho abaixo:

Programador - Acho que se não houvesse o arcabouço, o código ficaria mais parecido com o antigo [iVprog], eu acho que por conta da direção que o arcabouço dá, é bem diferente [sic].

Nas entrevistas, o programador e o coordenador também citaram o fato do arcabouço fornecer funcionalidades prontas e uma estrutura que permite ao programador focar apenas em seu domínio. Um ponto negativo citado foi com relação a uma modelagem de objetos do arcabouço que, em um dado momento, limitou a flexibilidade do programador.

4.2.5 Métricas de código fonte

A seleção das métricas de código fonte como indicadores da qualidade do código foi retirada dos trabalhos de Li e Henry [18] e Chidamber e Kemerer [19]. Os dados foram obtidos usando o *plugin* para Eclipse chamado *Metrics* [49] analisando a totalidade do código dos dois aplicativos. A Tabela 2 apresenta os dados das duas versões do *iVProg* desenvolvidas, a anterior e a posterior à adoção da LPS, para comparação e análise, feita na próxima seção.

4.3 Análise dos dados e ameaças à validade

A análise de dados foi feita da seguinte maneira. As entrevistas foram transcritas e posteriormente tabuladas para resumir e sintetizar seu conteúdo. As informações mais relevantes eram quando o entrevistado falava sobre o método usado, sobre os resultados obtidos e sobre a relação entre os dois, para que durante a análise fosse possível correlacionar as causas e consequências. Foi dada ênfase na percepção das consequências provocadas por fatores não controlados (como a mudança na experiência do programador), com o objetivo de destacar aquelas provocadas diretamente pela LPS.

Com relação às métricas de código todo o código dos dois aplicativos foi analisado. Foram separados os resultados das principais métricas selecionadas pela bibliografia [18-19]. Em seguida os valores obtidos para os dois códigos foram tabulados e comparados.

De acordo com os dados coletados, é possível dizer, em linhas gerais, que o método de desenvolvimento utilizado pelo programador no segundo momento analisado foi aquele definido pela LPS. Além disso, o código fonte produzido possui uma boa qualidade pois possui valores dos indicadores próximos aos desejados. A produtividade percebida para as funcionalidades implementadas foi muito superior àquela antes da adoção da LPS e a impressão geral do programador com a LPS foi muito positiva. O fato do programador ter seguido o método definido pela LPS permite relacionar os dados coletados e conclusões feitas a partir deles com a utilização da LPS no desenvolvimento do *iVProg*. Assim, os valores das métricas obtidas na análise do código fonte e a percepção de produtividade são também produto do uso da LPS.

A comparação dos indicadores de qualidade de código entre as duas versões do *iVProg* mostra que a nova versão tem melhores valores nas métricas consideradas. As principais diferenças podem ser vistas nas métricas "métodos ponderados por classe", "acoplamento entre classes" e "falta de coesão entre métodos". A qualidade do código da nova versão é maior que a da anterior. Parte dessa qualidade é produto da utilização da LPS, conforme os relatos do programador.

Métricas	*iVProg* sem LPS		*iVProg* com LPS	
	Média	Máximo	Média	Máximo
Métodos Ponderados por Classe	13,5	383,0	7,54	30,0
Profundidade da Árvore de Heranças	4,07	10,0	4,90	8,00
Número de Filhos	0,69	17,0	0,62	18,0
Acoplamento entre Classes	7,35	44,0	3,15	9,00
Falta de Coesão entre Métodos	0,19	1,57	0,34	1,00

Tabela 2: Métricas de código fonte das versões do iVProg anterior e posterior à adoção da LPS em comparação com a referência desejada.

Além da qualidade do código, a LPS também influenciou o processo de desenvolvimento. O programador relatou que as etapas para a realização das tarefas eram claras e o ajudavam a realizar seu trabalho. Exemplos incluem o arcabouço forçar o programador a utilizar padrões de projeto e outras boas práticas de desenvolvimento. Sem a LPS, ele teria que definir sozinho a ordem e a forma de solução de alguns dos problemas enfrentados.

Por fim, uma terceira influência positiva da LPS é o fornecimento de funcionalidades prontas, que o programador não precisa desenvolver. Dessa forma, a LPS reduz o tempo de desenvolvimento de um aplicativo que se beneficia dessas funcionalidades.

Todas essas são evidências do aprimoramento do desenvolvimento do *iVProg* provocados pela LPS, que independem de fatores externos. Em comparação com a situação anterior, o processo de desenvolvimento com a LPS foi significativamente diferente, tanto por causa da LPS quanto de outros fatores, como a experiência do programador e a existência de código legado. A redução de alguns problemas enfrentados provavelmente foi provocada por fatores externos. Porém, os benefícios descritos acima são consequências da influência da LPS, seja por sua natureza ou justificados pelo relato do programador. Assim, por influência da LPS, em compa-ração ao utilizado na versão anterior, o processo de desenvolvimento pode ser considerado:

(i) mais produtivo, pois a LPS oferece funcionalidades reutilizáveis que não afetam negativamente o desenvolvimento de novas características;

(ii) que possui mecanismos para manter alta a qualidade do código, pois o arcabouço e o processo definido ajudam a estruturar o código e remover defeitos;

(iii) mais satisfatório para o programador, que relatou estar insatisfeito com o método anterior e que o código e a documentação atuais o ajudaram a superar dificuldades que seriam maiores caso não tivesse ajuda da LPS.

Essas foram as principais contribuições da LPS para o aprimoramento do processo de desenvolvimento do iMA. Para que os resultados descritos aqui, no caso específico do *iVProg*, possam ser utilizados por outros desenvolvedores de iMA, é necessário uma discussão adicional. Considerando as três principais contribuições provocadas pela adoção da LPS no desenvolvimento do *iVProg*, analisamos cada uma com relação ao caso da criação de outros iMA.

Primeiro, o aproveitamento das funcionalidades independentes de domínio fornecidas pelo arcabouço não é alterado para o caso de outros iMA, a menos que seja escolhido deliberadamente que o aplicativo não as utilizará. O código a ser reutilizado pode oferecer obstáculos para o desenvolvimento pelo tempo de aprendizado, o que pode reduzir o efeito positivo do reúso. Isso foi relatado em um caso pelo programador. Espera-se que esses obstáculos sejam reduzidos com o aprimoramento contínuo do código e dos manuais de utilização. Também espera-se que com a ampliação do número de iMA utilizando a LPS, a própria LPS como um todo possa ser aprimorada. Assim, ao longo do tempo o número de funcionalidades para serem reutilizadas aumentará e os obstáculos de aprendizagem serão reduzidos, permitindo que o desenvolvimento de qual-quer iMA aproveite dessa contribuição.

Em segundo lugar, consideramos a qualidade do código produzido no desenvolvimento de diferentes iMA. Essa contribuição é sensível à experiência do programador, sendo que programadores mais experientes são menos influenciados. Isso ocorre porque programadores iniciantes têm mais necessidade de ajuda para saber quando e como resolver problemas ou implementar funcionalidades. Assim, a LPS pode ter uma contribuição mais significativa quando os programadores têm menos experiência com desenvolvimento de software.

Por fim, a terceira contribuição é sobre a satisfação do desenvolvedor. No caso do *iVProg*, as tarefas se tornaram mais satisfatórias em comparação ao desenvolvimento anterior usando código legado. As instruções do método passo a passo, principalmente para desenvolvedores com pouca experiência, podem servir de guia para o programador e reduzir a sensação de falta de planejamento. Isso é uma facilidade, mesmo em comparação à implementação de um novo aplicativo, sem código legado.

Concluindo, a LPS apresentada neste trabalho contribuiu para aprimorar o desenvolvimento do *iVProg* e pode contribuir no caso de outros iMA. O desenvol-vimento de iMA em geral recebe mais benefícios da LPS nos casos em que o aplicativo utiliza todas as funcio-nalidades fornecidas e quando o programador possui pouca experiência em desenvolvimento de software.

5. Conclusões e trabalhos futuros

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

Os reúsos de código, arquitetura e processo promovidos pela LPS contribuem significativamente ao desenvol-

vimento de novos iMA. O programador economiza tempo por utilizar um código que teria de desenvolver e por não precisar pensar na estrutura geral de funcionamento do aplicativo, mas é beneficiado principalmente por dispor de funcionalidades básicas completamente prontas no núcleo comum. 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 de código.

Como subprodutos da pesquisa realizada podemos citar a centralização do conhecimento anteriormente disperso nos programadores de cada iMA, a divulgação do uso de método sistemático em contexto acadêmico de desenvolvimento de software educacional e a implementação de dois novos 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. Um desses trabalhos deverá ser a incorporação de padrões de anotação como o SCORM [50], o que permitirá que uma atividade elaborada com um iMA seja mais facilmente inserida no ambiente de curso, por exemplo, no *Moodle* [12]. Esse trabalho deve ser acompanhado da implementação de melhorias nos iMA existentes.

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.

Especificamente, a arquitetura pode receber melhorias para facilitar ainda mais o trabalho dos desenvolvedores ou se adaptar a funcionalidades adicionais. O código, por sua vez, deve ser constantemente atualizado para manter ou aumentar sua qualidade, prevenindo problemas de manutenção e evolução de software. A utilização de técnicas mais sofisticadas para o desenvolvimento do arcabouço, como anotações e metadados, pode ser aplicada com o objetivo principal de facilitar o uso do arcabouço pelos programadores.

Quando se menciona a expansão da LPS para iMA, trata-se principalmente do desenvolvimento de novos iMA, aumentando assim a família de aplicativos e a contribuição dos desenvolvedores para a sociedade. Cada iMA novo, desenvolvido com o auxílio da LPS, adiciona um novo domínio à família iMA, que podem ser empregados em mais atividades didáticas. Outras formas de expandir a LPS criada podem representar outros trabalhos futuros, como aqueles relacionados a aumentar o número de funcionalidades independentes de domínio fornecidas pelo arcabouço e aumentar o escopo dos aplicativos que podem ser criados com a linha.

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

[1] B. S. Bloom, The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13:4–16, 1984.

[2] J. M. Raines, L. M. Clark, A brief overview on using technology to engage students in mathematics. *Current Issues in Education*, 14, 2011.

[3] B. C. Tang. Interactive e-learning activities to engage learners - a simple classification. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 4092–4097, 2005.

[4] Y. Mor, N. Winters, Design approaches in technology-enhanced learning. *Interactive Learning Environments*, 15:61–75, 2007.

[5] A. M. Spalter, A. van Dam, Problems with using components in educational software. *Computers and Graphics*, 27:329–337, 2003.

[6] P. McAndrew, P. Goodyear, J. Dalziel, Patterns, designs and activities: unifying descriptions of learning structures. *International Journal of Learning Technology*, 2:126–242, 2006.

[7] E. Hinostroza, L. E. Rehbein, H. Mellar, C. Preston, Developing educational software: a professional tool perspective. *Education and Information Technologies*, 5:103–117, 2000.

[8] K. Pohl, G. Böckle, F. van der Linden. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Berlin Heidelberg New York. ISBN 3540243720, 2005.

[9] D. L. Dalmon, A. A. F. Brandão, L. O. Brandão, Uso de métodos e técnicas para desenvolvimento de software educacional em universidades brasileiras. *I Workshop de Desafios da Computação Aplicada à Educação*, 2012.

[10] T. K. Schleyer, L. A. Johnson, Evaluation of educational software. *Journal of Dental Education*, 67:1221–1229, 2003.

[11] D. L. Dalmon, A. A. F. Brandão, S. Isotani, L. O. Brandão, A Domain Engineering for Interactive Learning Modules. *Journal of Research and Practice in Information Technology*, 44:3, 2012.

[12] P. A. Rodrigues, L. O. Brandão, A. A. F. Brandão. Interactive assignment: a Moodle component to enrich the learning process. *Proceedings of the 40th Frontiers in Education*, 2009.

[13] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns. *Addison-Wesley Professional*. ISBN 0201703327, 2001.

[14] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-oriented domain analysis (FODA) feasibility study. Relatório técnico*, Carnegie Mellon University Software Engineering Institute*, 1990.

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

[16] N. Winters, Y. Mor, Idr: A participatory methodology for interdisciplinary design technology enhanced learning, *Computers and Education*, 50:579–600, 2008.

[17] J. C. Braga, S. Dotta, E. Pimentel, B. Stransky, Desafios para o Desenvolvimento de Objetos de Aprendizagem Reutilizáveis e de Qualidade. *I Workshop de Desafios da Computação Aplicada à Educação – DesafIE*, 2012.

[18] Wei Li, Sallie Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 1993.

[19] Shyam R. Chidamber e Chris F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20, 1994.

[20] Ian Douglas. Instructional design based on reusable learning objects: Applying lessons of object-oriented software engineering to learning systems design. *Proceedings of 31st Frontiers in Education Conference*, 2001.

[21] T. Boyle. Design principles for authoring dynamic, reusable learning objects. *Australian Journal of Educational Technology*, 19:46–58, 2003.

[22] R. P. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, 3, 2003.

[23] V. Pankratius. Aspect-oriented learning objects. *Proceedings of the IASTED International Conference on Web-based Education*, 2005.

[24] K. Ateyeh, P. C. Lockemann. Reuse- and aspect-oriented courseware development. Educational Technology and Society, 9:95–113, 2006.

[25] C. Choquet, A. Corbière, Reengineering framework for systems in education. Educational Technology and Society, 9:228–241, 2006.

[26] J. M. Dodero, T. Zarraonandia, C. Fernandez, D. Diez, Generative adaptation and reuse of competence development programmes, *Journal of Interactive Media in Education*, 4, 2007.

[27] R. I. Nicolson, P. J. Scott. Computers and education: the software production problem. *British Journal of Educational Technology*, 17(1):26–35 1986.

[28] J. Roschelle, J. Kaput, W. Stroup, T. M. Kahn. Scaleable integration of educational software: Exploring the promise of component architectures. *Journal of Interactive Media in Education*, 6:1–31, 1998.

[29] O. Conlan, C. Hockemeyer, V. Wade e D. Albert. Metadata driven approaches to facilitate adaptivity in personalized elearning systems. *Journal of Information and Systems in Education*, 1:38–44, 2002.

[30] M. L. Bote-Lorenzo, D. Hernández-Leo, Y. A. Dimitriadis, J. I. Asensio-Pérez, E. Gómez-Sánchez, G. Vega-Gorgojo, L. M. Vaquero-González. Towards reusability and tailorability in collaborative learning systems using ims-ld and grid services. *International Journal on Advanced Technology for Learning*, 1:129–138, 2004.

[31] A. van Dam, S. Becker e R. M. Simpson. Next generation education software - why we need it and a research agenda for getting it. *Educause Review*, 26–43, 2005.

[32] A. Oberweis, V. Pankratius e W. Stucky. Product lines for digital information products. *Information Systems*, 32:909–939, 2007.

[33] R. M. Reis. Development of educational software. *International Journal of Education and Information Technologies*, 1, 2007.

[34] B. Gadelha, E. Cirilo, H. Fuks, M. A. Gerosa, A. Castro, C. J. P. Lucena. Uma abordagem para o desenvolvimento de linhas de produto de groupware baseados em componentes utilizando o groupware workbench. *VII Simpósio Brasileiro*

de Sistemas Colaborativos, 32–38, 2010.

[35] L. S. Oliveira, M. A. Gerosa. Collaborative features in content sharing web 2.0 social networks: A domain engineering based on the 3c collaboration model. *17th CRIWG Conference on Collaboration and Technology*, 142–157, 2011.

[36] V. Stuikys, R. Damasevicius. Development of generative learning objects using feature diagrams and generative techniques. *Informatics in Education*, 7, 2008.

[37] F. Ahmed e I. Zualkernan. A software product line methodology for development of e-learning system. *International Journal of Computer Science and Emerging Technologies*, 2:285–295, 2011.

[38] B. C. Tang, Interactive e-learning activities to engage learners – a simple classification. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pp. 4092-4097, 2005.

[39] C. Quintana, J. Krajcik, E. Soloway, C. Norris. A framework for understanding the development of educational software. *The human-computer interaction handbook, editors J. A. Jack, A. Sears*, pp. 823-834, 2003.

[40] I. I. Bittencourt, P. Brito, A. Pedro, S. Isotani, P. A. Jaques, C. Rubira, 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*, 2012.

[41] H. Gomaa. Designing Software Product Lines with UML: from use cases to pattern-based software architectures. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA. ISBN 0201775956, 2004.

[42] L. O. Brandão, R. S. Ribeiro, A. A. F. Brandão, A system to help teaching and learning algorithms.. In: 2012 Frontiers in Education Conference, 2012, Seattle. 2012 Danvers: IEEE, 2012. v. 1. p. 1103-1108.

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

[44] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. *Addison-Wesley Professional*, 1994.

[45] E. M. Guerra, A Conceptual Model for Metadata-based Frameworks, Tese de Doutorado, Instituto Tecnológico de Aeronáutica, 2010.

[46] R. S. Wazlawick, Metodologia de Pesquisa em Ciência da Computação, Editora Campus, 2009.

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

[48] Robert K. Yin. Case Study Research: Design and Methods. Sage Publications, 4th ed. ISBN 1412960991, 2008.

[49] Sourceforge. Metrics plugin for eclipse. http://metrics.sourceforge.net/, 2012. Último acesso em 05/09/2013

[50] O. Bohl, J. Scheuhase, R. Sengler, U. Winand, The sharable content object reference model (SCORM) - a critical review, *Computers in Education*, 2002. vol.2, pp.950,951, 2002