Desafios da Engenharia de Software na Educação: Variabilidade de Sistemas Educacionais Inteligentes e Instanciação em Larga Escala

Ig Ibert Bittencourt[1], Patrick Brito[1], Alan Pedro[1], Seiji Isotani[2], Patrícia A. Jaques[3], Cecília Rubira[4]

NEES/UFAL[1], ICMC/USP[2], PIPCA/UNISINOS[3], IC/UNICAMP[4]

Abstract. *Intelligent Educational Systems have been used by millions of user by providing different requirements, common changes of needs, context awareness, and so on. The production of systems with these features has several challenges related to software engineering, mainly with regards to dynamic software product line. The goal of this paper is to present how the Brazilian educational challenges are connected to the challenges faced by the software engineering community.*

Resumo. *Sistemas educacionais inteligentes têm sido utilizados por milhares ou milhões de usuários que demandam requisitos distintos, constantes mudanças de necessidades, sensibilidade ao domínio de conhecimento, entre outras. A produção de sistemas com estas características gera diversos desafios relacionados à engenharia de software, mais especificamente da área de linha de produto de software dinâmica. O objetivo deste artigo é abordar como os desafios educacionais brasileiros estão em consonância com desafios atualmente enfrentados pela comunidade de engenharia de software.*

1. Introdução

A engenharia de software é uma disciplina da computação relacionada com a produção de software de alta qualidade, respeitando os requisitos e as restrições de tempo e custo (Sommervile, 2011). Diversos são os aspectos que tornam o desenvolvimento de sistemas de software complexos. Os principais fatores são relacionados à exigência de requisitos de qualidade do software, tais como facilidade de manutenção, escalabilidade, segurança e disponibilidade. Essa complexidade é potencializada ainda mais em aplicações que possuem milhões de usuários com necessidades de requisitos distintos, constantes mudanças de necessidades dos usuários, a sensibilidade ao contexto, separação de assuntos, etc. Diante disto, variabilidade é um conceito abordado na engenharia de software voltada para a produção de famílias de sistemas.

Variabilidade de software é a capacidade de um sistema de software ou artefato ser modificado, personalizado ou configurado, para ser utilizado em um contexto específico (Gurp, Bosch e Svahnberg, 2001). Um alto grau de variabilidade permite a utilização do software em diferentes contextos, o que torna o artefato mais fácil de ser reutilizado. Por essa razão, lidar com a variabilidade nas diversas fases do ciclo de vida do software vem sendo uma preocupação constante da engenharia de software. No contexto de desenvolvimento de software baseado em componentes, por exemplo, a separação entre a especificação e implementação de um componente favorece uma

maior variabilidade de implementação. Um outro tipo de variabilidade estrutural está relacionado à composição de componentes e conectores na arquitetura de software. O fato de um sistema poder ser composto por diferentes versões de componentes e conectores de software favorece o projeto e desenvolvimento de sistemas de software com grande variabilidade funcional e não-funcional.

Com o objetivo de antecipar alguns tipos de variabilidade e sistematizar o reuso entre sistemas similares, surgiu o conceito de Linhas de Produto de Software (LPS), há várias evidências reais de que a utilização de LPS traz benefícios consideráveis no desenvolvimento de produtos com alto grau de similaridade entre si (Clements e Northrop, 2002; Bosch, 2005; Steger et al., 2004; Verlage e Kiesgen, 2005; Silva, 2012). Os benefícios relatados são tanto em termos de redução de custo e esforço, quanto na melhoria da qualidade e produtividade.

Apesar das vantagens em se utilizar LPS, há grandes desafios sendo tratados por pesquisas atuais. Tais desafios estão relacionados com a modularização de software, reuso em larga escala, evolução de aplicações, evolução da LPS, produção efetiva de famílias de linhas, gerenciamento de variabilidades dinâmicas, interferência dinâmica entre características, projeto e implantação de LPS, entre outros. Entretanto, uma das áreas que representa diversos desafios e que poderia ser melhor explorada pela comunidade de engenharia de software equivale à educação. O desenvolvimento efetivo e ágil de aplicações educacionais tem sido um grande desafio tanto para a comunidade científica e da indústria de software no Brasil quanto no exterior. Dentre os grandes desafios de engenharia de software enfrentado pela indústria de software educacional, têm-se: produção em larga escala de sistemas educacionais, a evolução dinâmica e autonômica de aplicações educacionais, a produção de famílias de produtos e autoria destes sistemas educacionais pelos professores. Por um lado, a comunidade de engenharia de software, mais especificamente de variabilidade e linha de produto de software, tem atacado os problemas supracitados. Por outro lado, a área de educação tem desafios que se mostram em consonância com os desafios da comunidade de engenharia de software.

Este artigo tem por objetivo abordar tanto os desafios relacionados com a produção efetiva e ágil de sistemas educacionais inteligentes e ao mesmo tempo os desafios que atualmente são enfrentados pela comunidade de linha de produto de software. Desta forma, apresenta-se neste artigo como estas duas áreas podem trabalhar em conjunto para que os desafios enfrentados pelo Brasil possam ser superados para uma educação de qualidade mediada pelas tecnologias da informação e comunicação.

2. Informática na Educação Brasileira

A Educação é a base para o crescimento sustentável de qualquer nação. Diversas nações que fizeram investimentos numa educação de qualidade tiveram grandes resultados, que vão desde o crescimento no número de investimentos em todos os setores, estímulo e aumento da inovação, repatriação de profissionais, diminuição da corrupção, redução do desemprego, aumento da qualidade de vida, e finalmente aumento do Produto Interno Bruto (PIB) e da Felicidade Interna Bruta (FIB).

É também evidente que estes resultados ocorrem num longo prazo e que as políticas públicas devem ser direcionadas para os diferentes níveis de formação de um

cidadão. Ou seja, quanto maior a escala alcançada pela melhoria da qualidade do ensino e da formação, maior o potencial de aumento do PIB e FIB.

Entretanto, a necessidade de escalabilidade gera outro problema, principalmente em países em desenvolvimento como do BRICAS que possuem grande diversidade populacional e extensão territorial. O problema equivale à universalização do acesso à informação, problema este que já foi destacado tanto no Plano Nacional de Educação 2010 – 2020, quanto nos Grandes Desafios da Computação 2006 – 2016[1]. Este problema torna-se ainda maior, sendo considerado um grande desafio, quando se considera a universalização da informação de qualidade. De fato, este problema não é novo e pode ser inclusive encontrar motivações na Constituição Federal de 1988, de acordo com os direitos e garantias fundamentais dos cidadãos.

Atacar este desafio representa uma orquestração de decisões estratégicas em nível federal articuladas com as esferas estaduais e municipais, e composta por políticas públicas em prol de melhoria da qualidade do ensino, da transparência e da formação de professores que atendam à demanda nacional. Quando se considera toda esta orquestração, fica evidente que a mediação tecnológica torna-se um pilar importante de toda esta estrutura. A Lei de Diretrizes e Bases da Educação Nacional define os níveis e modalidades de educação e ensino. Quando se pense numa educação mediada por tecnologia, deve-se levar em consideração, por um lado, as garantias individuais e coletivas dos cidadãos e, por outro lado, a composição dos níveis escolares.

Diante da grande diversidade presente nos cidadãos, o desafio é ainda mais potencializado, fazendo com que a mediação tecnológica seja diferenciada de acordo com o público no qual a tecnologia será utilizada. Isto faz com que diversos campos de investigação da Informática na Educação sejam aplicados de acordo com o público. Por exemplo, para a educação infantil e ensino fundamental, acredita-se que uma abordagem lúdica seja mais adequada. Entretanto, para o ensino fundamental, talvez seja mais adequadas abordagens baseadas em *gamefication* e sistemas tutores inteligentes (aprendizagem individualizada) (Bittencourt, 2009; 2011). Já para a educação superior, observa-se uma maior necessidade de um modelo baseado na colaboração com sistemas tutoriais inteligentes. Para os alunos do EJA e educação profissionalizante, há a necessidade de utilização de agentes pedagógicos animados (provenientes da computação afetiva). Com relação à educação especial, há a necessidade de usar diferentes recursos multimidiáticos associados a diferentes abordagens citadas anteriormente. Observa-se que dentro de cada um destes níveis educacionais e de ensino, a mesma variabilidade pode ser encontrada. Além disso, há a necessidade de uma computação que busque a personalização da instrução de acordo com as características do usuário, necessitando de sistemas educacionais inteligentes.

É importante frisar que independente do nível e da modalidade de educação e ensino, a variabilidade de recursos é altíssima, tornando-se um grande desafio para os profissionais de computação. Destaca-se ainda que não é interesse deste artigo levantar o desafio de uma única solução de engenharia de software que cubra os diversos níveis e modalidade de ensino e educação. O que se pretende abordar é que cada uma destas

[1] Disponível em http://www.gta.ufrj.br/rebu/arquivos/SBC-Grandes.pdf

modalidades possuem uma imensa escala, ao mesmo tempo em que se tem uma grande variabilidade de recursos a considerar.

3. Linha de Produto de Software Dinâmica

Uma Linha de Produto de Software (LPS) define um conjunto de produtos de software com alto grau de similaridade entre si, que atendem às necessidades específicas de um segmento de mercado e compartilham um conjunto de artefatos básicos (Clements e Northrop, 2002), (Muthig et al., 2004). De maneira geral, em uma LPS, os artefatos de software devem ser flexíveis o suficiente de modo a permitir que detalhes de implementação de um produto específico possam ser postergados para fases posteriores do desenvolvimento. Estas decisões de projeto postergadas são denominadas de pontos de variação. Um ponto de variação é o local do artefato de software em que uma decisão de projeto pode ser tomada, e variantes são alternativas de projeto associadas a este ponto (Van Gurp, Bosch e Svahnberg, 2001). Dessa forma, em uma LPS, o domínio de aplicações é utilizado tanto para tratar similaridades entre sistemas similares, quanto para definir o escopo da infra-estrutura de reutilização.

Uma família de produtos é um conjunto de produtos construídos a partir de um conjunto de artefatos comuns, tais como arquitetura e um conjunto de componentes. O gerenciamento das funcionalidades comuns e variáveis da família de produtos não é uma tarefa simples, e para isso a engenharia de linha de produto divide o ciclo de vida de desenvolvimento em duas fases complementares (Muthig et al., 2004): engenharia de família, e engenharia de aplicação. A engenharia de família consiste em analisar os potenciais produtos de um domínio ou empresa em relação às suas funcionalidades comuns e variáveis, que são então utilizadas para construir uma infraestrutura para a linha de produtos, que contemple tanto artefatos reutilizáveis, quanto a utilização sistemática destes artefatos na construção de novas aplicações. De forma complementar, a engenharia de aplicação é responsável por instaniar diferentes produtos de maneira sistemática, a partir da infraestrutura de artefatos reutilizáveis que foram desenvolvidos na engenharia de família.

Além das duas fases já comentadas, segundo o *Software Engineering Institute* (SEI), para uma implantação efetiva de LPS se faz necessária uma terceira fase denominada gerenciamento técnico e organizacional, que contempla atividades como gerência de configuração, análise de mercado, entre outras. Em resumo, o desenvolvimento de uma LPS envolve o desenvolvimento de bens comuns (*Core Asset Development*), o desenvolvimento de produtos utilizando estes bens comuns (*Product Development*), através de um gerenciamento técnico e organizacional (*Management*). O desenvolvimento de bens comuns é equivalente à engenharia de família, e o desenvolvimento de produtos é equivalente à engenharia de aplicação. As três atividades são representadas como círculos em rotação, indicando que estão intrinsecamente ligadas, a saída de uma realimenta a entrada da outra, e são iterativas.

Uma das principais limitações de LPS é o fato da engenharia de aplicação ocorrer de maneira estática, isto é, em tempo de projeto. Porém, alguns tipos de sistemas, normalmente denominados adaptativos, necessitam de mudança de variabilidade em tempo de execução. Quando a LPS permite que a fase de engenharia de aplicação seja executada após a instanciação do produto (variabilidade atrasada ou *late variability*), torna-se possível considerar ajustes nos requisitos em tempo de

execução. Linhas de produto com essa característica são consideradas linhas de produto de software dinâmicas (LPSDs) (Gomaa e Hussein, 2004), (Hallsteinsen et al, 2008).

A área de LPSD é considerada uma área de pesquisa emergente, que pode sistematizar a configuração de sistemas de software que adaptam seus requisitos em tempo de execução. Dessa forma, LPSDs reduzem consideravelmente a complexidade de se lidar com o gerenciamento de reconfiguração dinâmica, uma vez que essa questão é tratada explicitamente na linha de produtos, através do conceito de variabilidade atrasada. Em geral, LPSDs possuem duas fases de engenharia de aplicação: estática e dinâmica. A engenharia de aplicação estática é realizada da mesma maneira que na LPS tradicional e nessa são selecionadas todas as características que o produto possa necessitar. Em seguida, essas características podem ser ativadas e desativadas em tempo de execução (engenharia de aplicação dinâmica). Um desafio de pesquisa relacionado à LPSD concerne à detecção e resolução de inconsistências relacionadas à interação entre características (i.e. *feature interaction*) durante a engenharia de aplicação dinâmica.

4. Linhas de Produto de Software Dinâmicas aplicadas à Educação

Esta seção tem por objetivo abordar os desafios tanto para a comunidade de computação quanto para a comunidade que trabalha com educação. Consideram-se fundamentalmente os sistemas educacionais inteligentes que podem/devem ser aplicados aos diferentes níveis e modalidades de educação e ensino vigentes na Lei de Diretrizes e Bases da Educação Brasileira.

Com o objetivo de apresentar os desafios da forma mais objetiva e clara possível, esta seção focará num tipo de sistema educacional inteligente aplicado ao contexto de educação superior, a saber, os Sistemas Tutores Inteligentes (STIs). Entretanto, frisa-se que o desafio é estendido para os outros níveis de educação e com outras abordagens da IA aplicada à Educação.

4.1 Diferentes requisitos

O primeiro desafio está relacionado com a variabilidade de funcionalidades presentes nos sistemas educacionais inteligentes. Ao se construir sistemas tutores inteligentes, deve-se projetar os seguintes modelos: (i) modelo do domínio de conhecimento: este modelo possui o conhecimento sobre o que deve ser ensinado para o estudante. Ou seja, equivale aos conteúdos de conhecimento específico (e.g. cálculo, anatomia, programação); (ii) modelo pedagógico: este modelo possui estratégicas e táticas sobre como ensinar a um estudante um determinado assunto específico ou problema; (iii) modelo do estudante: neste modelo estão presentes as informações do estudante referentes ao domínio de conhecimento em questão. Além dos três modelos citados, ainda há o módulo de interface que é responsável por refletir todas as decisões do sistema na interface de comunicação com o usuário. Sendo assim, um sistemas de software voltados para educação devem lidar bem com diversos domínios de conhecimento, que potencialmente utilizarão recursos computacionais diferentes, tais como textos e conteúdos multimídia.

Deve-se considerar que os domínios de conhecimento podem ser bem definidos (do inglês *well-defined domains*) ou mal definidos (do inglês *ill-defined domains*). De acordo com a característica do domínio de conhecimento, os requisitos do STI podem ser completamente diferentes. Além disso, para que o ambiente possa se adaptar às

necessidades dos usuários pode ser importante considerar não somente o conhecimento do estudante no domínio de conhecimento, mas também outras características dos mesmos (e.g. emoção e personalidade). Frisa-se ainda que as estratégias e táticas de ensino podem variar de acordo com as características do domínio de conhecimento e as especificidades de cada estudante. Não bastando, a variabilidade presente nos requisitos supracitados terá impacto direto na interface do usuário, havendo assim a necessidade de adaptar também essa a interface. Potencializa-se ainda este desafio ao considerar que os sistemas atualmente adotados na educação superior são inerentemente colaborativos e utilizados no modelo 24x7 (i.e. 24 horas por dia 7 dias na semana), tornando fundamental que as mudanças de requisitos ocorram sem a necessidade do sistema ficar *off-line*.

Do ponto de vista de LPS, percebe-se dois tipos de desafios. O primeiro desafio é relacionado à variabilidade estática do sistema, durante a instanciação do STI, tais como disponibilizar ou não da funcionalidade de "chat" para todos os domínios de conhecimento. O segundo desafio refere-se à variabilidade dinâmica, isto é, habilitar ao STI de se adaptar automaticamente, não só às particularidades de cada domínio de conhecimento, mas também às características individuais de cada estudante. A questão da melhoria de manutenibilidade do sistema, minimizando a necessidade do sistema ficar *off-line* também é endereçado pelas LPS, uma vez que o fato de se considerar variabilidades explicitamente, inclusive na arquitetura de software, promove maior isolamento entre as funcionalidades do sistema.

Desafio 1: Construção de Linhas de Produtos de Software Dinâmicas que atendem à variabilidade de requisitos dos Sistemas Educacionais Inteligentes.

4.2 Diferentes domínios de conhecimento

O segundo desafio que se aborda em se construir sistemas tutores inteligentes para a educação superior está relacionado com a variedade de domínios e subdomínios de conhecimento presentes na educação superior. Além disso, os STIs são sistemas *dependentes de domínio de conhecimento* enquanto que os sistemas voltados para a educação superior (e.g. AVAs) são sistemas *independentes de domínio de conhecimento*.

Este desafio não está somente na produção de recursos educacionais (e.g. Vídeo-aula, exercícios, provas, textos) para os diferentes domínios de conhecimento, mas também na granularidade dos recursos produzidos para cada domínio. Além disso, o conteúdo deve levar em consideração que tipo de aluno está tendo acesso ao mesmo. Por exemplo, os recursos educacionais de algoritmo para alunos do curso de computação terão um nível de detalhamento maior do que alunos do curso de meteorologia; ou há alunos que preferem explicações de assuntos através de conteúdos textuais enquanto outros preferem vídeo-aula. Este problema é potencializado ainda mais, pois os STIs devem possuir o *Modelo do Domínio de Conhecimento* que está sendo ensinado, uma vez que espera-se que nesse tipo de sistema, o mapeamento das dificuldades do estudante seja realizado automaticamente, de acordo com o conteúdo sendo aprendido e a sua interação com o sistema. Apesar da sua dependência de domínio de conhecimento, para que os STIs possam ser utilizados em larga escala na educação superior, eles deveriam ser independentes de domínio. Entretanto, mesmo que

sejam produzidos STIs independentes de domínio, há a necessidade de proporcionar o reuso do conhecimento gerado. O reuso e compartilhamento de conhecimento deve ocorrer sob diferentes perspectivas: (i) requisito: equivale a um conhecimento fundamental de determinada disciplina (e.g. ensinar diagrama de *venn* para um curso de teoria dos conjuntos); (ii) pré-requisito: há situações que um conhecimento se torna pré-requisito de outro conhecimento ou outro domínio (e.g. teoria dos conjuntos é pré-requisito ao ensinar probabilidade); (iii) adicional: equivale ao conhecimento que pode ser um adicional para determinado conhecimento, representando um conhecimento lateral (e.g. teoria dos conjuntos é conhecimento lateral ao se ensinar álgebra booleana). Do ponto de vista de projeto, faz-se necessário um estudo de como possíveis modelos podem ser utilizados. Por exemplo, em qual situação uma rede bayesina pode ser utilizada como modelo pedagógico, ou modelo de domínio?

Do ponto de vista da engenharia de software, o desafio especificado aqui é facilitar o desenvolvimento em larga escala de aplicações educacionais como STIs, que são dependentes do domínio de conhecimento. Para isso, o conceito de LPS pode ser um aliado bastante eficaz, uma vez que é possível definir uma família de STIs, onde as características de cada STI gerado fossem selecionadas automaticamente, a partir de características do domínio de conhecimento.

Desafio 2: Reuso e Compartilhamento de Conhecimento em Linhas de Produto de Software para Sistemas Educacionais Inteligentes independentes de domínio.

4.3 Instanciação em Larga escala

O terceiro desafio está relacionado à escalabilidade para geração de produtos de uma mesma família. Quando se está interessado em construir sistemas educacionais inteligentes que sejam utilizados nos diferentes domínios de conhecimento, surge outro problema equivalendo à produção em larga escala dos sistemas. Por exemplo, um instituto de computação que possui 5 cursos, sendo 3 de graduação (ciência da computação, engenharia de computação e sistema de informação), 1 especialização (engenharia de software) e 1 mestrado (informática). Para este instituto, deve-se considerar o número de disciplinas presentes em cada curso para que daí seja construído um STI para cada disciplina. Com isso, dado que cada curso de graduação possua 45 disciplinas com 15 da especialização e 10 do mestrado, somam-se um total de 160 Sistemas Tutores Inteligentes a serem construídos.

Este desafio é potencializado quando se considera todo o contexto de uma universidade. Em uma universidade de médio porte, haverá a necessidade de construir mais de 6 mil sistemas tutores inteligentes. No caso da instanciação de diferentes STIs em larga escala, essa seria a vantagem mais imediata do conceito de LPS, uma vez que se trata da aplicação direta da fase de Engenharia de Aplicação (Seção 3), presente explicitamente no processo de criação da linha de produtos.

Desafio 3: Instanciação de Sistemas Educacionais Inteligentes em Larga Escala.

4.4 Autoria de Produtos

Os sistemas educacionais inteligentes possuem requisitos e características onde as decisões devem ser tomadas, normalmente, por especialistas do domínio de

conhecimento no qual se pretende construir determinado STI. Desta forma, tais especialistas devem decidir qual conhecimento deve ser reutilizado, quais devem ser compartilhados, quais requisitos devem ser reutilizados, bem como quais estratégias e táticas pedagógicos o sistema deve adotar com os estudantes.

Diante deste cenário, os especialistas do domínio são os professores de cada disciplina a ser ensinada em cada período letivo (i.e. anual ou semestral). No início de cada período letivo o professor deve configurar o STI de sua disciplina. Desta forma, torna-se essencial a participação ativa dos professores na geração dos produtos de software. Sabendo que não se pode exigir conhecimento de desenvolvimento de software de todos os professores, essa abstração do desenvolvimento torna-se mais um desafio a ser superado. É neste contexto que surge o quarto desafio.

Para atender a este desafio, vale a pena, mais uma vez, lançar mão do conceito de LPS. Mais especificamente, a fase de engenharia de aplicação, já mencionada no desafio anterior, possibilita que as decisões relacionadas à instanciação dos produtos de uma LPS sejam realizadas no contexto de um modelo de características (do inglês *feature model*) [Clements e Northrop, 2002]. Tal modelo representa, em um alto nível de abstração, variabilidades relacionadas aos requisitos e demais características do domínio de aplicação, no nosso exemplo, STIs. Sendo assim, não é necessário conhecimento de computação para a instanciação de diferentes produtos de software. Dessa forma, os profissionais de educação não só estão aptos a instanciar diferentes produtos, mas são os profissionais mais indicados para tal.

Desafio 4: Autoria de Linhas de Produto de Software Dinâmicas.

4.5 Autoria ágil

O quinto desafio está intimamente relacionado com o quarto desafio. Os professores devem ser capazes de gerar os próprios sistemas educacionais inteligentes para cada disciplina no qual for ensinar. Como dito anteriormente, isto ocorre no começo de cada período letivo. Entretanto, é senso comum que o professor possui um tempo relativamente curto (i.e. semanas) para o planejamento de todas as disciplinas que necessita ensinar. Além disso, o professor está envolvido em diversas atividades que torna ainda mais curto o tempo disponível para o planejamento da disciplina. Diante disto, o tempo para a preparação do sistema educacional inteligente será também muito curto. Isto faz com que o sistema necessite de bastante automatização para a autoria dos produtos de tal forma que o mesmo possa ser instanciado de forma rápida. Caso o produto não seja fácil e rápido de ser instanciado, cairá no desuso por parte dos professores prejudicando todo o esforço despendido no desenvolvimento e na superação dos desafios citados anteriormente. Com isso, o quinto desafio que surge é sobre a autoria ágil de produtos. Além de facilitar a instanciação de produtos por não especialistas de computação, após a seleção das características desejadas para o produto, como mencionado no desafio anterior, o restante do processo de engenharia de aplicação é, normalmente executado automaticamente. Essa automação agiliza consideravelmente a criação dos produtos.

Desafio 5: Rápida Instanciação de Sistemas Educacionais Inteligentes por Autores.

4.6 Família de Linhas Educacionais

O sexto e último desafio está relacionado com o nível de variabilidade que os sistemas educacionais inteligentes necessitam. Os STI adaptam-se de acordo com as características dos usuários durante suas interações com o sistema. Além disso, as decisões sobre quais requisitos, estratégias e táticas pedagógicas dependem dos professores que preparam o STI de cada domínio de conhecimento. Pode-se com isso perceber que a variabilidade presente nestes tipos de produtos ocorre em dois níveis diferentes: (i) tempo de projeto: as características do STI que serão instanciadas dependem do domínio de conhecimento e serão definidas pelo professor; (ii) tempo de execução: ocorre dinamicamente, pois o STI deve decidir quais das características disponíveis melhores se adéquam às características atuais de um determinado estudante. De fato, o professor não escolhe qual requisito específico será utilizado, mas sim quais conjuntos de requisitos devem estar disponíveis para que o STI, em tempo de execução, possa decidir qual característica deve ser utilizada com cada estudante de acordo com o estado atual do seu modelo. Isto faz com que se tenham dois níveis de decisão (um estático e outro dinâmico) com relação ao modelo de decisão presente em uma linha de produto de software.

Esse é, sem dúvida, o desafio mais interessante e que possui grande potencial de contribuição nas duas áreas: informática na educação e engenharia de software. A engenharia de aplicação de uma SPL tradicional considera a instanciação sistemática de produtos apenas em tempo de projeto, isto é, instanciação estática. Para considerar a adaptação dinâmica dos produtos é necessário adentrar na área de Linha de Produto de Software Dinâmica (LPSD). Porém, ao observar que cada modelo de estudante pode possuir um estado único em relação aos demais estudantes, chega-se à conclusão de que, na verdade, deve ser mantido um STI para cada estudante, e não apenas para cada disciplina do curso, como esperado inicialmente. Sendo assim, nesse cenário real demandado pela educação, vê-se claramente a necessidade de considerar a instanciação dos produtos da LPS em dois níveis: (i) instanciação das "sub-famílias" da linha, que deve ser realizada para cada disciplina, pelo professor e de maneira estática (o termo "sub-família" foi usado no sentido de um produto estático, mas que ainda não representa a última versão que será executada por cada estudante, uma vez que ainda será personalizado dinamicamente); e (ii) instanciação dos produtos, que deve ser realizado automaticamente pelo STI, de maneira dinâmica. Em relação à instanciação dinâmica dos produtos, destacasse aqui o grande potencial de tecnologias largamente utilizadas em educação para a área de LPSD. Um exemplo disso é o conceito de ontologia, que por representar estruturas semanticamente, podem favorecer a resolução de um dos grandes problemas de LPSD: realizar verificações dinâmicas da consistência do modelo de características, para impedir combinações incompatíveis de características (*feature interaction*).

Desafio 6: Construção de uma Família de Linhas de Produto de Software Dinâmicas para Sistemas Educacionais Inteligentes.

5. Conclusão

Este artigo teve por objetivo descrever de forma sucinta como as pesquisas em educação podem se beneficiar da engenharia de software na produção de software com

alta qualidade, que efetivamente atendam às necessidades dos milhões de usuários e que possam se adaptar de acordo com a demanda solicitada em cada cenário educacional. Por questão de escopo e espaço, este trabalho focou principalmente em aspectos desafiadores ligados à informática na educação e o conceito de linhas de produto de software. Porém, outros desafios também importantes do ponto de vista de informática na educação e engenharia de software não foram discutidos, tais como: escalabilidade, desempenho e disponibilidade.

Referências

Bittencourt, Ig Ibert ; COSTA, Evandro de Barros ; Marlos Silva ; SOARES, Elvys . A Computational Model for Developing Semantic Web-based Educational Systems. Knowledge-Based Systems, v. 22, p. 302-315, 2009.

Bittencourt, Ig Ibert ; COSTA, Evandro de Barros . Modelos e Ferramentas para a Construção de Sistemas Educacionais Adaptativos e Semânticos. Revista Brasileira de Informação na Educação, v. 19, p. 85-98, 2011.

Bosch, Jan. Software Product Families in Nokia. In Software Product Lines: 9th International Conference (SPLC 2005), Rennes, France, pages 2-6, 2005.

Clements, Paul and Northrop, Linda. Salion, Inc.: A Software Product Line Case Study. CMU/SEI-2002-TR-038 ESC-TR-2002-038, 2002.

Gomaa, H. and Hussein, M. Dynamic software reconfiguration in software product families. LNCS, pages 435-444, 2004.

Hallsteinsen, S.; Hinchey, M.; Park, S.; and Schmid, K. Dynamic software product lines. Computer, 41(4):93-95, 2008.

Muthig, Dirk; John, Isabel; Anastasopoulos, Michalis; Forster, Thomas; Dörr, Jörg; and Schmids, Klaus. GoPhone - A Software Product Line in the Mobile Phone Domain. IESE-Report No. 025.04/E, Version 1.0, March 5, S 2004.

SEI, Software Engineering Institute. 2012. “A Framework for Software Product Line Practice, Version 5.0”. Disponível em: http://www.sei.cmu.edu/productlines/frame_report/config.man.htm. acessado em Abril de 2012.

Silva, Alan Pedro da ; Costa, Evandro ; Bittencourt, Ig Ibert . Uma Linha de Produto de Software baseada na Web Semântica para Sistemas Tutores Inteligentes. Revista Brasileira de Informação na Educação, v. 20, p. 87, 2012.

Steger, Mirjam; Tischer, Christian; Boss, Birgit; Müller, Andreas; Pertler, Oliver; Stolz, Wolfgang; and Ferber, Stefan. Introducing PLA at Bosch Gasoline Systems: Experiences and Practices. In Software Product Line Conference (SPLC), 2004.

Sommerville, Ian. Engenharia de Software. 9º Edição. Ed. Pearson, 2011.

Van-Gurp, Jilles; Bosch, Jan; and Svahnberg, Mikael. On the Notion of Variability in Software Product Lines. In Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01), 2001.

Verlage, Martin and Kiesgen, Thomas. Five years of product line engineering in a small company. In ICSE '05: Proceedings of the 27th international conference on Software engineering, pages 534-543, New York, NY, USA, 2005. ACM Press.