Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

EasyT
Apoiando a Construção de Objetos de Aprendizagem para uma Linha de Produtos de Software

Jônatas Castro dos Santos

JUIZ DE FORA
OUTUBRO, 2012

EasyT
Apoiando a Construção de Objetos de Aprendizagem para uma Linha de Produtos de Software

Jônatas Castro dos Santos

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência de Computação
Bacharelado em Ciência da Computação

Orientador: José Maria Nazar David
Co-orientador: Fernanda Claudia Alves Campos

JUIZ DE FORA
OUTUBRO, 2012

EASYT

Apoiando a Construção de Objetos de Aprendizagem para uma Linha de Produtos de Software

Jônatas Castro dos Santos

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

José Maria Nazar David
DSc.

Fernanda Claudia Alves Campos
DSc.

Liamara Scortegagna
DSc.

JUIZ DE FORA

29 DE OUTUBRO, 2012

Aos pais, pelo apoio e sustento.

A minha namorada, pelo amor, paciência e carinho.

Resumo

O cenário das tecnologias disponíveis atualmente favorece cada vez mais a criação de componentes para apoiar a construção de ferramentas voltadas para o ensino e a aprendizagem. Tais componentes são frequentemente denominados de Objetos de Aprendizagem (OAs). Entretanto, a construção de OAs torna-se cada vez mais complexa considerando-se as necessidades que as ferramentas demandam. Para tratar essa complexidade a abordagem de Linha de Produtos de Software (LPS) tem sido frequentemente proposta. LPS é caracterizada como um conjunto de componentes que compartilham características comuns e gerenciadas com o objetivo de satisfazer um determinado cenário de segmento de mercado ou missão. Este trabalho busca relacionar a abordagem de LPS para apoiar o reuso sistemático de OAs, bem como propõe uma solução para apoiar a construção de um tipo de OA que irá compor a LPS. Esta solução é um passo em direção à construção de uma LPS para apoiar o reuso de Objetos de Aprendizagem.

Palavras-chave: Objetos de Aprendizagem, Linha de Produtos de Software, Objetos de Aprendizagem Generativo.

Abstract

The scenario of the available technology increasingly favors the creation of components to support the construction of tools to support teaching and learning. Such components are often called Learning Objects (LOs). However, the construction of LOs becomes increasingly complex, considering the requirements demanded by the tools. To address this complexity, Software Product Line (SPL) approach has usually been proposed. SPL is characterized as a set of components that share common characteristics and that is managed in order to fulfill requirements of a particular scenario of market segment or mission. This work aims to discuss SPL approach to support the systematic reuse of LOs as well as to propose a solution in order to semi automatically create LOs. This is one step towards SPL construction to support Learning Objects reuse.

Keywords: Learning Objects, Software Product Line, Generative Learning Objects.

Agradecimentos

A Deus por seu amor, graça e misercórdia. Deus é o único motivo para minha existência.

Aos meus pais Eli e Laureci. Sem vocês eu não seria capaz de concluir este curso. Obrigado pelo esforço que fizeram para me manter aqui. Obrigado pelo amor, carinho e aconselhamento.

A minha namorada Celinha por cada "boa noite"e "eu te amo"que me deu durante todo este tempo. Obrigado pela paciência que tem tido comigo. Você é parte integrante da minha vida! Eu te amo!

A minha irmã Elaine e meu cunhado Luis Fernando pela amizade, aconselhamento, reciprocidade e cumplicidade.

Ao Pr. João Aine e família que conviveram comigo nos primeiros anos em Juiz de Fora. Foram uma família para mim durante esse tempo.

Aos meus orientadores e professores José Maria e Fernanda. Agradeço por terem feito possível a produção do artigo que levou a este trabalho, além de contribuir para que eu viajasse para o exterior.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

A todos os meus amigos e colegas de curso pela parceria e amizade, em especial, Adriano, Brunão, Gazzola, Amaral, Claudson, Rogerson, Marcus, Átila, Fernando Paiva, Alan e Marcelão.

"Porque dEle e por Ele, e para Ele, são todas as coisas; glória, pois, a Ele eternamente. Amém.".

Romanos 11:36

Sumário

Lista de Figuras

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
NEnC	Núcleo de Engenharia do Conhecimento
BROAD	Busca e Recuperação de Objetos de Aprendizagem
SEI	Software Engineering Institute
LPS	Linha de Produtos de Software
OA	Objeto de Aprendizagem
LOM	Learning Object Metadata

1 Introdução

Objeto de Aprendizagem (OA) pode ser caracterizado como um recurso para ajudar no processo de ensino e ser reutilizado em diversos contextos (Najjar et al., 2003). Wiley (2000) define OA como quaisquer recursos digitais que podem ser utilizados para o ensino e que são construídos de forma a dividir o conteúdo em pequenos módulos reutilizáveis em diversos ambientes seguindo os princípios da orientação a objetos. A reutilização é um conceito chave para Objetos de Aprendizagem (Burbaite e Stuikys, 2011). Desse modo, um OA deve ser granular, isto é, deve ser pequeno ou composto de pequenas partes. Quanto maior o grau de granularidade de um OA, maior será a sua possibilidade de reuso. Um OA pode conter várias partes que formam vários OA granulares (Burbaite e Stuikys, 2011).

Para organizar os Objetos de Aprendizagem, foram criados repositórios. Basicamente, eles fornecem mecanismos que facilitam na recuperação de objetos para o usuário. Nestes repositórios podem ser utilizados metadados para identificar os OAs. Metadado é definido como "informação sobre um objeto; seja físico ou digital"(Najjar et al., 2003).

Para facilitar a expansão da abordagem de OAs, são criados padrões de metadados, como por exemplo, o IEEE LOM (Learning Object Metadata) (Wiley, 2000). A criação de padrões como o LOM possibilita a interoperabilidade entre as tecnologias de aprendizagem das instituições que utilizam repositórios de OAs (Wiley, 2000). Para o contexto deste trabalho, "interoperabilidade é a capacidade de um sistema de hardware ou de software de se comunicar e trabalhar efetivamente no intercâmbio de dados com outro sistema, geralmente de tipo diferente, projetado e produzido por outro fabricante (Sayao e Marcondesm, 2008)". É comum que projetistas de repositórios de OAs selecionem atributos de mais de um padrão de metadados, conforme a necessidade da organização. A especificação desta seleção de atributos é chamada de "perfil de aplicação", ou apenas "perfil"(Najjar et al., 2003).

Najjar et al. (2003) propõem uma abordagem para promover a interoperabilidade entre a estrutura de metadados do ARIADNE e o IEEE LOM. Para atingir este objetivo,

faz-se um mapeamento sistemático do perfil do ARIADNE no padrão LOM. O grande desafio nesse contexto é o de identificar mapeamentos coerentes dos campos do metadado e respectivos conjuntos de valores.

No contexto de desenvolvimento de software para a área de educação, a construção de OAs que potencializem a reutilização se faz necessária dada a complexidade dos processos envolvidos. Mecanismos são necessários para agilizar o processo de construção de OAs e facilitar a integração deles com repositórios como o do projeto BROAD - Busca e Recuperação de Objetos de Aprendizagem a Distância. O repositório deste projeto oferece um conjunto próprio de metadados baseado em um estudo de diversos padrões como o IEEE LOM, ARIADNE e SCORM (Campos et al., 2011).

O objetivo deste trabalho é apoiar a construção de OAs e integrá-los ao repositório do BROAD de forma automática. Para tanto, será apresentada uma proposta para desenvolvimento de OA que será o ponto de partida para a implantação de uma Linha de Produtos de Software de OAs que utilizam metadados para apoiar a busca e recuperação desses artefatos.

A primeira seção deste trabalho, após a introdução, apresenta a fundamentação teórica sobre Linha de Produtos e Objetos de Aprendizagem. Posteriormente, são discutidos os trabalhos relacionados ao tema da pesquisa. Como o OA proposto neste artigo utiliza o repositório do projeto BROAD-PL, na sessão seguinte, este projeto será apresentado. Em seguida, será mostrada a especificação, projeto, implementação e avaliação da proposta de solução para apoiar a construção de OAs. Finalmente, o artigo apresenta as conclusões e sugestões para trabalhos futuros.

2 Referencial teórico

2.1 Linha de produtos de software

Os estudos sobre Linha de Produtos de Software (LPS) têm atingido importância na área de Engenharia de Software. A necessidade de fabricar softwares customizados de grande porte e alta complexidade tem incentivado as práticas de reuso em LPS com o objetivo de diminuir o esforço e o tempo e aumentar a qualidade (Lee e Dirk, 2012). Por isso, muitas organizações já adotaram práticas de LPS com sucesso.

Linha de Produtos de Software é definida pelo SEI (2011) como um conjunto intensivo de sistemas de software que compartilham e gerenciam características comuns e variabilidades, satisfazem as necessidades de um segmento particular de mercado ou missão e são desenvolvidos a partir de um núcleo de artefatos. Núcleo de artefatos é uma base que contém artefatos comuns de uma família de softwares. Tais artefatos são itens reutilizáveis utilizados como bloco de construção de uma linha de produtos.

LPS pode ser caracterizada como uma família de produtos de software, onde cada membro pode compartilhar características semelhantes e possuir características distintas. Estas características são parametrizadas e gerenciadas para gerar produtos com o objetivo de atender as necessidades do cliente.

Geralmente, o escopo de uma LPS é especificado quando os produtos já existentes de uma organização possuem muitas características em comum (McGregor et al., 2002). O escopo é a descrição dos produtos que irão constituir a linha de produto (Clements e Northrop, 2001). Ele determina o que pertence à LPS, isto é, delimita os objetivos da LPS. A partir da descrição do escopo deve-se analisar como os membros da família de produtos irão se diferenciar entre si. É importante que ele esteja alinhado a diversos fatores, como por exemplo, as direções econômicas do segmento de mercado alinhado a LPS, o montante de recursos que serão injetados para desenvolvimento do núcleo de artefatos, o grau de variabilidade no comportamento dos produtos de software e a evolução dos artefatos no tempo (McGregor et al., 2002).

Uma das principais atividades presentes na LPS é a de representar as variabilidades que podem ocorrer entre os artefatos da LPS. As variabilidades são observadas a partir da diferenciação entre *features* de produtos. Basicamente, *feature* pode ser definida como uma característica que o usuário final considera importante na descrição e distinção de membros de uma família de produtos (Griss, 2000). As *features* de uma LPS podem ser representadas através de um modelo. Com este modelo é possível visualizar e analisar as características comuns e variabilidades entre os produtos da LPS.

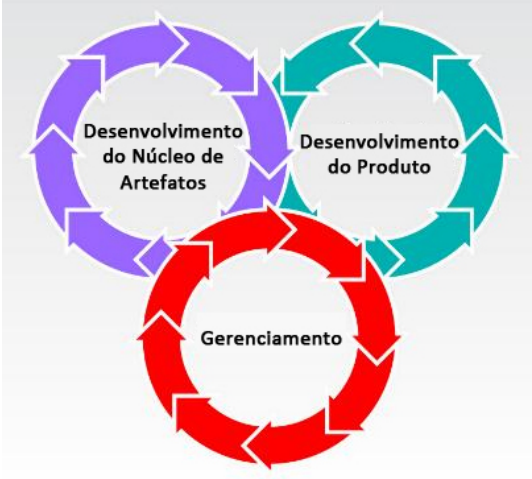

Figura 2.1: Ciclos de vida da LPS (SEI, 2011).

Sendo assim, abordagem de LPS tradicional pode ser divida em três ciclos, conforme ilustrados na Figura 2.1 (SEI, 2011): (i) o primeiro é o desenvolvimento do núcleo de artefatos, também conhecido como Engenharia de Domínio; (ii) o segundo ciclo trata-se do desenvolvimento do produto, ou Engenharia de Aplicação; e (iii) o terceiro ciclo pode ser chamado de Gerenciamento do Produto (Gimenez et al., 2002). O desenvolvimento do núcleo de artefatos e o desenvolvimento do produto se traduzem na essência da LPS. Associada a estes dois ciclos está a atividade de gerenciamento. Não existe uma ordem pré-definida em que eles ocorrem e o impacto causado em um ciclo pode influenciar os outros (Clements e Northrop, 2001).

A Engenharia de Domínio é responsável por estabelecer uma plataforma reutilizável na linha de produtos, isto é, o núcleo de artefatos. Estarão presentes nesta plataforma todos os artefatos que definem aspectos comuns e variabilidades da LPS (Silva, 2011). O núcleo de artefatos constitui a base para a geração de produtos na LPS. Para definir as *features* dos artefatos, é necessário que se tenha definido o escopo da LPS. Este

assume um papel importante na Engenharia de Domínio. Todas as mudanças que podem ocorrer com o surgimento de novas oportunidades de negócio, no segmento de mercado e no plano organizacional podem impactar no escopo (Clements e Northrop, 2001). A análise realizada no escopo deve gerar como saída um modelo de domínio.

O modelo de domínio é devidamente validado e documentado sendo este a base para o projeto da arquitetura genérica da LPS. Dentre os artefatos de LPS, a sua arquitetura genérica tem um papel de suma importância, pois ela representa a infraestrutura central da LPS. Com a arquitetura pode-se definir as estruturas gerais que compõem o sistema, as quais descrevem cada componente e as interações entre eles (Gimenez et al., 2002). Para (McGregor et al., 2002), "a arquitetura é chave de uma linha de produto bem sucedida".

Com o projeto de arquitetura genérica, é possível então partir para o segundo ciclo de LPS, o desenvolvimento do produto, ou Engenharia de Aplicação. Este ciclo é responsável por derivar a aplicação a partir da plataforma estabelecida na Engenharia de Domínio (Silva, 2011).

Nesta etapa, o modelo de domínio é utilizado para selecionar informações relevantes do produto específico a ser construído. Além dessas informações, outros requisitos obtidos com o cliente que não estão no modelo de domínio serão necessários para a aplicação. Pode ser que estes requisitos exijam a extensão da LPS e ainda, a aquisição de novos componentes para a LPS. As listas de características (*features*) obtidas com o modelo de domínio, mais os outros requisitos específicos da aplicação, compõem a especificação do novo produto (Gimenez et al., 2002).

Após a especificação do produto, faz-se necessário estabelecer uma arquitetura específica a partir da arquitetura genérica. O processo de criação desta arquitetura consiste em analisar várias vezes a arquitetura genérica e tomar decisões recursivamente sobre esta, moldando assim a arquitetura específica do produto. Analiticamente, pode haver pontos de variações e decisões causadas por requisitos que fogem da arquitetura genérica. A solução que resulta dessas decisões dependerá da perspectiva de utilização dos requisitos por outros clientes (Gimenez et al., 2002).

Finalmente, o ciclo de vida da Engenharia de Aplicação é concluído com a criação

de componentes de acordo com a arquitetura. A fonte desses componentes poderá ser: o núcleo de artefatos, uma nova implementação ou ainda aplicações legadas.

O terceiro ciclo da LPS, o Gerenciamento do Produto, tem como objetivo gerenciar os recursos utilizados na LPS e garantir que as atividades sejam coordenadas e supervisionadas. Para este ciclo, deve ser estabelecido um plano de adoção da LPS que descreva o estado desejado e as estratégias que serão utilizadas (Lazilha, 2002). A separação lógica entre a Engenharia de Domínio e a Engenharia de Aplicação caracteriza a primeira geração de metodologias de LPS. Cada vez que um novo produto é inserido na LP, é necessário manter uma nova equipe de desenvolvedores dedicados ao produto. Outro problema frequente é que cada produto pode agregar um contexto único e isolado em que os artefatos são reutilizados. Em casos não triviais, isso dificulta o reaproveitamento do núcleo de artefatos (Kueger, 2006).

Estudos mais recentes possibilitaram a criação de novos métodos para apoiar o reuso sistemático em LPS e diminuir os problemas das metodologias da primeira geração. A metodologia de Customização de Software em Massa, por exemplo, diminui os efeitos negativos da Engenharia de Aplicação através da automatização deste ciclo. Em vez de utilizar uma equipe voltada apenas para o desenvolvimento do produto, são utilizados "configuradores de LPS"para instanciar produtos automaticamente a partir de duas entradas: modelos de *features* e núcleos de artefatos. Dessa forma, todos os produtos gerados dependem dos ativos do núcleo de artefatos. A vantagem desta abordagem é que todo o trabalho existente na LPS é focado apenas no desenvolvimento e manutenção do núcleo de artefatos. Neste viés, mudanças no núcleo de artefatos implicam que todos os produtos serão atualizados de forma automática, sem intervenções manuais (Kueger, 2006).

A estratégia de LPS pode apoiar outras áreas em relação à sistematização do reuso. É o caso, por exemplo, de estudos que unem LPS com o paradigma de Orientação a Serviços (OS) (Medeiros et al., 2010) (Ribeiro et al., 2010). "Serviço é definido como um ato ou desempenho oferecido de uma parte para outra. Embora o processo possa ser vinculado a um produto físico, o desempenho é essencialmente intangível e normalmente não resulta na posse de qualquer um dos fatores de produção"(Sommerville, 2011). No

contexto de software, a essência do paradigma de OS é que o fornecimento do serviço é independente da aplicação que o usa. Os provedores de serviço podem desenvolver serviços especializados e oferecê-lo para uma variedade de usuários de diferentes organizações (Sommerville, 2011). O paradigma de OS pode potencializar o reuso e a interoperabilidade entre sistemas (Turner et al., 2011). Os serviços podem ser implementados utilizando a tecnologia de *web service*s. Geralmente, um *web service* é uma representação padrão para algum recurso computacional ou de informações que podem ser usadas por outros programas (Sommerville, 2011).

Software como serviço (Turner et al., 2011) necessita de uma abordagem sistemática para apoiar a adoção ou utilização de componentes. Portanto, LPS em um contexto de OS pode ajudar a criar serviços de forma sistemática, menos custosa e customizável de acordo com as necessidades do cliente.

Medeiros et al. (2010) propõem uma abordagem para desenvolvimento de um conjunto de sistemas orientados a serviço como uma linha de produtos de software. A abordagem, chamada SOPLE-DE, serve para a identificação, projeto e documentação sistemática de um núcleo de artefatos orientados a serviço que apoia o reuso não sistemático. O SOPLE-DE recebe como entrada um modelo de *features*, um modelo de processo de negócio e cenários de atributos de qualidade. Como saída, é gerada uma arquitetura descrevendo a linha de produtos orientada a serviço com os componentes e serviços e as ligações que ocorrem entre eles.

Abordagem de LPS com o foco em serviços também pode ser aplicada no contexto de construção de Objetos de Aprendizagem.

2.2 Objetos de aprendizagem

No contexto de desenvolvimento de software para a área de educação, a construção de artefatos que potencializem a reutilização faz necessária dada a complexidade dos processos envolvidos. Tais artefatos estão associados, sobretudo aos Objetos de Aprendizagem (OAs).

Objeto de Aprendizagem é "uma entidade, digital ou não digital que pode ser usada, reusada ou referenciada durante o ensino com suporte tecnológico. Exemplos de

ensino com suporte tecnológico incluem sistemas de treinamento baseados no computador, ambientes de aprendizagem interativos, sistemas instrucionais auxiliados por computador, sistemas de ensino a distância e ambientes de aprendizagem colaborativa. Exemplos de objetos de aprendizagem incluem conteúdo multimídia, conteúdos instrucionais, objetivos de ensino, software instrucional e software em geral e pessoas, organizações ou eventos referenciados durante um ensino com suporte tecnológico"(IEEE, 2000).

Wiley (2000) define OA como "qualquer recurso digital que pode ser reutilizado para apoiar o ensino". Esta definição deixa claro que um OA não pode ser "não digital", como por exemplo, um livro impresso, e ainda, o recurso deve possuir a capacidade de ser reutilizado. Como este trabalho trata de produtos de software esta definição será utilizada.

Os OAs devem ser projetados de forma a integrar-se com outros e, esta junção deve possibilitar a utilização de recursos em contextos mais amplos. Esta característica diz respeito à modularidade e granularidade de um OA. A modularidade é a capacidade de uma entidade de decompor em partes com pouco grau de dependência uma das outras (Gracindo, 2009). Um OA pode ser composto de vários módulos ou pode ser simplesmente um módulo. A granularidade é o nível de detalhamento de uma entidade. Quanto maior a granularidade de um OA, maior será a sua possibilidade de reuso. Pode ser que um OA contenha várias partes que podem ser extraídas formando vários OA granulares (Burbaite e Stuikys, 2011).

Outras características que permeiam os OAs são: reusabilidade, autonomia, interatividade, interoperabilidade e facilidade de busca. Reutilização é a capacidade e flexibilidade para ser utilizado mais de uma vez e em diferentes situações. A autonomia é capacidade de OA funcionar sem depender de outras aplicações para funcionamento e entendimento. A interatividade ocorre quando um OA propicia a participação do usuário de forma bidirecional, sendo o mesmo capaz de interferir no processo. A interoperabilidade é a possibilidade de um OA funcionar em diferentes plataformas sem a necessidade de alterar suas características e a facilidade de busca diz respeito à flexibilidade de um usuário recuperar um OA específico em um conjunto de OAs (Gracindo, 2009) (Vieira, 2007).

O projeto de OAs pode ser muito complexo. Deve levar em conta tanto aspectos inerentes a teorias de aprendizagem como o conhecimento de áreas como engenharia de software, além de levar em conta as potencialidades e limitações da tecnologia envolvida. Construir OAs demanda tempo e necessita de habilidades multidisciplinares (Konrath, 2006).

3 Infraestrutura BROAD-PL

A localização e recuperação de OA exigem que eles tenham sido cadastrados a partir de um modelo robusto de metadados, capaz de descrevê-lo de forma completa e correta, com vistas a facilitar seu uso. Ocorre que há muitos padrões disponíveis, mas eles quase sempre são difíceis de serem adotados, além de serem extensos, complexos e muitas vezes negligenciarem os aspectos educacionais envolvidos no OA. O Projeto BROAD - Busca e Recuperação de Objetos de Aprendizado a Distância - é um projeto de pesquisa contínuo que utiliza tecnologias como *web service*s, ontologias, agentes, serviços web semânticos e workflow no desenvolvimento de infraestruturas com anotações semânticas na Web em repositórios heterogêneos de dados (Braga et al., 2011).

Com o avanço da pesquisa no projeto BROAD, novas arquiteturas têm sido propostas conforme a adoção de novas tecnologias: BROAD-WP (Workflow) e BROAD-PL (Linha de Produtos).

A infraestrutura BROAD-WP busca estender a infraestrutura BROAD através da utilização de diferentes Objetos de Aprendizagem de acordo com as características do aprendiz. A oferta de artefatos educacionais em diferentes mídias é uma forma de personalizar a aprendizagem, e atender diferenças individuais tanto de preferências quanto de necessidades físicas especiais.

As especificações do padrão de metadados BROAD auxiliam o sequenciamento das atividades de aprendizagem e permitem identificar as preferências e características de um estudante. A infraestrutura BROAD-WP propõe apresentar a especificação de uma arquitetura para geração de workflow de conteúdos educacionais de forma a personalizar a oferta desses módulos. A personalização é feita pela geração de um workflow, que considera o conteúdo a ser estudado, o Perfil do Estudante e os diferentes artefatos educacionais disponíveis no repositório. Para a definição do workflow a proposta se apóia numa ontologia de Objetos de Aprendizagem, já desenvolvida, a qual busca promover a semântica do ambiente (Braga et al., 2011).

A infraestrutura BROAD-PL visa maximizar o reuso de OA para facilitar a com-

posição dos módulos educacionais. Considerando os mecanismos que simplificam o processo de construir OAs reutilizáveis e facilitar a integração deles com repositórios, o uso da abordagem de Linha de Produtos de Software deve oferecer um suporte sistemático para reuso.

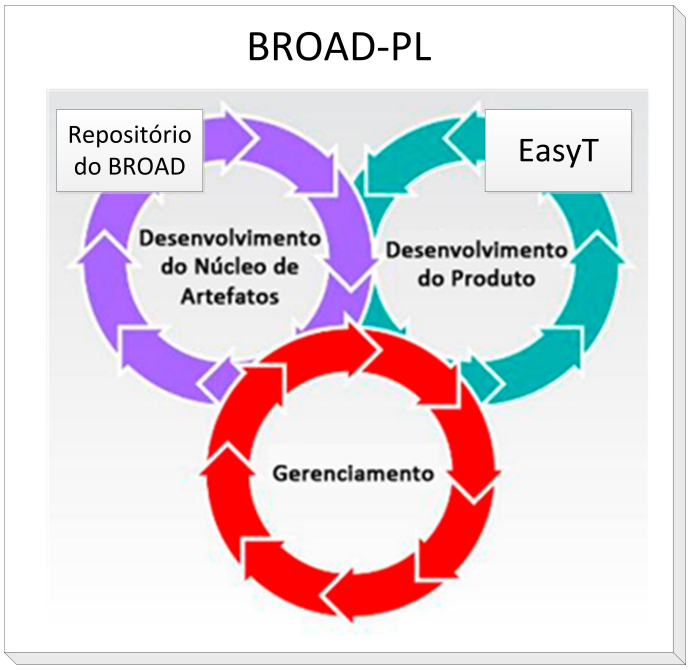

Figura 3.1: BROAD-PL e os ciclos de vida da LPS. Modificado de (SEI, 2011).

O projeto BROAD compõe parte do ciclo de vida da LPS (figura 3.1). O repositório de OAs do BROAD apresenta uma estrutura que se encaixa no contexto ciclo de Desenvolvimento do Núcleo de Artefatos. O repositório do BROAD é capaz de registrar dados de um OA, permitindo a sua busca e recuperação posterior. Como os OAs são artefatos reutilizáveis, o repositório, portanto, se assemelha a um núcleo de artefatos, parte essencial da LPS. Entretanto, outros aspectos envolvidos na busca e recuperação necessitam ser tratados posteriormente, tais como, a gerência de configuração dos artefatos armazenados.

A ferramenta Easy Tutorial E-learning (EasyT), proposta neste trabalho, deve apoiar o Desenvolvimento do Produto no BROAD-PL. Esta ferramenta tem como objetivo ajudar na construção de OAs e integração deles ao núcleo de artefatos, isto é, o repositório do BROAD. Trata-se de um template de tutorial Hipermídia e será apresentada na sessão seguinte.

3.1 Trabalhos relacionados

A especificação, criação e manipulação de Objetos de Aprendizagem (OA) são desafios que têm sido frequentemente discutidos em trabalhos na área de e-learning (Burbaite e Stuikys, 2011) (Campos et al., 2011) (Damasevicius e Stuikys, 2009) (Stuikys e Damasevicius, 2008). De modo geral, ainda não existe um consenso sobre uma metodologia ou um padrão para apoiar a construção de OAs.

Em (Burbaite e Stuikys, 2011), (Damasevicius e Stuikys, 2009) e (Stuikys e Damasevicius, 2008) é discutido o conceito de OA Generativo (OAG). Do ponto de vista tecnológico, um OAG é um OA genérico que pode ser instanciado gerando OAs específicos. O objetivo dos OAGs é aumentar o grau de reusabilidade através da adaptação de conteúdo e capacidade de geração automática ou semiautomática de OAs.

De acordo com Burbaite e Stuikys (2011), esse tipo de OA reduz significativamente o esforço para a sua construção. Os autores descrevem estudos de casos que utilizam modelos de *features* para representar OAs. Entretanto, eles não propõem uma forma para apoiar a implementação dos OAs.

Segundo Morales et al. (2005), a criação de OAGs pode ser divida em duas etapas: a criação de um Template de OA (TOA) e a adição de um conteúdo de assunto específico. O TOA deverá conter a estrutura ou o projeto de aprendizagem. Após a criação do TOA, tutores ou professores poderão adicionar o conteúdo específico de uma disciplina ou assunto - estrutura superficial - para produzir vários OAs de acordo com as necessidades pedagógicas de cada um.

Tanto o modelo de *features* como o conceito de OAGs estão fortemente associados ao conceito de Linha de Produtos de Software (McGregor et al., 2002) (Damasevicius e Stuikys, 2009). Essa relação é explicitada em (Damasevicius e Stuikys, 2009) da seguinte forma: "cada diagrama de *features* descreve uma família de produtos", contudo, o resultado da seleção de *features* é um produto específico. Esses conceitos são utilizados por Damasevicius e Stuikys (Damasevicius e Stuikys, 2009) (Stuikys e Damasevicius, 2008) para especificar e modelar sequências de OAs.

Um diagrama de *features* estendido é derivado do diagrama proposto por Burbaite e Stuikys (Burbaite e Stuikys, 2011) através da técnica de especialização parcial. A técnica

consiste em derivar versões menos genéricas de entidades de domínios que possuem um subconjunto de *features* resolvido, enquanto outros poderão ser tratados posteriormente (Damasevicius e Stuikys, 2009). Desse modo, as variabilidades de *features* podem ser classificadas como variabilidades fixas, resolvidas ou adiadas e, portanto, as sequências podem ser construídas a partir de *features* fixas, resolvidas ou adiadas. Features fixas são parâmetros que permanecem com o mesmo valor em todo o conjunto de sequências. Features resolvidas são aquelas que são usadas para construir uma sequência e são resolvidas sequencialmente. Features adiadas são aquelas que permanecem não resolvidas, mas poderão ser resolvidas posteriormente quando o usuário final (aluno, professor) decidir sobre a construção de uma sequência de produtos (Damasevicius e Stuikys, 2009).

Após a análise do modelo de *features* estendido, o estudo de caso apresentado em (Damasevicius e Stuikys, 2009) descreve a implementação das sequências de OAs utilizando o conceito de OAG e meta-programação. A implementação resulta na geração semiautomática de sequência de OAs e deve ser alimentada por conjunto de OAs já existentes. No entanto, a interoperabilidade dessas sequências com repositórios é um problema que não foi tratado em (Damasevicius e Stuikys, 2009).

Já existem, portanto, ideias de como utilizar LPS para apoiar a construção de OAs. Os modelos baseados em *features* dão uma visão abrangente acerca das variabilidades e aspectos comuns de softwares. A utilização destes modelos se mostra fundamental na especificação de OAG para a implantação de uma Linha de Produtos de OAs. Entretanto, não vemos uma solução que resolva o problema da interoperabilidade dos OAs gerados com repositórios de OAs.

4 EasyT: Uma proposta para a construção de objetos de aprendizagem

Easy Tutorial E-learning (EasyT) é um template que visa facilitar a construção e a integração de OAs com repositórios. Trata-se de um Tutorial Hipermídia - documento hipertexto dinâmico que permite estabelecer ligações entre páginas e embutir recursos de páginas HTML. O EasyT é formado por uma ou mais páginas distintas. Cada página, além do texto formatado, pode conter qualquer tipo de mídia externa - imagem, áudio, vídeo ou animação - e ligações para outros OAs. O template permite a geração de OAs específicos e a integração automática com repositórios de OA. Os metadados do EasyT são carregados uma única vez pelo autor e ficam armazenados no tutorial. Quando se deseja integrar o tutorial a um repositório, os metadados serão cadastrados automaticamente através do mecanismo presente no template.

4.1 Especificação da solução

A Figura 4.1 apresenta os requisitos funcionais do EasyT. A solução proposta permite visualizar páginas, recuperar metadados do tutorial, metadados de cada página e cadastrar automaticamente o tutorial em repositórios de OA.

Através de um índice ordenado alfabeticamente por título das páginas o usuário poderá acessar uma página e visualizá-la. Portanto, o índice contém as ligações para qualquer página do tutorial. Cada página possui uma identificação e um conjunto de metadados de OA que descrevem informações, tais como, título, autor, idioma, versão etc. Esses metadados são apresentados no momento da visualização de cada página. Dessa forma o usuário pode recuperar essas informações. Além dos metadados de cada página, o tutorial também possui seus próprios metadados, os quais são mostrados na tela inicial do tutorial.

Para integrar o EasyT e suas partes aos repositórios de OA, os metadados do

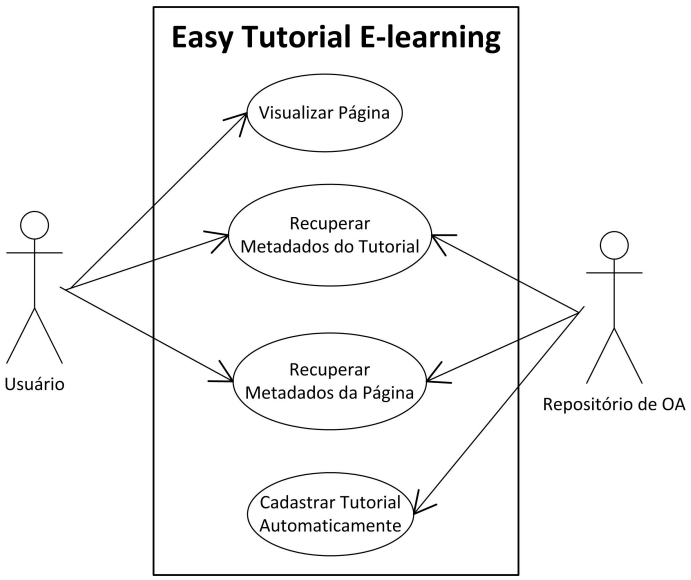

Figura 4.1: Diagrama de casos de uso do ferramenta Easy Tutorial E-learning.

tutorial podem ser recuperados através de um mecanismo programável. Esse mecanismo provê que o tutorial seja facilmente cadastrado (automático) em repositórios de OA. Dessa forma, não há necessidade que o usuário redija os metadados do EasyT no momento do cadastramento em algum repositório de OA.

O tutorial possui um padrão de metadados que é compatível com os mecanismos de busca do projeto BROAD (Campos et al., 2011). Isso não significa que a integração do EasyT esteja restrita somente ao repositório do BROAD, entretanto, antes de promover a integração do tutorial com outros repositórios é necessário verificar a compatibilidade de metadados. Esta verificação é necessária por causa do mecanismo de interoperabilidade presente na ferramenta que será apresentado mais adiante.

Com o objetivo de facilitar os testes de implementação, foi especificado um subconjunto de metadados do projeto BROAD, presentes também na maioria dos padrões internacionais. Os metadados escolhidos foram: title, author, description, language, keyword e version. Os metadados title e author são essenciais para o funcionamento do template já que estes aparecerão diretamente na interface. A escolha dos outros metadados foi arbitrária, pois eles não são mandatórios para o tutorial. Entretanto, é necessário verificar a compatibilidade dos metadados escolhidos com aqueles do repositório que se deseja integrar com o EasyT. Para exemplificar esta questão da compatibilidade, temos que: se o repositório que se deseja integrar com o EasyT possui este subconjunto de me-

tadados, e este subconjunto é suficiente para o cadastro do OA no repositório, podemos assumir que o repositório aceita o mecanismo de interoperabilidade do EasyT.

A seguir serão apresentados os requisitos não funcionais que foram elicitados para o EasyT.

A interoperabilidade é um requisito não funcional necessário para a ferramenta, uma vez que visa promover flexibilidade na interação do OA com repositório, OAs e outras aplicações. O mecanismo de interoperabilidade presente na ferramenta permite que repositórios e outras aplicações recuperem os metadados do OA de forma programável. Dessa forma, o esquema do tutorial permite que sistemas com diferentes plataformas consigam integrá-lo. Quando o EasyT é cadastrado no repositório do BROAD, o mecanismo permite que todas as páginas sejam automaticamente cadastradas como se fossem novos OAs. Dessa maneira, temos como recuperar qualquer parte do tutorial no repositório.

A portabilidade é outro requisito não funcional necessário visto que o OA gerado com o EasyT precisa ser facilmente acessado de qualquer dispositivo e lugar. A maneira mais trivial de satisfazer este requisito é disponibilizando o tutorial em formato HTML através de um sistema dinâmico de páginas. Atualmente, as tecnologias permitem a visualização deste formato sem nenhuma condição altamente específica de hardware ou software.

A reusabilidade é um requisito chave para o OA implementado como um serviço, pois influencia diversos fatores importantes, tais como (Erl, 2009): (i) Autonomia - o conteúdo do tutorial e suas páginas devem ser autossuficientes, isto é, devem depender pouco de outras aplicações para funcionamento e entendimento. Dessa forma, um tutorial autônomo fornece ambiente para promover a reusabilidade; (ii) Fraco acoplamento - o tutorial deve ser independente entre os módulos. Isto significa que uma alteração no tutorial, deverá afetar pouco os OAs que estejam ligados a ele; (iii) Composição - quanto maior o grau de reusabilidade do tutorial maior será a chance de ele compor outros OAs; (iv) Abstração - é a forma que o OA é empacotado ou disponibilizado para os utilizadores; (v) Localização - é o mecanismo pelo qual o tutorial pode ser buscado e recuperado. O EasyT é composto por páginas independentes, sendo que cada página possui seu próprio conjunto de metadados. Desta forma, temos como garantir que cada página possa ser

reutilizada independentemente em diferentes contextos. A ferramenta é empacotada por um *web service* que disponibiliza os metadados do tutorial e suas páginas.

4.2 Projeto e implementação da solução

O EasyT pode ser caracterizado como um Template de Objeto de Aprendizagem e, ao mesmo tempo, um Objeto de Aprendizagem Generativo (OAG). Ele é um template de OA porque constitui uma estrutura base que aguarda a inclusão de um conteúdo específico para gerar um OA específico. Também é considerado um OAG porque se trata de uma estrutura genérica que será instanciada em estruturas específicas. Para modelarmos analiticamente esse cenário, utilizamos o diagrama de *features*. Este modelo é importante para obtermos aspectos comuns e variabilidades que as instâncias do OAG podem ter. Analisando o diagrama de *features*, da Figura 4.2, o EasyT pode ser composto por uma ou mais páginas. Para cada página temos a *feature* **<Metadados de Página>**, mas esta *feature* não é mandatória, i.e., o EasyT não depende desta *feature* para funcionar. As páginas também podem conter **<Recursos>** que também não são mandatórios. Esta *feature* pode conter as seguintes *features*: **<Imagem>**,**<Áudio>** ou **<Video>**.

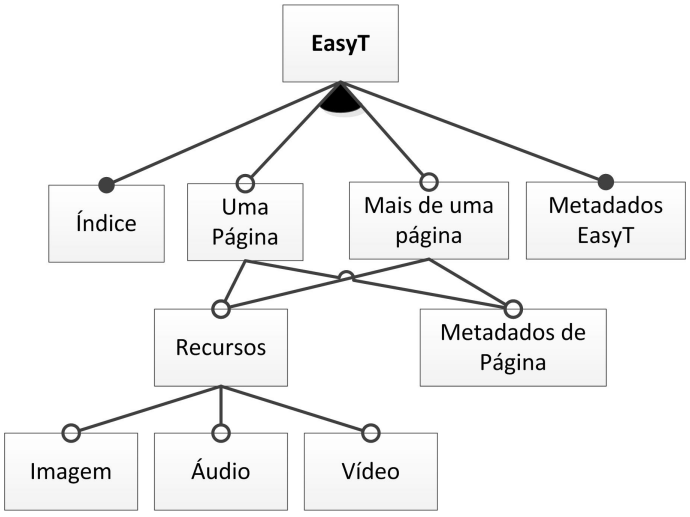

Figura 4.2: Diagrama de Features do Easy Tutorial E-learning.

A *feature* **<Metadados EasyT>** indica que o EasyT deve conter metadados, que são diferentes dos metadados de página que são obrigatórios. A *feature* **<Índice>** trata do índice de páginas. Apesar de este índice ser gerado automaticamente (conforme a composição das páginas), ele é mandatório. Com a especificação e análise baseada no

diagrama *features* temos uma visão abrangente sobre o que será necessário no projeto do tutorial. O EasyT é um sistema orientado a serviços. O uso deste paradigma visa facilitar o entendimento, separar responsabilidades e a facilitar manipulação dos dados no uso do mecanismo de suporte à interoperabilidade. Então, para modelar o sistema, utilizamos o diagrama de classes da UML (Figura 4.3).

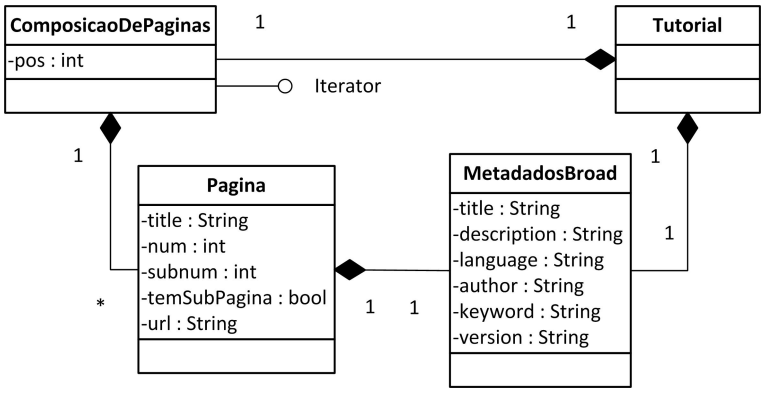

Figura 4.3: Diagrama de classes da ferramenta Easy Tutorial E-learning.

Conforme a Figura 4.3, a classe **<Tutorial>** é uma composição das classes **<Paginas>** e **<MetadadosBroad>**. A classe **<ComposicaoDePaginas>** contém uma coleção de páginas as quais são compostas de um ou mais objetos do tipo **<Pagina>**. Para iterar sobre estes objetos, a classe **<ComposicaoDePaginas>** implementa os métodos da interface nativa **<Iterator>**. A classe **<MetadadosBroad>** deve conter como atributos cada metadado utilizado no repositório do BROAD. Nesse caso, o subconjunto reduzido dos metadados do projeto BROAD foi encapsulado. As classes **<Tutorial>** e **<Pagina>** são compostas por objetos do tipo **<MetadadosBroad>**. A classe **<Pagina>** possui como atributos *num*, *subnum*, *temSubPagina* e *title*. Estes atributos estão relacionados com a identificação de uma página e ajudam na organização do índice de cada página. O atributo *num* indica o número de página e o atributo *subnum* indica o número de subpágina. Na implementação atual, o tutorial pode conter um conjunto de páginas e subpáginas. Uma subpágina está a um nível abaixo de uma página.

A Figura 4.4 mostra a arquitetura da ferramenta EasyT. O sistema estará hospedado em um servidor WEB. Isso permite que o tutorial possa ser acessado de qualquer plataforma com conexão a internet. O tutorial é formado por um arquivo que descreve seus metadados e páginas numeradas de 1 a n. Cada página possui embutido metada-

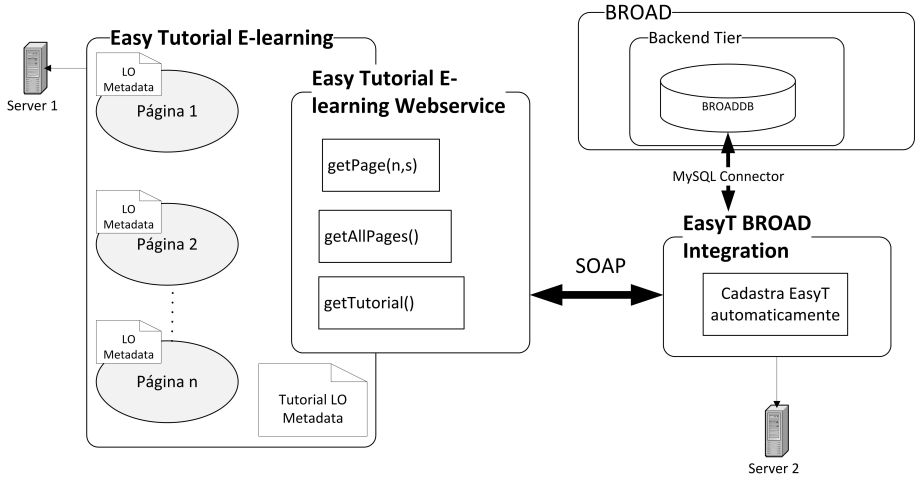

Figura 4.4: Arquitetura da ferramenta Easy Tutorial E-learning e a integração com o Projeto BROAD.

dos de OA. O container **<Easy Tutorial E-learning Webservice>** é o serviço do tutorial. Contém a interface responsável por promover a interoperabilidade do EasyT com repositórios de OAs. O *web service* possui os seguintes métodos: `getAllPages()`, `getPage(n,s)` e `getTutorial()`. O método getPage(n,s) tem como parâmetro **<n>** (número de página) e **<s>** (número de subpágina) e retorna um objeto do tipo **<Pagina>** (descrito no diagrama de classes da Figura 4.3), isto é, ele retorna o endereço de acesso a página e o seus metadados. O método `getAllPages()` retorna a coleção com todas as páginas do tutorial e seus respectivos metadados. O método `getTutorial()` retorna a coleção com todas as páginas do tutorial, seus respectivos metadados e os metadados do tutorial como todo (indicado por **<Tutorial LO Metadata>** na Figura 4.4).

A arquitetura mostra como o tutorial pode ser integrado ao repositório do BROAD. O container **<EasyT BROAD Integration>** é a aplicação responsável por promover a integração de tutoriais EasyT com o repositório do BROAD. A aplicação se comunica com o **<EasyT Webservice>** através do protocolo SOAP - Simple Object Access Protocol. Através deste protocolo é possível acessar os métodos do *web service* para retornar os dados do tutorial. O EasyT BROAD Integration se conecta à base de dados do repositório através de um conector MySQL para inserir os dados obtidos no *web service* na base de dados do repositório.

As classes do sistema foram implementadas usando a linguagem PHP. O dia-

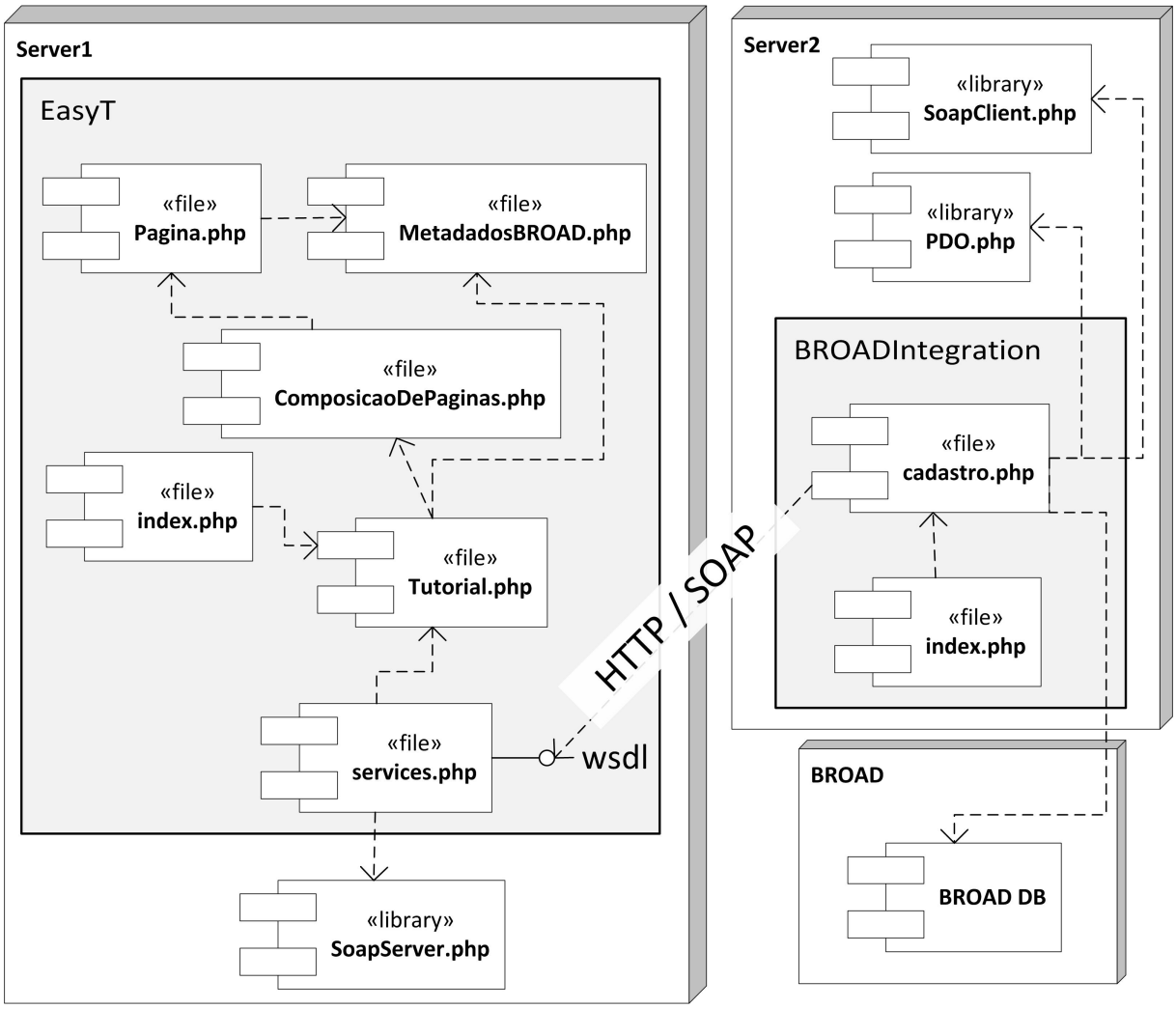

Figura 4.5: Diagrama de Componentes do Easy Tutorial E-learning e o EasyT BROAD Integration.

grama de componentes da figura 4.5 apresenta o esquema de como o sistema é implementado. No **Server 1**, o arquivo `Tutorial.php` contém a classe **<Tutorial>** . Esta classe, como mostrado no diagrama de classes da figura 4.3, conterá uma instância da classe **<ComposicaoDePaginas>** e, portanto, requer o arquivo que a contenha, `ComposicaoDePaginas.php`. Este arquivo contém o algoritmo responsável pela leitura do diretório do tutorial e pela geração das instâncias de **<Pagina>** . Este processo ocorre cada vez que uma requisição ao arquivo `index.php` no **Server 1** é realizada. Desse modo, através objeto **<Tutorial>** instanciado é possível extrair todos os recursos do sistema.

No arquivo `services.php` estão os métodos para o *web service* do tutorial. Cada método do *web service*, especificado na arquitetura da Figura 6, instancia uma das classes do tutorial. Com uso da biblioteca nativa `SoapServer.php`, a interface dos métodos contidos no arquivo `services.php` são mapeados de forma automatizada para linguagem WSDL. Dessa maneira, obtemos a interface para o *web service*, indicado por **<wsdl>** na Figura 6.

O EasyT BROAD Integration funciona como o cliente do *web service* do tutorial. Na aplicação para integração do EasyT com o BROAD, presente no nó Server2 do diagrama de componentes da Figura 6, o arquivo `index.php` contém um formulário que é processado no arquivo `cadastro.php`. O formulário contém um campo de texto, onde o usuário deve inserir a URL do tutorial. O `cadastro.php` recupera a string do campo e concatena com "/service.php?wsdl" obtendos a URL do *web service* do tutorial. De posse do *wsdl*, o arquivo `cadastro.php` inicializa a biblioteca nativa `SoapClient.php`, e então a aplicação consegue acessar os métodos do *web service* do tutorial. Como resultado, a aplicação obtém os objetos com os metadados do tutorial.

Para de fato integrar o EasyT ao BROAD, o EasyT BROAD Integration acessa o banco de dados do BROAD através da biblioteca nativa `PDO.php`. Esta biblioteca manipula a conexão entre a aplicação e o banco de dados conferindo segurança às transações. Através da linguagem SQL, os dados obtidos com os objetos recuperados do *web service* são inseridos em tabelas do banco de dados do BROAD.

Para preencher os metadados title, description, language, keyword e version do EasyT de forma compatível ao BROAD foram inseridos dados na tabela `‘broad‘.‘oa‘`.

Para inserir o metadado `author`, foram realizadas inserções na tabela `‘broad‘.‘oacontribute‘`. Para inserir o endereço do tutorial que é um metadado gerado pela própria ferramenta, foi necessário inserir dados na tabela `‘broad‘.‘oatechnical‘`.

Para formar as instâncias do tutorial, a estrutura de pastas da implementação genérica deve ser copiada para um novo diretório no servidor. Na pasta raiz deste novo diretório, o criador do tutorial deverá colocar um arquivo HTML para cada página. Dentro de cada página HTML devem estar: os metadados da página e o conteúdo específico. As páginas devem conter a nomenclatura: `paginaX-Y.html`, onde `X` corresponde ao número da página e `Y` corresponde ao número da subpágina. O criador da instância do tutorial deve nomear estes arquivos com a ordem numérica que se deseja para o índice, o qual é gerado automaticamente pela ferramenta. Dessa forma, cada vez que a instância do tutorial é acessada no navegador, o algoritmo da ferramenta lê a pasta raiz, recupera as informações de cada arquivo HTML (inclusive os metadados), forma o objeto **<Paginas>** e gera a página. A partir deste objeto o *web service* do tutorial consegue recuperar os metadados. O arquivo `MetaDados.xml`, contido no diretório raiz do tutorial, é um arquivo XML que contém os metadados do tutorial. Este arquivo é validado por ObjetoDeAprendizagem.dtd que define como deve ser a estrutura dos metadados.

Com a construção das páginas e o preenchimento correto dos metadados, o tutorial pode ser acessado através da URL:

`http://<host>/<diretorio>/ (1)`

onde, <host>é o endereço do servidor, <diretorio>é o diretório do tutorial. As páginas do tutorial podem ser acessadas individualmente através da URL:

`http://<host>/<diretorio>/index.php?num=<X>&subnum=<Y >(2)`

onde, <X>e <Y>correspondem aos números de página e subpágina, respectivamente. O *web service* do tutorial pode ser acessado através da URL:

`http://<host>/<diretorio>/services.php?wsdl (3)`

A Figura 4.6 mostra a tela do EasyT BROAD Integration. No centro da tela, encontra-se o campo de texto onde deve ser inserido a URL do tutorial no formato EasyT

que se deseja integrar ao repositório do BROAD. Ao clicar no botão "CADASTRAR" logo abaixo do campo do texto, a URL é submetida para o cadastro.php onde ocorre o processamento das informações do OA e o cadastro automático ao BROAD.

Figura 4.6: Tela do Easy Tutorial E-learning BROAD Integration.

4.3 Avaliação da solução proposta

Um estudo experimental foi realizado com o objetivo de avaliar a solução proposta. O experimento procurou analisar a eficácia do EasyT em relação ao apoio à construção de OAs para uma LPS. Neste caso, o OA está relacionado a um tutorial hipermídia para a linha de produtos do BROAD-PL.

A definição de uma pesquisa é o fundamento para o estudo experimental, ou seja, enquanto o planejamento da pesquisa determina como o estudo será conduzido, a definição da pesquisa indica o porquê do experimento (Medeiros et al., 2010).

Desse modo, a questão que define a pesquisa é:

> "Qual a extensão na qual a ferramenta apóia a construção de OAs no contexto do BROAD-PL?"

Esta questão nos leva à seguinte hipótese:

> "Se oferecermos uma ferramenta baseada no paradigma de Orientação a Serviços, então a construção e a integração de OAs no contexto do BROAD-PL poderão ser potencializadas."

Consequentemente, o experimento deve verificar a extensão na qual os princípios da Orientação a Serviços podem apoiar a construção do OA, isto é, a facilidade e eficácia da instanciação do tutorial. Também deve verificar se após a construção do OA, ocorre a integração automática ao ambiente do BROAD-PL. Integrar o OA ao ambiente significa que este poderá ser recuperado através dos mecanismos de busca presentes no repositório do BROAD-PL.

Para tanto, tutoriais hipermídias foram construídos utilizando o template do EasyT e estes foram integrados ao repositório do BROAD-PL. Foram especificados três conteúdos para os quais os tutoriais deveriam ser construídos:

- No domínio de Parasitologia: "Influenza Virus A: H1N1" (Figura 4.7)
- No domínio de Engenharia de Software: "Requisitos de Software: Especificação de Requisitos"
- No domínio de Engenharia de Software: "Requisitos de Software: Requisitos Não Funcionais" (Figura 4.8)

O conteúdo específico de cada domínio já estava disponibilizado em páginas na internet. Além disso, os tutoriais deveriam conter entre cinco e dez páginas.

Dessa forma, o experimento foi conduzido no laboratório do Núcleo de Engenharia do Conhecimento (NEnC) da Universidade Federal de Juiz de Fora por dois alunos de graduação em Ciência da Computação. Os alunos possuíam pouca experiência com o uso de HTML e nenhuma experiência com programação em PHP. Antes do processo de construção, eles foram submetidos a um treinamento com duração de uma hora. Junto ao treinamento, foi entregue uma documentação objetiva sobre o processo de instanciação.

O aluno 1 construiu um tutorial para o domínio de Parasitologia: "Influenza Virus A: H1N1", e o aluno 2 construiu dois tutoriais no domínio de Engenharia de Software: "Requisitos de Software: Especificação de Requisitos" e "Requisitos de Software: Requisitos Não Funcionais".

Após a construção dos tutoriais, foi realizada uma entrevista informal com cada aluno sobre o processo de construção e integração. Os tutoriais construídos foram submetidos à avaliação. Algumas observações em relação à construção foram apontadas:

(i) Os alunos gastaram cerca de uma hora para instanciar o tutorial. É um tempo relativamente curto se comparado ao tempo de construção de um OA que possua os mesmo recursos do EasyT sem o uso de um template.

(ii) A maior parte do tempo da construção foi gasta na formatação dos documentos HTML e foram cometidas incoerências relacionadas à formatação do HTML e o preenchimento dos metadados das páginas. Isso demonstra que o autor de uma instância do tutorial deve ter conhecimentos de HTML. Esta questão pode ser resolvida com a autoria de ferramentas que facilitem a edição e a manipulação dos arquivos HTML.

(iii) Para todos os alunos, o processo de renomeação das páginas se mostrou um pouco trabalhoso. É necessário, portanto, que haja um mecanismo para agilizar o processo de renomeação dos arquivos que é mandatório para a ordenação dos links no índice do tutorial.

(iv) Não foram necessários conhecimentos de linguagem PHP para a construção. Isso é um ponto positivo para o EasyT, já que a maior parte dos recursos fornecidos pelo template utilizam a linguagem PHP.

(v) Não foram necessários conhecimentos sobre a tecnologia de *web service*s para a integração do tutorial ao BROAD-PL. Isto também é um ponto positivo, pois o mecanismo que EasyT utiliza a tecnologia de webservices, entretanto, ela se mostrou transparente ao usuário.

(vi) Após a integração ao BROAD-PL, foi possível recuperar o tutorial através do mecanismo do repositório. Além do tutorial, cada página também foi integrada ao mecanismo de busca do BROAD-PL.

Através da observação (vi) foi possível identificar alguns problemas relacionados ao aspecto do processo de integração. Na maioria dos tutoriais criados, todas as páginas

tinham um mesmo autor, isto é, o metadado author possuía o mesmo valor para todas as páginas. Ocorria que, cada vez que os metadados de uma página do tutorial eram inseridos no banco de dados do BROAD-PL, os dados de um mesmo autor eram inseridos várias vezes na tabela 'broad'.'oacontribute' gerando uma redundância desnecessária. Esta visão causa redundância tanto no mecanismo de integração do EasyT quanto no mecanismo de busca do BROAD. Uma possível solução para o EasyT seria que a aplicação de integração realizasse a varredura dos autores de todas as páginas antes de inseri-las procurando por registros repetidos. As repetições não devem ser inseridas. Assim, o registro de cada página deve fazer referência ao único registro do autor.

Além disso, outra questão diz respeito a uma página possuir um conteúdo autossuficiente para se comportar como um OA autônomo no BROAD-PL. Como o mecanismo de integração do EasyT não faz esta distinção, deveria haver uma maneira de informar ao mecanismo de integração quais páginas não devem ser tratadas como um OA autônomo.

A seguir são apresentadas algumas telas dos tutoriais construídos pelos alunos (Figuras 4.7 e 4.8).

A figura 4.7 mostra a tela inicial do tutorial "Influenza Virus A: H1N1", enquanto Figura 4.8 mostra uma página do tutorial "Requisitos de Software: Requisitos Não Funcionais". Essas figuras ilustram os dois estados genéricos do sistema: o estado de tela inicial e o estado de página ativada. No estado de tela inicial, nenhuma página está ativada.

Apenas são apresentados os metadados "Descrição", "Autor" e "Idioma" que estão descritos no `MetaDados.xml`. No estado de página ativada, o conteúdo de página aparecerá na parte central da tela. Na Figura 4.8, a página "1.5 - Usabilidade" encontra-se ativada e o link que corresponde à página corrente aparece destacado no índice.

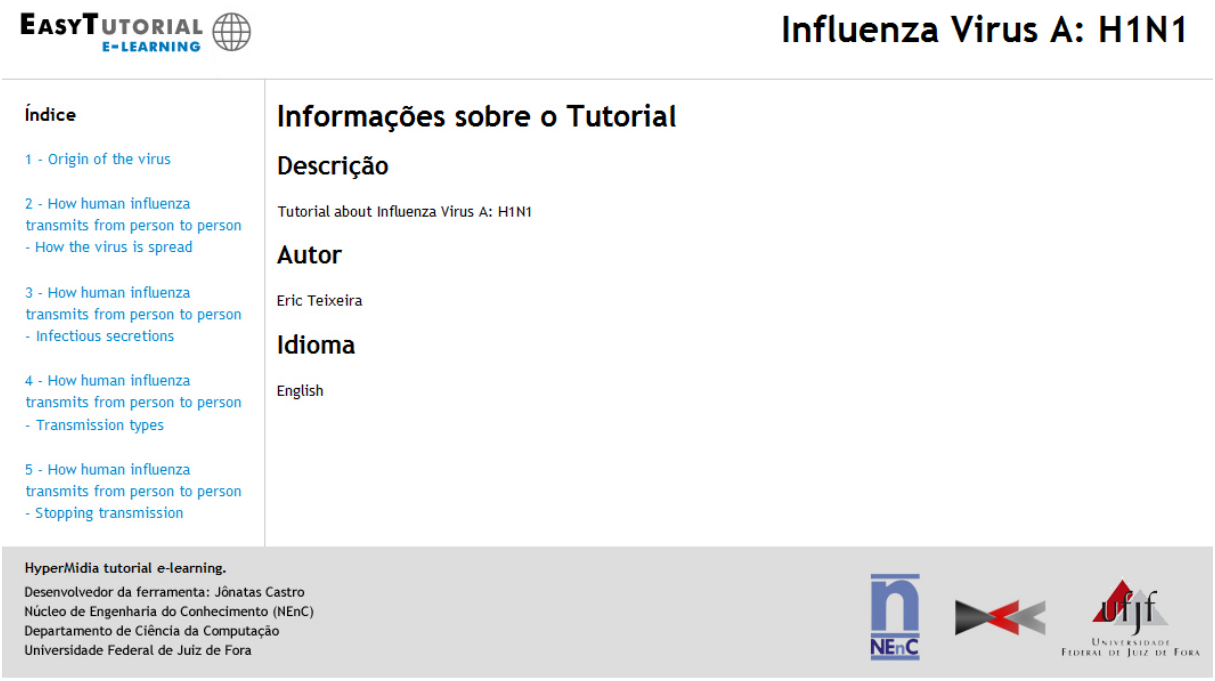

Figura 4.7: Tela inicial do Easy Tutorial E-learning - "Influenza Virus A: H1N1".

Figura 4.8: "Página 1.5 - Usabilidade" do Easy Tutorial E-learning - "Requisitos de Software: Requisitos Não Funcionais".

Como pode ser visto nas Figuras 4.7 e 4.8 o tutorial apresenta na interface cabeçalhos e rodapés fixos. Na parte esquerda do cabeçalho está o logotipo da ferramenta e na parte direita está o título que foi especificado no metadado do tutorial. Na Figura 4.7, o cabeçalho mostra o título "Influenza Virus A: H1N1". Este título está descrito no arquivo MetaDados.xml, contido na estrutura genérica da ferramenta. No lado esquerdo da tela está o índice que é gerado automaticamente. Cada item do índice corresponde ao link para uma página do tutorial. Antes do título de cada item, são indicados os números de página e subpágina.

5 Conclusões

Este trabalho apresentou uma solução para a criação semiautomática de OAs considerando-se o contexto de uma Linha de Produtos de Software. Para tanto, foi especificada, projetada e implementada uma ferramenta para apoiar a construção de OAs. Estes artefatos estão inseridos no contexto de uma Infraestrutura Baseada em Ontologias, Workflow e Objetos de Aprendizagem para a Composição de Conteúdos Educacionais.

A contribuição principal deste trabalho está no mecanismo de interoperabilidade presente no EasyT, isto é, o encapsulamento do OA como serviço. O mecanismo possibilita a integração automática do tutorial a repositórios de uma linha de produtos. Isto representa um avanço para a implantação da LPS de OAs porque a gerência da complexidade e variabilidade, inerente à LPS, necessita de mecanismos de suporte automatizados. Com o uso de *web service*s, foi possível recuperar os metadados do OA de forma automática e então, automatizar o processo de integração do OA com repositórios considerando os princípios da Orientação a Serviços.

Entretanto, a construção de OAs pode abranger diversos domínios de conhecimento, portanto é necessário analisar as necessidades de cada domínio bem como as tecnologias envolvidas na construçao do OA. Por exemplo, pode-se ter um OA na área de Matemática em que é necessário utilizar notações especiais para equações. Para este cenário, seria necessário disponibilizar um mecanismo eficiente para escrever as equações, além de estabelecer um padrão para as notações utilizadas. A ferramenta proposta não oferece este tipo de suporte, porém pode ser expandida no intuito de tratar outras tecnologias além do HTML. Outra situação, por exemplo, é um OA para a área de Ciência da Computação que necessita de animações para algoritmos de ordenação, bem como um ambiente para testá-los. Neste caso, o autor do OA deveria ter conhecimentos sobre uma ferramenta para produzir animações e sobre como implementar um ambiente de teste integrado ao OA. Isto também necessita ser tratado no contexto da ferramenta EasyT.

O EasyT trata o processo de composição do conteúdo, porém não lida com os elementos de contexto do OA durante a sua criação. Diferente de outras abordagens,

ele é um template independente do contexto pedagógico. Esta característica favorece a reutilização do OA porque evita o acoplamento ocasionado pelas relações dos elementos de contexto presentes em um domínio. Entretanto, o conteúdo e o contexto de um OA podem ser tratados em uma etapa posterior ao considerarmos as facilidades oferecidas pela ferramenta em relação à reutilização, composição, autonomia, ausência de estado, abstração e baixo acoplamento.

Outra questão diz respeito à granularidade do tutorial. No momento de integração do EasyT ao BROAD, cada página do tutorial é inserida como um OA distinto. Isso se torna um problema quando o conteúdo de uma página não é suficiente para sobreviver fora de um contexto. Portanto, faz-se necessário identificar o nível de granularidade de cada página para saber se vale à pena inseri-la ao repositório de OA distintamente.

O EasyT é um template de OA. Para a produção de OAGs é necessária uma ferramenta para a customização de templates e ferramentas para a geração de OAs. Ferramentas como essa facilitariam o processo de instanciação do tutorial já que o autor não precisaria tanto de conhecimentos de HTML. Isto não faz parte do escopo desta pesquisa.

Vale ressaltar também a necessidade de um suporte mais amplo à interoperabilidade entre OAs, considerando-se, por exemplo aspectos relacionados à semântica dos dados que serão trocados entre os objetos. Essas lacunas deverão ser resolvidas posteriormente.

Referências Bibliográficas

Braga, R.; Campos, F.; Santos, N.; Souza, A. C.; Mattos, E. ; Nery, T. **Applying semantics for the retrieval of learning objects on the web**. In: 4th Brazilian Workshop on Semantic Web and Education in SBIE 2011, p. 2374–2383, Aracaju, 2011.

Burbaite, R.; Stuikys, V. **Analysis of learning object research using feature-based models**. In: Rimantas, B.; Butkiene, R., editors, Proceedings of the 17th International Conference on Information and Software Technologies, Information Technologies, p. 201–208, Kauanas, Lithuania, 2011. Kauanas University of Technology.

Campos, F.; Braga, R.; Souza, A. C.; Santos, N.; Matos, E. ; Nery, T. **Projeto broad: Busca semântica por objetos de aprendizagem**. In: VIII Congresso Brasileiro de Ensino Superior a Distância - ESUD 2011, Ouro Preto, MG, 2011.

Clements, P.; Northrop, L. **Software product lines: practices and patterns**. Boston: Addison-Wesley, 2001.

Damasevicius, R.; Stuikys, V. **Specification and generation of learning object sequences for e-learning using sequence feature diagrams and metaprogramming techniques**. In: The 9th IEEE International Conference on Advanced Learning Technologies, ICALT 2009, July 15-17, 2009, Riga, Latvia, p. 572–576. IEEE, 2009.

Erl, T. **SOA: principles of service design**. Pearson Prentice Hall, 2009.

Gimenez, I. M. S.; Travassos, G. H. **O enfoque de linha de produto para desenvolvimento de software**. In: XXII Congresso da Sociedade Brasileira de Computação - Jornada de Atualização em Informática. SBC, 2002.

Gracindo, H. B. R. **Objetos digitais de aprendizagem: uma ferramenta para o ensino**. Maceió, 2009. Dissertação de Mestrado - UFAL.

Griss, M. L. **Implementing product-line features with component reuse**. In: Proc. 6th International Conference on Software Reuse, p. 137–152, Jun. 2000.

LTSC, L. T. S. C. **Wg12 - learning object metadata (lom)**. http://ltsc.ieee.org/wg12/, 2000. acessado em 17/08/2012.

Konrath, M.; Tarouco, L.; Carvalho, M. K. ; Avila, B. **Formação de professors para produção de uso de objetos de aprendizagem**. In: Revista Novas Tecnologias na Educação RENOTE, volume 4, p. 1–10, Jun. 2006.

Kueger, C. W. **Implementing product-line features with component reuse**. In: Communications of the ACM, volume 49, p. 37–40, Jun. 2006.

Lazilha, F. R. **Uma proposta de arquitetura de linha de produto para sistemas de gerenciamento de workflow**. 2002. Dissertação de Mestrado - Universidade Federal do Rio Grande do Sul.

Lee, J.; Muthig, D. **Feature-oriented variability management in product line engineering**. In: Communications of the ACM, volume 12, p. 55–59, 2012.

McGregor, J. D.; Northrop, L. M.; Jarrad, S. ; Pohl, K. **Guest editors' introduction: Initiating software product lines**. volume 19, p. 24–27, Los Alamitos, CA, USA, Jul 2002. IEEE Computer Society Press.

Medeiros, F. M.; de Almeida, E. S. ; de Lemos Meira, S. R. **Designing a set of service-oriented systems as a software product line**. In: Fourth Brazilian Symposium on Software Components, Architectures and Reuse, p. 70–79, 2010.

Morales, R.; Leeder, D. ; Boyle, T. **A case in the design of generative learning objects: Applied statistical methods**. In: Kommers, P.; Richards, G., editors, Proceedings of World Conference on Educational Multimedia, p. 2091–2097, 2005.

Najjar, J.; Duval, E.; Ternier, S. ; Neven, F. **Towards interoperable learning object repositories: The ariadne experience**. In: Proc. IADIS Int'l Conf. on WWW/Internet 2003, p. 219–226, 2003.

Ribeiro, H. B. G.; de Lemos Meira, R.; de Almeida, E. S.; Lucrédio, D.; Álvaro, A.; Alves, V. ; Garcia, V. C. **An assessment on technologies for implementing core assets in service-oriented product lines**. In: Fourth Brazilian Symposium on Software Components, Architectures and Reuse, p. 90–99, 2010.

McGregor, J. D.; Northrop, L. M. ; P. C. Clements, L. G. J. **A framework for software product line practice**. http://www.sei.cmu.edu/productlines/start/index.cfm, 2011. acessado em 25/04/2012.

Sayao, L. F.; Marcondesm, C. H. **O desafio da interoperabilidade e as novas perspectivas para as bibliotecas digitais**. In: TransInformação, volume 20, p. 133–148, Campinas, 2008.

da Silva, A. P. **Uma linha de produto de software baseada na web semântica para sistemas tutores inteligentes**. 2011. Dissertação de Mestrado - Universidade Federal de Campina Grande.

Sommerville, I. **Engenharia de Software**. São Paulo: Pearson Prentice Hall, 2011, 355–368p.

Stuikys, V.; Damasevicius, R. **Development of generative learning objects using feature diagrams and generative techniques**. volume 7, p. 277–288, Vilnius, 2008. Institute of Mathematics and Informatics.

Turner, M.; Budgen, D. ; Brereton, P. **Turning software into a service**. In: Computer, volume 36, p. 2374–2383, Oct. 2003.

Vieira, C. E. M. **Desenvolvimento de objetos de aprendizagem, baseado em especificações de normatização scorm, para o caso de suporte Ã aprendizagem de funções**. In: IX Ciclo de Palestras sobre Novas Tecnologias na Educação, p. 1–10, 2007.

Wiley, D. A. **Learning object design and sequencing theory**. June 2000. 1–142p. Tese de Doutorado - Brigham Young University.