Adaptação de Conteúdo SCORM em Ambientes Inteligentes de Aprendizagem

Adilson Vahldick[1, 2], André L.A. Raabe[1]

[1]Mestrado em Computação Aplicada – Grupo de Informática na Educação
Universidade do Vale do Itajaí (UNIVALI) – São José, SC – Brasil

[2]Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC

adilsonv@furb.br, raabe@univali.br

Abstract. *This paper presents a way for the use of didactic material in the SCORM standard by Intelligent Learning Environments (ILE). The content SCORM has an internal structure of sequencing and navigation that was not designed to be adapted by intelligent environments as the needs of students. In order to meet this need, this paper presents a component that enables environments perform and adapt courses in SCORM standard. The adaptation happens by generating a new course joining other courses, total or partially, or changing the rules of sequencing and navigation. The component was developed in Java for web applications.*

Keywords: *SCORM; Intelligent learning environments; Learning objects; Adaptability*

Resumo. *Este trabalho apresenta a viabilidade de utilizar material didático no padrão SCORM em Ambientes Inteligentes de Aprendizagem (AIA). O padrão SCORM possui uma estrutura interna de seqüenciamento e navegação e não consegue ser adaptado segundo as preferências ou estado cognitivo do aluno. Com intuito de suprir essa necessidade, esse artigo apresenta um componente que oferece a infra-estrutura para os ambientes executarem e adaptarem cursos no padrão SCORM. A adaptação ocorre com a geração de um novo curso unindo outros cursos, seja total ou parcialmente, assim como a alteração das regras de seqüenciamento e navegação. O componente foi desenvolvido em Java para aplicações Web.*

Palavras-chave: *SCORM; Ambientes inteligentes de aprendizagem; Objetos de aprendizagem; Adaptabilidade*

1. Introdução

Um problema inicial na operacionalização de Ambientes Inteligentes de Aprendizagem (AIA), como os Sistemas Tutores Inteligentes (STI) ou Sistemas de Hipermídia Adaptativa (SHA), é a disponibilização do material instrucional. O que é comum encontrar na literatura, é que os conteúdos apresentados por esses sistemas são produzidos neles e para eles, restringindo a reutilização desse material de e para outros

sistemas. E muitas vezes, o próprio desenvolvedor desses ambientes é o responsável por manter o conteúdo.

Em contrapartida, com a popularização do e-learning, materiais didáticos passaram a ser produzidos em larga escala e impulsionaram a organização destes em unidades reusáveis que foram denominadas Objetos de Aprendizagem (OA). Com a crescente demanda na utilização desses tipos de recursos, houve a necessidade de estipular uma padronização para a identificação e utilização desses objetos. Nesse sentido, entidades como IEEE, IMS e ADL, produziram especificações e padrões para suprir essas necessidades. A ADL agrupou especificações e produziu o modelo SCORM (Sharable Content Object Reference Model), com intuito de estabelecer um padrão para utilização de material em ambientes Web (ADL, 2006).

Um material no formato SCORM contém uma estrutura e regras de apresentação desse conteúdo. Ele pode conter qualquer arquivo que possa ser exibido em um navegador, bem como atividades interativas, exercícios e questionários. Existem ferramentas de autoria para que professores possam produzir material nesse formato, e ambientes que possam executá-los. Uma vez construído o arquivo, ele pode ser executado por qualquer ambiente que esteja em conformidade com o padrão.

Nesses ambientes toda a interação do aluno com o material é registrada e de acordo com o seu desempenho nas atividades, novos assuntos vão sendo disponibilizados a ele. O foco do padrão SCORM é o auto-aprendizado, ou seja, o aluno interage exclusivamente com o material, sem interferência de outros atores no seqüenciamento. A ordem do que será ensinado já vem estabelecida no próprio arquivo.

Com isso, surgem dificuldades para incorporar tal material em um ambiente de aprendizado, pois o conteúdo abrangido pode estar num nível de granularidade não desejável para o ensino de um conceito, sendo muitas vezes o suficiente uma pequena parte desse material. Ainda, não se deseja em AIA que as regras definidas para avançar no conteúdo sejam as mesmas para todo tipo de aluno. Conforme Fischer (2001), uma das propriedades natas de AIA, e o que dá o atributo de "Inteligência" a esses sistemas, é a adaptação do que será ensinado e como será apresentado de acordo com as preferências e características de cada aluno. Um pacote completo SCORM é apresentado em uma estrutura que não contempla as particularidades de um aluno.

Com intuito de flexibilizar a introdução de novos materiais instrucionais que sigam o formato SCORM de forma que estes possam ser adaptáveis às necessidades do aluno, apresenta-se nesse artigo uma infra-estrutura, em forma de componente, para que os AIA baseados na Web desenvolvidos em Java possam utilizar, e interferir na estrutura destes materiais.

O artigo está organizado da seguinte forma: na seção 2 serão analisadas as propostas de outros autores para o problema. Na seção 3 são discutidas as funcionalidades e arquitetura do componente desenvolvido. Na seção 4 são descritos resultados de dois ambientes que demonstraram a validade e funcionalidade do componente. Na seção 5 são apresentadas as conclusões da utilização dessa infra-estrutura.

2. Trabalhos Correlatos

Segundo Brusilovsky (2001), para os STIs com material instrucional incorporado dinamicamente à sua base de conhecimento, a tendência é que os desenvolvedores desses sistemas se baseiem no padrão Learning Object Metadata (LOM). O LOM (IEEE LTSC, 2002) é um padrão de metadados que possui elementos para a identificação de conteúdo instrucional. Um exemplo de STI que usa essa estratégia é o ABITS (Capuano, Marsella e Salerno, 2000). Ao adicionar OA – que pode ser em qualquer formato, desde que renderizável num navegador Web – o instrutor precisa definir os metadados desse objeto. A geração do currículo é feita com esses metadados, baseando-se no mapa conceitual do modelo de domínio, em conjunto com o estado cognitivo e preferências do usuário que constam no modelo do estudante. Com esses dados, é feita uma busca no repositório de dados procurando nos metadados os valores dos atributos "formato", "proposta pedagógica", "nível de interatividade" e de "dificuldade".

De forma similar, os trabalhos de Hoermann *et al* (2003) e Bhatt e Rao (2006) propõem formas de se utilizar o LOM na identificação dos objetos para posterior criação do curso. Hoermann *et al* (2003) definem uma forma de estruturar o modelo de domínio, através de uma semântica para o mapa conceitual e o relacionamento com as mídias (que são os OAs). É possível adicionar novos objetos sem diretamente estar relacionado a um conceito, mas criar uma associação com outros objetos, já que o LOM contém um campo para esse propósito. A semântica definida no mapa conceitual permite que se crie o curso como uma estrutura de árvore, onde as folhas são os OAs. O trabalho de Bhatt e Rao (2006) apresenta um guia de como identificar OAs no LOM utilizando-se da taxonomia de Bloom-Vincenti e como um STI pode realizar inferências a partir dessas informações para decidir o próximo objeto de aprendizagem a ser apresentado.

Também existem propostas de estender o SCORM para prover a adaptabilidade do conteúdo, como em Silva *et al* (2006), Shin, Lee e Baik (2007) e Rey-Lopez *et al* (2008). Silva *et al* (2006) e Shin, Lee e Baik (2007) propõem alterações no arquivo de manifesto[1]: o primeiro para direcionar a seqüência das atividades segundo o estilo de aprendizado do aluno; e o segundo para relacionar o nível de utilidade da atividade em relação ao aprendizado de um conceito.

No artigo de Rey-Lopez *et al* (2008) é apresentada a adaptabilidade em dois níveis: de SCO[2] e de atividades. No nível de SCO, introduziram novos elementos no Data Model[3] para que os SCOs possam solicitar ao LMS informações sobre as características do usuário, e assim montar as páginas de acordo com essas informações. No nível de atividades, foram criadas novas regras de seqüenciamento para que a estrutura apresentada para o aluno também dependa de seu estado cognitivo e suas preferências. Para a montagem de cursos com essa extensão do SCORM, foi

[1] Arquivo de manifesto é um documento XML contido no pacote SCORM que descreve a estrutura das atividades e as regras de seqüenciamento do curso.

[2] SCO (Sharable Content Object) representa uma unidade indivisível de um conjunto de recursos, e que controla o início e o fim de uma atividade. Um curso é composto de uma estrutura em árvore de SCOs.

[3] Data Model são identificadores de variáveis que os SCOs utilizam para se comunicar com o LMS.

desenvolvida uma ferramenta de autoria própria, e um plug-in para a ferramenta Reload Editor.

3. Componente desenvolvido

O componente apresentado nesse artigo, batizado de CELINE, foi desenvolvido com tecnologias Java para aplicações web. Os seus objetivos principais são permitir que sejam executados pacotes SCORM, assim como interferir durante a interação do estudante com esses pacotes. O componente, conectado à aplicação web, intermedia vários recursos necessários para atender a esses objetivos.

O componente é um artefato de software facilmente acoplado à aplicação (Barroca *et al*, 2005; McClure, 2001), que contém um conjunto de soluções completas para um problema. Um componente não é desenvolvido para ser extensível, mas contém interfaces para a personalização de suas funções. Uma aplicação usa o componente.

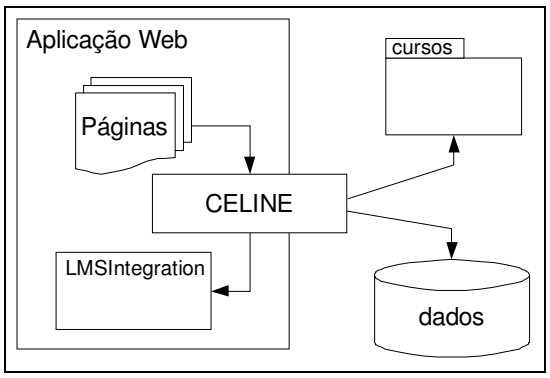

Figura 1 – Visão de alto nível da arquitetura do componente ligado à aplicação

A Figura 1 apresenta a arquitetura de alto nível do componente com a aplicação. De forma resumida, dentro da aplicação existem páginas (JSP ou JSF) que podem acessar os recursos do CELINE. LMSIntegration representa a classe que o CELINE se comunica para adaptar os cursos. A pasta cursos é onde são descompactados os pacotes SCORM, e o componente necessita de uma área de armazenamento para guardar as informações das interações dos usuários com os cursos. Nos próximos parágrafos essa arquitetura será detalhada.

Para dar suporte ao objetivo de executar os pacotes SCORM, são necessários recursos para gerenciar esses pacotes e os usuários. Nesse sentido foram disponibilizadas *tags* para que o desenvolvedor possa implementar páginas (JSP ou JSF) na sua aplicação que façam acesso a esses recursos. Existem tags para as seguintes tarefas:

- **Importar cursos**: monta um formulário contendo um campo de seleção de arquivos para *upload*. O formulário é submetido ao componente que descompacta o curso em uma pasta (apresentada na Figura 1 como "cursos") e inclui o curso na área de dados. Aqui cursos são usados como sinônimos de conteúdo SCORM;

- **Gerenciar usuários:** listar usuários e montar um formulário com os campos de nome, senha e tipo de usuário. O componente inclui, altera ou exclui o usuário, de acordo com as opções vindas do formulário;
- **Registro e cancelamento do registro de cursos:** o usuário precisa estar registrado para abrir um curso. Duas *tags* distintas criam links para o registro e o cancelamento do registro;
- **Listar cursos:** fornece a lista de cursos incluídos, e inclusive fornece se o curso está registrado ao usuário corrente. Essa lista oferece links para abrir o curso. Quando o usuário abre um curso, o componente apresenta uma página dividida em duas partes: na esquerda é apresentada uma estrutura de árvore com todos os itens do curso, e à direita é apresentado o conteúdo da atividade;
- **Visualizar a árvore de atividades:** disponibiliza a lista de atividades contidas em um conteúdo SCORM. Ao desenvolvedor fica disponível o título e o nível da atividade. Assim ele pode implementar uma estrutura visual de árvore. Esse recurso é utilizado para consulta dos títulos das atividades e não o conteúdo delas;
- **Encerramento e suspensão da interação:** são utilizadas durante a interação do aluno com o conteúdo. O ato de suspender a interação garante que o usuário continue um curso em outro instante, e o encerramento implica em recomeçar nesse outro instante;
- **Autenticar usuários:** é a tradicional página de *login*. O acesso a qualquer dos recursos acima exige que o usuário esteja autenticado no componente.

Para a adaptação do conteúdo a ser apresentado ao aluno, é preciso implementar uma classe que siga uma interface definida no componente. Essa classe está representada na Figura 1 como LMSIntegration. Brusilovsky e Vassileva (2003 apud Karampiperis e Sampson, 2006) citam dois momentos de gerar o seqüenciamento adaptável do conteúdo: baseado no conhecimento inicial do aluno (geração adaptativa de conteúdo) e durante a interação do aluno com o curso (geração dinâmica de conteúdo).

Seguindo essas idéias, a classe LMSIntegration precisa implementar os seguintes métodos:

- **Listar cursos:** fornece a lista de cursos disponíveis ao usuário. Ao método é entregue uma lista de todos os cursos registrados pelo usuário, porém, ela pode ser alterada para que apresente, por exemplo, somente aqueles que o usuário esteja apto a executar segundo o desempenho das atividades anteriores;
- **Abrir um curso:** esse método é executado antes de abrir um curso, e corresponde à geração adaptativa de conteúdo. Através dele, o ambiente monta um curso juntando partes de outros cursos. As atividades podem ser desabilitadas ou escondidas, para que em outro momento possam estar disponíveis ao aluno;
- **Alterar a atual árvore de atividades:** esse método corresponde à geração dinâmica de conteúdo. Esse método é chamado toda vez que o usuário encerra

uma atividade (SCO). O ambiente pode avaliar o desempenho do aluno, e habilitar ou apresentar novas atividades previstas quando da abertura do curso;

- **cmi.interactions alterado ou solicitado:** esse é o Data Model sugerido pelo modelo SCORM para manter dados específicos do curso. Quando um curso precisa guardar e obter informações específicas, por exemplo, as respostas de um questionário, o Data Model *cmi.interactions* é o identificador aconselhado para isso.

Para auxiliar na montagem do curso e na atualização da árvore de atividades, o CELINE fornece duas classes: EasyContentPackage e EasyActivityTree. Ambas possuem métodos que abstraem o entendimento do desenvolvedor em relação à especificação SCORM.

Através da classe EasyContentPackage pode-se adicionar pacotes SCORM, ou partes deles, e gerar um novo curso baseado nos pacotes previamente adicionados. Essa classe é utilizada para geração de um curso quando o usuário entra.

Por outro lado, a classe EasyActivityTree é utilizada durante a interação do usuário com o curso gerado. Por exemplo, ela possui métodos que retornam a quantidade de tentativas, ou acessos, de cada atividade, permite que atividades apareçam ou sejam escondidas durante a interação. Ela também consegue alterar regras de seqüenciamento e navegação, por exemplo, supondo que uma atividade seja visível somente quando um conjunto de atividades totalizarem 80% de sucesso, o sistema pode alterar esse peso, ou adicionar novas atividades para considerar esse peso.

Na área de dados o CELINE armazena os usuários e cursos cadastrados. Também são guardados os dados das atividades executadas, tanto para que a aplicação possa exibir o desempenho do aluno, quanto para que ele continue a interação em outro momento. O CELINE consegue armazenar esses dados em arquivos XML, em um banco de dados relacional (BDR) ou em outro mecanismo personalizado de persistência. No caso de arquivos XML, os dados dos usuários, cursos e cursos registrados do usuário são armazenados em um arquivo, e um arquivo separado para cada interação de um curso para um usuário. Quando usa BDR, devem ser especificados o driver JDBC, endereço do servidor e do esquema, e nome do usuário e senha para conexão. Nesse caso o componente necessita das tabelas apresentadas no modelo entidade-relacionamento da Figura 2. A terceira alternativa é a implementação de uma classe que siga uma interface definida no componente.

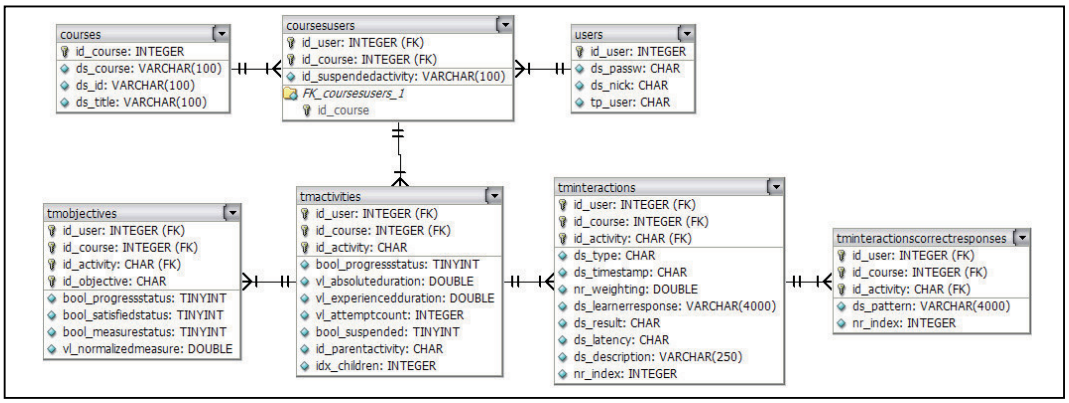

Figura 2 – Modelo entidade-relacionamento para manutenção do CELINE

O tipo de acesso é determinado em um arquivo de configurações do componente. No mesmo arquivo também são configuradas a pasta onde serão descompactados os cursos e a localização da classe de integração (LMSIntegration).

Um servlet no componente é o responsável por intermediar as interações do usuário nas páginas da aplicação. O componente é fornecido como dois arquivos JAR e sua conexão com a aplicação é feita no arquivo *web.xml*, mapeando a URL "lms" com o servlet do componente.

O arquivo de configurações do componente se chama *celine-config.xml* e deve ficar na mesma pasta do arquivo *web.xml*. A Figura 3 apresenta um exemplo desse arquivo onde está configurado "courses" como a pasta para armazenar os pacotes SCORM; a classe LMSIntegrationExample como a classe de adaptação; a página "error.jsp" para ser exibida quando acontece um erro no componente; e para usar arquivos XML como mecanismo de persistência.

```
<config>
   <courses-folder>courses</courses-folder>
   <lmsIntegration>br.univali.bruce.integration.LMSIntegrationExample</lmsIntegration>
   <error-page>error.jsp</error-page>
   <database-source>
       <xml>WEB-INF/dados.xml</xml>
   </database-source>
</config>
```

Figura 3 – Exemplo de arquivo de configuração do CELINE

4. Resultados da utilização do CELINE

Essa seção apresenta duas aplicações usadas para teste e validação dos recursos do CELINE. A primeira aplicação, batizada de BRUCE (*Basic Reference of Using the CELINE Component*), serve como uma implementação de referência na utilização do CELINE, e é um exemplo de um LMS sem recursos de adaptação: nele é possível incluir novos cursos e novos usuários, e esses usuários podem se registrar e abrir todos os cursos disponíveis no ambiente. A segunda aplicação é um STI resultado do trabalho de conclusão de curso de Schappo (2008). Essa aplicação validou o recurso do CELINE para a geração adaptativa de conteúdo, ou seja, monta o curso quando ele é iniciado.

O BRUCE foi montado contendo um servlet que simplesmente verifica se o usuário está autenticado (utiliza uma classe do CELINE): se negativo direciona para uma página de *login*, senão apresenta a página solicitada. Essa foi a única classe que necessitou ser implementada. As páginas JSP direcionam os links e formulários para o servlet, que por sua vez verifica a autenticação e direciona para a página responsável pela apresentação. As páginas não contêm *scriptlets* e utilizam as *tags* do CELINE.

A primeira página da aplicação contém um formulário montado por uma *tag* do CELINE, que mostra dois campos: um para digitar o nome e outro a senha. Ao submeter o formulário, e a autenticação ser validada pelo componente, é apresentada a página central que contém cinco links: para uma página que lista os cursos registrados para o usuário, e que depois ele pode abrí-los (a Figura 4 apresenta a aparência de um curso aberto com o CELINE: na lateral esquerda está a árvore de atividades e à direita o conteúdo da atividade selecionada); outro que apresenta todos os cursos cadastrados, onde cada um possui um link para que o usuário registre ou cancele o registro do curso, e outro para apresentar o conteúdo do curso em forma de árvore. Os outros três links são

apresentados somente se o usuário for um administrador (essa informação também é obtida do CELINE): um para administração de usuários e outros dois para administração de cursos.

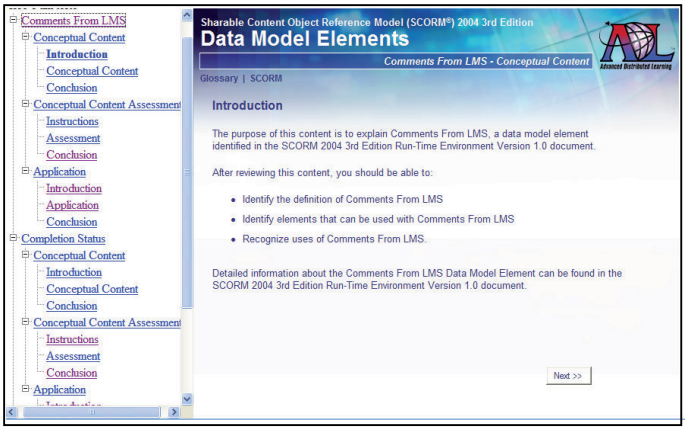

Figura 4 – Curso aberto através do componente CELINE

O BRUCE foi testado tanto com o mecanismo de persistência padrão de XML, quanto com o banco de dados MySQL, e executado no servidor Apache Tomcat.

O foco no desenvolvimento do STI de Schappo (2008) foi na montagem do curso. Não foram montadas as páginas de administração como demonstradas no BRUCE. Os dados foram armazenados num banco de dados PostgreSQL seguindo o MER apresentado na Figura 2.

A lógica fuzzy foi utilizada como a técnica de IA para o STI, tanto no modelo de domínio, como de aluno e pedagógico. O modelo de domínio do STI é composto por um grafo direcionado representando o mapa conceitual. Os vértices são os conceitos, e as arestas representam as pré-condições para que o conceito seguinte seja apresentado ao aluno, baseado nos valores: *conhece pouco*, *conhece*, *conhece bem*, *aprendeu* ou *aprendeu bem*. O conteúdo a ser ensinado corresponde a um conjunto de pacotes SCORM relacionados a cada conceito do mapa. Cada curso pode ser categorizado como de **conteúdo** ou de **exercício**. O exercício pode estar entre três níveis de dificuldade: fácil, médio ou difícil. O modelo do aluno é uma instância do mapa conceitual, porém, os vértices contêm o nível de conhecimento do aluno sobre aquele conceito. Inicialmente todos os vértices começam com o valor *desconhece*. O modelo pedagógico define o nível de dificuldade do próximo exercício e do nível de conhecimento da atividade que o aluno acabou de realizar, de acordo com o tempo despendido, a quantidade de acertos e o nível de dificuldade da atividade realizada.

Para que o STI possa gerar o curso ao aluno, foi necessário implementar uma classe que realiza a interface LMSIntegration. O curso, nesse caso, representa o conceito a ser ensinado. Conforme os métodos da interface citadas na seção anterior, a classe LMSIntegration foi implementada baseada nos seguintes princípios:

- **Listar cursos:** retorna todos os conceitos habilitados para o aluno, atendendo os pré-requisitos do modelo de domínio, e utilizando o estado atual do modelo do aluno;

- **Abrir um curso:** acontece quando o usuário seleciona o conceito que ele deseja aprender ou rever. O STI monta o curso adicionando todos os pacotes do tipo **conteúdo**, e o pacote do tipo **exercício** de acordo com o nível de dificuldade sugerido pelo modelo pedagógico;
- **cmi.interactions alterado ou solicitado:** foi desenvolvido um objeto de aprendizagem direcionado para o que o STI pretende ensinar. Esse objeto de aprendizagem se comunica com o STI através do Data Model *cmi.interactions*. Por esse elemento, o OA envia a quantidade de acertos e o tempo despendido para realizar a atividade. Essas informações são processadas pelo modelo pedagógico para que possa atualizar o modelo do aluno.

O STI foi executado no servidor Apache Tomcat, utilizou de tecnologia JSF e banco de dados PostgreSQL.

O STI consegue gerar um curso a ser apresentado quando o aluno clica num conceito a ser aprendido. Essa adaptação pode incluir um ou mais pacotes de conteúdo, sejam parcialmente como totalmente. Como trabalho futuro desse STI, está a possibilidade de alterar a estrutura de um curso durante a sua utilização, ou seja, implementar a geração dinâmica de conteúdo.

5. Conclusões

O componente apresentado nesse artigo apresenta uma alternativa em manter conteúdo adaptável que segue o formato SCORM. A vantagem em utilizar tal alternativa é que o conteúdo não fica vinculado exclusivamente ao AIA, o que permite acrescer e alterar material instrucional independente do desenvolvedor do ambiente.

Em relação aos trabalhos correlatos, o componente aqui relatado não exige alterações na especificação do SCORM, e permite que se acesse individualmente os SCOs, para montar um novo curso, ou alterar as condições de seqüenciamento do original. O AIA pode se utilizar de um relacionamento do conceito a ser ensinado com os pacotes SCORM adicionados ao ambiente. Tal iniciativa foi descrita no trabalho de Schappo (2008), onde os pacotes relacionados aos conceitos devem ser categorizados entre **conteúdo** ou **exercício**, e ainda os exercícios também são categorizados entre fácil, médio e difícil, e dessa forma, o STI proposto montou os cursos utilizando-se dessas categorias.

Analogamente à utilização do LOM sugerido por Hoermann *et al* (2003) e Bhatt e Rao (2006), o componente também permite extrair informações de metadados dos cursos registrados, e fornecer uma lista de cursos que atendam um filtro montado pelo ambiente.

Com o BRUCE foi possível verificar a existência de todos os recursos necessários para um LMS manter pacotes SCORM e usuários, e comprova que o desenvolvedor não precisa de conhecimento sobre SCORM, exigindo dele somente o conhecimento quanto às *tags* disponíveis.

Como perspectiva futura pretende-se desenvolver um ambiente que explore a geração dinâmica de cursos, para que possa seguir as seis possibilidades de configuração de seqüenciamento descritas em Vahldick *et al* (2007). Acredita-se que com isso é

possível oferecer um componente completo para que os AIA possam transformar o conteúdo de forma que atenda às especificidades do ambiente e dos seus usuários.

6. Referências

ADL (2006) SCORM 2004 2nd Edition – Overview.

Barroca, L. et al. (2005) Conceitos Básicos. p 1-26. In: Desenvolvimento Baseado em Componentes. Editora Ciência Moderna, Rio de Janeiro.

Bhatt, C.B.; Rao, N.J. (2006) SCORM Metadata in the Context of Bloom-Vincenti Taxonomy and Intelligent Tutoring System. In: ICWE'06, July 11-14, 2006, Palo Alto, California, USA.

Brusilovsky, P. (2001) Adaptive Educational Hypermedia. In: Proceedings of Tenth International PEG conference, Tampere, Finland, June 23-26, 2001, p. 8-12.

Capuano, N.; Marsella, M.; Salerno, S. (2000) ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. In: Proceedings of ITS 2000, Montreal, Canada, June 19-23, 2000, Springer-Verlag.

Fischer, S. (2001) Course and exercise sequencing using metadata in adaptive hypermedia learning systems. In: Journal on Educational Resources in Computing.

Hoermann, S. et al. (2003) Building Structures of Reusable Educational Content Based on LOM. In: Conference on Advanced Information Systems Engineering (CAiSE).

IEEE LTSC. (2002) 1484.12.1. IEEE Standard for Learning Object Metadata.

Karampiperis, P; Sampson, D. (2006) Automatic Learning Object Selection and Sequencing Web-Based Intelligent Learning Systems. In: Web-based intelligent e-learning systems: technologies and applications, p. 56-71.

McClure, C. (2001) Software Reuse: a standards-based guide. IEEE, Los Alamitos.

Rey-López, M. et al. (2008) An extension to the ADL SCORM standard to support adaptivity: The t-learning case-study, Computer Standards & Interfaces, doi:10.1016/j.csi.2008.02.006

Schappo, J.C. (2008) Sistema Tutor Inteligente para o Ensino do Gerenciador de Armazenamento e Arquivos. TCC Bacharel em Ciências da Computação, FURB, Blumenau.

Shin, S.-O.; Lee, J.-O.; Baik, D.-K. Self-learning System Based on Metadata Management Module (MMM) for Providing Self-learning Service. p. 100-105. In: I IEEE International Symposium on Information Technologies and Applications in Education, 2007.

Silva, L.; Stringhini, D.; Mustaro, P.N.; Silveira, I. Adaptive Learning through Conceptual Lattice-based SCORM Meta-Objects. p. 739-748. In: VII International Conference on Information Technology Based Higher Education and Training, 2006.

Vahldick, A.; Santiago, R.; Raabe, A.L.A. (2007) Aplicação das Técnicas de Projeto Instrucional 4C/ID na Produção de Objetos de Aprendizagem em Conformidade com o SCORM Usando um Software Livre como Ferramenta de Autoria. In: Revista Novas Tecnologias na Educação.

Wiley, D.A. (2000). Learning Object Design and Sequencing Theory. (PH.D. Thesis). Brigham Young University. Provo, EUA.