CRIMSONHEX:
UM REPOSITÓRIO DE OBJECTOS DE APRENDIZAGEM

Leal, J.P. [1], **Queirós, R. [2]**

[1] Universidade do Porto, Faculdade de Ciências, Departamento de Ciências e Computadores, Centro de Investigação CRACS - zp@dcc.fc.up.pt
[2] Instituto Politécnico do Porto, Escola Superior de Estudos Industriais e de Gestão, Departamento de Informática, Centro de Investigação CRACS - ricardo.queiros@eu.ipp.pt

Resumo

Este artigo descreve o desenho e implementação de um repositório de problemas de programação denominado crimsonHex. A motivação para este trabalho surge do projecto EduJudge que tem como objectivo principal abrir o repositório de problemas de programação da Universidade de Valladolid (on-line-judge.uva.es) para utilizações pedagógicas no ensino secundário e superior. Este projecto integra um repositório de objectos de aprendizagem que será usado para gerir colecções de exercícios de programação e obter os exercícios adequados de acordo com um perfil de um determinado aluno.
O artigo inicia com uma análise aos repositórios actuais. Seguidamente, apresenta-se a arquitectura conceptual do repositório, baseado na metáfora de um objecto de aprendizagem como um livro numa biblioteca, e cria-se uma definição de problemas de programação como objectos de aprendizagem. Posteriormente foca-se a sua arquitectura concreta, o seu modelo de comunicação e detalhes da sua implementação. O artigo termina com algumas considerações finais.

Palavras-chave: *e-Learning*, Problemas de Programação, Avaliação Automática, SCORM.

1. INTRODUÇÃO

A Universidade de Valladolid (UVA) possui um júri/avaliador online (on-line-judge.uva.es) que é utilizado, há já vários anos, como uma ferramenta de formação, sobretudo para equipas que participam no *International Collegiate Programming Contest* (ICPC). De facto, o repositório da UVA usa problemas de vários concursos ICPC, incluindo todos os conjuntos de problemas de concursos regionais e finais mundais dos últimos sete anos.

O projecto EduJudge tem como objectivo abrir o sistema de avaliação de problemas de programação do repositório da UVA para um uso mais pedagógico no ensino secundário e superior. A arquitectura do sistema EduJudge integra três tipos de componentes: Repositórios de Objectos de Aprendizagem (*Learning Objects Repository* - LOR), Motores de Avaliação

(*Evaluation Engine* - EE) e Sistemas de Gestão e da Aprendizagem (*Learning Management Systems* - LMS). O nosso principal objectivo é desenhar e implementar um repositório de problemas de programação e posicioná-lo como um prestador de serviços para um conjunto alargado de sistemas com mecanismos de avaliação automática, como o LMS ou até mesmo um sistema de gestão de concursos de programação.

2. ESTADO DA ARTE

Um repositório de objectos de aprendizagem (*Learning Object* - LO) pode ser definido como "um sistema electrónico que armazena objectos e meta-dados sobre esses objectos" (ADL, 2003-1). A necessidade deste tipo de repositórios está crescendo à medida que os educadores estão cada vez mais disponíveis para a criação e o uso de conteúdos educacionais digitais. Não é de surpreender que um número crescente deste tipo de sistemas tenha surgido nos últimos anos. Os repositórios actuais (ADL, 2003-2) incluem várias características relevantes desde a edição de meta-dados através de formulários Web com controlo de vocabulário até à revisão e classificação dos objectos de aprendizagem.

Um repositório simples pode ser um sítio *web* que liste apenas os objectos de aprendizagem, com características semelhantes às de um motor de busca na *web*. Pode ser apenas um sítio *web* com informações sobre objectos de aprendizagem usando o Google, ou qualquer outro mecanismo de pesquisa, para indexar as suas páginas. Na verdade, a maioria dos repositórios especializados na *web* são basicamente motores de busca, com indexação de campos que descrevem em detalhe um objecto da aprendizagem. Um dos melhores exemplos é o repositório Merlot (*Multimedia Educational Resource for Learning and On-Line Teaching*). O repositório fornece apontadores para materiais de aprendizagem online, classificados em disciplinas académicas e inclui um mecanismo de busca.

A *Jorum Team* fez um levantamento exaustivo dos repositórios existentes (JORUM, 2006) e constatou que a maior parte destes sistemas não armazenam realmente os LOs. Eles apenas armazenam meta-dados descrevendo o LO, incluindo apontadores para a sua localização real na web. Alegadamente, estes sistemas nem deveriam ser chamados de repositórios. Uma vez que estes apenas possuem catálogos de apontadores para LOs, não há garantia que os LOs listados se encontrem efectivamente disponíveis. A mesma coisa acontece com as listas de qualquer motor de busca (que não são repositórios de páginas *web*), apesar do Google deixar aceder a uma versão em cache das páginas. Verificou-se também que mesmo os poucos repositórios que armazenam LOs não usam ou suportam, na sua maioria, um standard para a sua definição e empacotamento. Esse facto impede esses repositórios de expor as suas funcionalidades a outros componentes de software. Consequentemente, eles carecem das características necessárias para conectarem-se a LMSs e EEs, uma vez que esses componentes precisam de consultar e descarregar automaticamente os LOs.

Uma das normas mais relevantes e usadas para a distribuição de LOs é a especificação SCORM (*Sharable Content Object Reference Model*) que usa como standards para a estruturação e empacotamento dos dados, o *Instructional Management Systems Content Pacakaging* (IMS CP, 2004) e para a definição de meta-dados, o *Institute of Electrical and Electronic Engineers Learning Object Metadata* (IEEE LOM, 2002). Contudo, o IEEE-LOM, não foi especificamente concebido para contemplar os requisitos de avaliação automática de problemas de programação. Friesen (2004) menciona quatro formas que têm sido usadas para estender a IEEE-LOM:

- Combinar os elementos IEEE-LOM com elementos de outras especificações;

- Definir extensões aos elementos IEEE-LOM preservando a sua série de categorias;
- Simplificar o LOM, reduzindo o número de elementos LOM;
- Alargar e reduzir simultaneamente o número de elementos LOM.

Entretanto, as expectativas relacionadas com repositórios de LOs tem vindo a aumentarem. Alguns estudos (ADL, 2004) revelam que os utilizadores estão preocupados com questões que não estão totalmente suportadas pelos sistemas existentes, tais como a interoperabilidade com outros sistemas e o suporte de standards de definição e estruturação de conteúdos de aprendizagem. Tanto quanto é do nosso conhecimento, não existe actualmente nenhum repositório focado em LOs para avaliação automática. Na prática, seria muito difícil de criar actividades para um curso de programação usando qualquer um dos repositórios existentes. Em resumo, nós consideramos que os repositórios actuais não possuem as características necessárias para interagir com LMSs e EEs, nomeadamente:

- Disponibilidade dos LOs, e não apenas apontadores para os mesmos;
- Interoperabilidade com outros componentes de software através do uso de standards;
- Número relevante de problemas de programação.

3. PROBLEMAS DE PROGRAMAÇÃO COMO OBJECTOS DE APRENDIZAGEM

O repositório foi modelado usando um livro de biblioteca como metáfora: os LOs podem ser vistos como livros que o aluno (o leitor) acede através de um LMS (a biblioteca). Como num livro de uma biblioteca, o repositório possui um catálogo sobre o qual podemos fazer pesquisas eficientes. Depois de escolher um LO o aluno irá receber uma cópia actual do mesmo, tal como foi fornecida pelo seu autor, e não apenas um apontador para outro serviço. Como os LOs são uma espécie de *e-books*, o repositório vai ter um número ilimitado de cópias para emprestar, ao contrário de um livro físico de uma biblioteca.

Esta abordagem destaca o objecto como um objecto de aprendizagem: uma unidade que deve ser armazenada e catalogada como tal. Assim, tal como um livro não é fragmentado e as suas múltiplas páginas mantidas em diferentes prateleiras, o objecto da aprendizagem também não deve ser armazenado em tabelas diferentes de uma base de dados relacional, ou transformada, de forma alguma, para se adequar às exigências de um qualquer mecanismo de persistência. Como um objecto digital, o LO deve ser conservado e catalogado como foi originalmente apresentado pelo seu produtor, e ser recuperado sem qualquer alteração.

Por outro lado, um repositório de LOs não pode aceitar quaisquer tipos de dados BLOB (*Binary Large Object*). Voltando à nossa metáfora, um livro é um tipo específico de objecto: uma colecção de páginas com dimensões normalizadas, ligadas por uma capa. As prateleiras das bibliotecas estão programadas para livros com estas características. Além disso, os bibliotecários esperam ser capazes de ler, pelo menos, partes do livro, a fim de obter meta-dados para poder catalogá-los. Pela mesma razão, os LOs devem ter um conjunto de características comuns para serem gravados num repositório. Eles podem ser uma colecção de ficheiros de vários formatos, que se unem através de um formato comum, incluindo um manifesto que descreva o conteúdo do LO.

As bibliotecas também mantêm registos das requisições. Por isso, é muito simples para um bibliotecário saber quais são os livros mais populares, aqueles que nunca foram solicitados e os livros que os leitores levam mais tempo para ler. Na nossa opinião, este tipo de informação será interessante para a próxima geração de LMSs. Hoje em dia, a maioria dos LMSs apresenta os

LOs numa ordem predefinida. Com base em informações sobre o seu uso prévio, os LMSs serão capazes de mudar a ordem de apresentação de forma dinâmica. Por exemplo, um determinado LO pode ser classificado como de fácil resolução, mas os dados do seu uso revelarem que não foi resolvido pela maioria dos estudantes num nível introdutório, portanto, deve ser indicado apenas para os estudantes em níveis avançados. Para acomodar estas características, o nosso repositório dará a opção ao LMS para gravar informações sobre o uso de um LO e fornecer estatísticas sobre esses dados.

Baseado nesta metáfora, passou-se para a definição de problemas de programação como LOs de acordo com a especificação IMS CP. Um objecto IMS CP é um ficheiro simples – normalmente um ficheiro ZIP – que inclui no nível superior um manifesto, que descreve os recursos constituintes do pacote. Os recursos são elementos físicos tais como páginas web, PDFs, elementos multimédia, etc.

Esta norma foi criada para definir um LO genérico e não especificamente para problemas de programação. Em particular, os esquemas IMS CP (incluindo a IEEE-LOM) não possuem as características para descrever todos os recursos necessários para suportar a avaliação automática de problemas de programação. Por exemplo, não há como definir o papel de recursos específicos, nomeadamente, os casos de teste ou o programa solução para o problema. Felizmente, o IMS CP foi concebido de forma a ser simples de estender. Nesse sentido, todos os meta-dados relacionados com a avaliação automática são codificados através de elementos de um novo esquema – o *EduJudge Meta-data Specification* (EJ MD). Esta combinação enquadra-se nas diferentes abordagens apontadas por Friesen para a extensão dos meta-dados e é semelhante ao perfil de aplicação SCORM 1.2, que estende o IMS CP com mecanismos sofisticados de comunicação e sequenciação. Esta extensão está também em conformidade com o *IMS Package Conformance Level 1*: o pacote inclui um ficheiro manifesto (`imsmanifest.xml`) que contêm extensões adicionais, referenciadas no manifesto através de um novo espaço de nomes e definidas através de um esquema incluído no pacote. O pacote IMS CP está representado na Fig. 1:

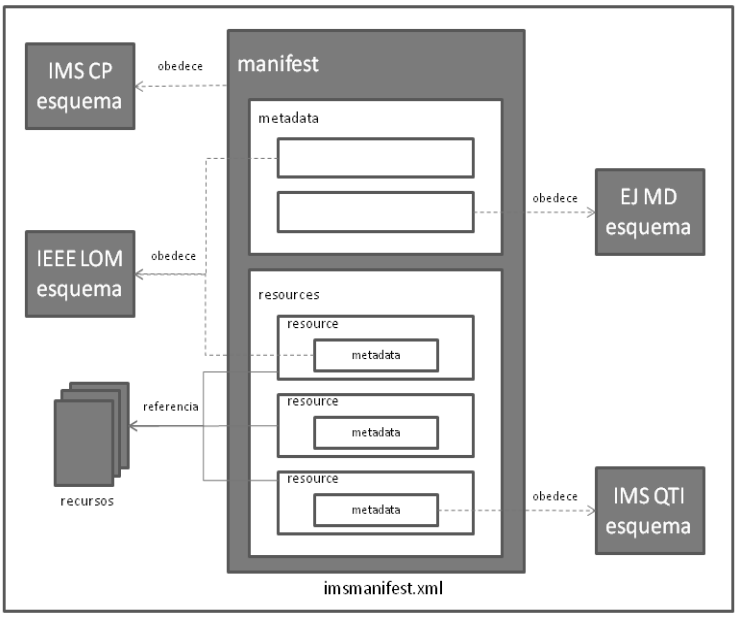

Figura 1 - Estrutura de um problema de programação como um objecto de aprendizagem

O arquivo contém vários ficheiros representados no diagrama como rectângulos cinza. O manifesto é um ficheiro *Extensible Markup Language* (XML) e a estrutura dos seus elementos é representada por rectângulos brancos. Os diferentes elementos do manifesto obedecem a diferentes esquemas incluídos no mesmo arquivo, como representado pelas setas tracejadas: o elemento raiz do manifesto está em conformidade com o esquema IMS CP; os elementos da secção `metadata` obedecem aos esquemas IEEE LOM ou EJ MD; os elementos `metadata` dentro dos recursos estão de acordo a especificação IEEE LOM ou *IMS Question & Test Interoperability* (IMS QTI, 2005). Os elementos `resource` apontam para recursos no pacote e são representados através de setas sólidas.

4. ARQUITECTURA DO CRIMSONHEX

O repositório irá desempenhar um papel fundamental na arquitectura global do Edujudge, pois irá actuar como um fornecedor de serviços para os outros sistemas de *e-Learning*. Os clientes do repositório devem conhecer a sua arquitectura, nomeadamente, os seus componentes e as suas funções, baseado na especificação *IMS Digital Repositories Interoperability* (IMS DRI, 2003).

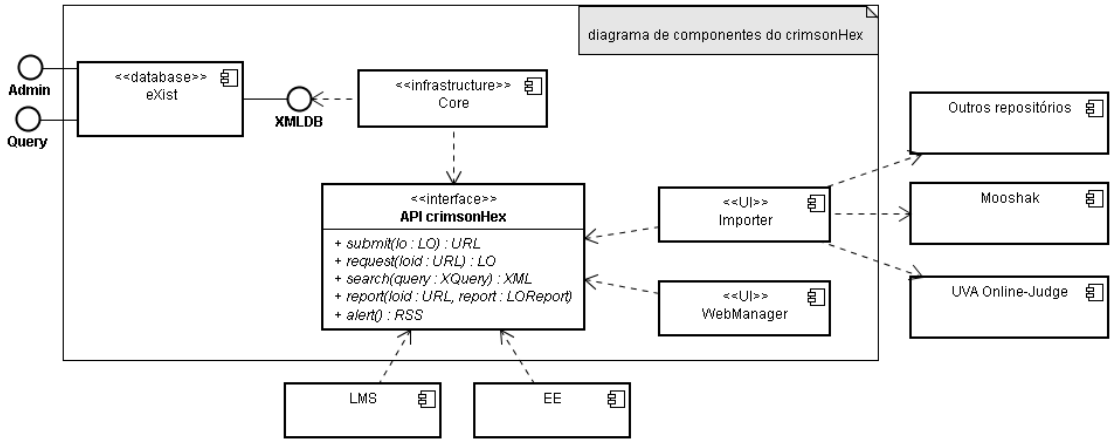

Figura 2 – Diagrama de componentes do repositório crimsonHex

4.1 Componentes

No desenho do crimsonHex foram definidos alguns requisitos iniciais que pretendemos que o repositório obedeça, nomeadamente, ser simples e eficiente. Na verdade, o núcleo das funções do repositório é o upload e download do LO - um arquivo ZIP - que são intrinsecamente funções simples que podem ser implementadas quase directamente sobre o protocolo de transporte. A Fig. 2 ilustra um diagrama UML com três componentes:

- **Core**: expõe as principais características do repositório, para os componentes externos, tais como o LMS e o EE, e para os componentes internos - o WebManager e o Importer;
- **Web Manager**: permite a criação, gestão de versões e o upload de LOs e meta-dados, através de uma interface Web que controla o cumprimento de vocabulários pré-definidos. Suporta também a exportação para padrões, como por exemplo, o *Dublin Core Metadata*;

- **Importer**: preenche o repositório com problemas de programação de repositórios já existentes, em especial o UVA Online-Judge.

O Core apresenta um conjunto mínimo de funções (segundo o IMS DRI) e pode ser eficientemente implementado através de um componente simples e estável. As restantes funcionalidades são relegadas para componentes auxiliares que se encontram ligados ao componente central. Outros componentes podem ser integrados no núcleo, potenciando a natureza *plug-and-play* do repositório.

4.2 Modelo de comunicação e principais funções

O ciclo de vida de um LO no repositório inicia-se com a função de submissão/armazenamento do LO. Depois disso, o LO está disponível para a pesquisa e entrega a outros sistemas de *e-Learning*. A Fig. 3 mostra um diagrama UML com a sequência completa da interacção entre o repositório e os outros sistemas. Para além do repositório, pode-se distinguir dois tipos de sistemas: o LMS que apresentará o exercício de programação para o aluno e o EE responsável pela avaliação e classificação automática da tentativa de resolução enviada pelo aluno.

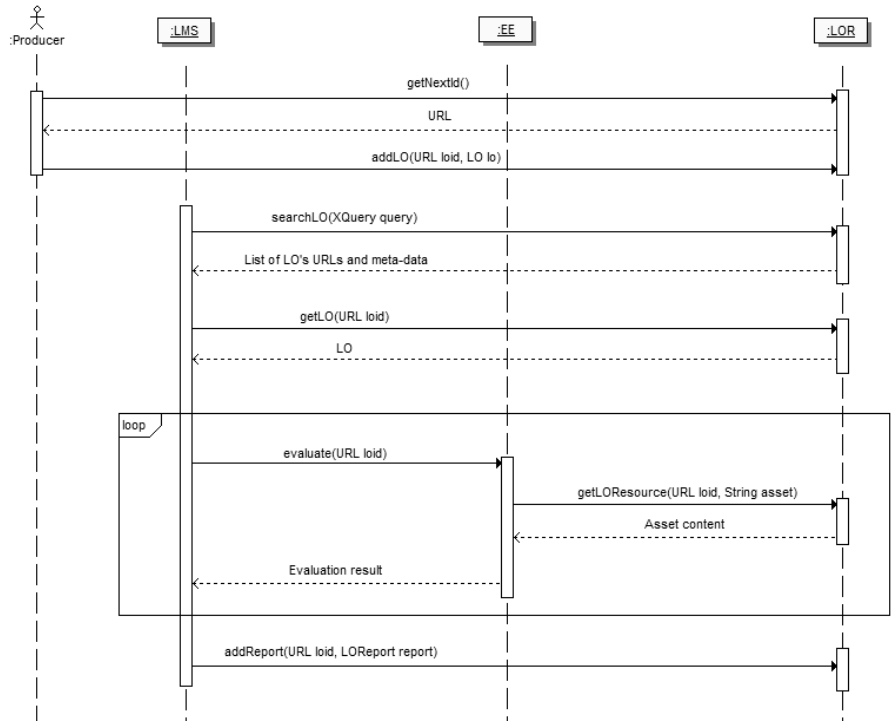

Figura 3 - Diagrama UML que modela a comunicação dos sistemas de e-Learning com o LOR

As funções apresentadas no diagrama anterior, baseiam-se na especificação IMS DRI. No entanto, estendemos a especificação para suportar a gravação de relatórios de utilização de um LO. Esta abordagem será, na nossa opinião, a base para uma próxima geração de LMS com a capacidade de adequar a ordem de apresentação dos exercícios de programação de acordo com as necessidades de um determinado aluno. Para cumprir as normas estabelecidas, o IMS DRI recomenda a implementação das funções básicas do repositório como serviços *web*. Foram implementados dois tipos distintos de interfaces para serviços *web*: *Simple Object Access*

Protocol (SOAP) e *Representational state transfer* (REST). A razão para tal é a de promover a sua utilização para uma futura adaptação a diferentes estilos arquitectónicos.

4.3 Implementação

Para o armazenamento interno dos LOs analisamos várias abordagens, desde o uso do sistema de ficheiros até bases de dados relacionais, mas optamos por usar bases de dados XML-Nativas. Este tipo de base de dados baseia-se na tecnologia XML e é indicado para esquemas de dados difíceis de adequar ao modelo relacional. Das várias alternativas existentes, seleccionou-se o eXist XML Database, um projecto de código aberto, devido às seguintes características:

- **Interrogação** (XPath/XQuery/XUpdate): os clientes do repositório podem fazer consultas XQuery e obter os dados no formato XML. O uso do XQuery num pedido de pesquisa está de acordo com a especificação IMS DRI;
- **Modularização**: a BD encontra-se dividida em colecções (directórios lógicos) que irá permitir uma melhor organização e modularização das pesquisas;
- **Indexação**: criação automática de índices que vão aumentar a eficiência da pesquisa na BD. Destaca-se o índice estrutural usado na execução de expressões XPath/XQuery;
- **Comunicação**: acesso às funções core do repositório através das interfaces HTTP: SOAP, REST, WebDAV, XMLRPC e Atom;
- **Transformação**: uso de *Extensible Stylesheet Language Transformations stylesheets* (XSLT): versão 1.0 (Apache Xalan) ou versão 2.0 (SAXON).

Apesar do eXist suportar validação implícita dos documentos XML esta não é extensível a arquivos de ficheiros. Por isso o crimsonHex implementa a sua própria validação de LOs, testando a sua conformidade com o nível 1 da norma IMS CP, quanto à estrutura do arquivo e quanto à estrutura do manifesto. A validação de um manifesto IMS CP é um processo bastante moroso devido ao grande número de esquemas envolvidos, com reflexo no tempo de submissão de um LO para o repositório. Para resolver este problema, criamos um mecanismo de *cache* que persiste os esquemas na primeira submissão e os reutiliza em pedidos posteriores.

Foram também feitos testes ao sistema de forma a recriar múltiplos cenários de utilização do repositório. Para além dos testes unitários às classes principais, foram feitos testes que invocam aleatoriamente as funções do *core* do repositório, tendo sido detectados e corrigidos alguns erros que só surgiriam em produção.

5. CONSIDERAÇÕES FINAIS

Neste trabalho é descrito o desenho e implementação de um repositório de exercícios de programação e a sua interacção com outros sistemas de e-Learning. A principal contribuição deste trabalho é a extensão das actuais especificações baseadas no padrão IMS aos requisitos específicos de avaliação automática. Os principais conceitos focados foram:

- A definição da programação problemas como LO;
- A estrutura do repositório e as suas funções básicas.

Para a primeira parte, definimos um modelo de avaliação de base para os problemas de programação e estendemos a especificação IMS CP com um esquema – EJ MD - para a representação de meta-dados relacionados com a avaliação automática. Para a segunda parte,

tomámos por base a especificação IMS DRI que define as funções fundamentais de um repositório e estendemos a norma para suportar a gravação de relatórios de utilização de LOs. Este artigo descreve ainda a implementação de um repositório segundo as extensões propostas por nós. O repositório crimsonHex é também uma das contribuições deste trabalho.

Agradecimentos

Este trabalho é parte do projecto intitulado " Integrating Online Judge into effective e-learning", com o projecto número 135221-LLP-1-2007-1-ES-KA3-KA3MP. Este projecto tem sido financiado com o apoio da Comissão Europeia. Esta comunicação reflecte apenas a opinião do autor, pelo que a Comissão não pode ser considerada responsável por qualquer uso que possa ser feito das informações nela contidas.

REFERÊNCIAS BIBLIOGRÁFICAS

Academic ADL Co-Lab. **From Local Challenges to a Global Community:Learning Repositories and the Global Learning Repositories Summit**, nov. 2003. Disponível em: http://www.academiccolab.org/resources/FinalSummitReport.pdf. Acesso em: 10 jul. 2008.

Academic ADL Co-Lab. **Learning Repositories included in Learning Repository Investigation,** out. 2003. Disponível em: http://www.academiccolab.org/resources/DraftRepositoriesList.pdf. Acesso em: 10 jul. 2008.

Academic ADL Co-Lab. **"What We Mean When We Say "Repositories" User Expectations of Repository Systems**, jul. 2004. Disponível em: http://www.hewlett.org/NR/rdonlyres/158FC043-A56F-43C6-ABA7-EB9A62656FCB/0/RepoSurvey2004-1.pdf. Acesso em: 10 jul. 2008.

FRIESEN, N. **Semantic and Syntactic Interoperability for Learning Object Metadata,** 2004. in: Hillman, D. (ed.) Metadata in Practice. Chicago, ALA Editions. Disponível em: http://www.cancore.ca/semantic_and_syntactic_interoperability.html. Acesso em: 11 jun. 2008.

IEEE LOM, **IEEE Standard for Learning Object Metadata** IEEE 1484.12.1-2002, 2002. Disponível em: http://www.ieeeltsc.org/standards/1484-12-1-2002/. Acesso em: 3 jul. 2008.

IMS CP, **IMS Content Packaging v1.1.4 Final specification**, 2004. Disponível em: http://www.imsglobal.org/content/packaging/index.html. Acesso em: 10 jul. 2008.

IMS DRI, **IMS Digital Repositories v1.0 Final specification**, 2003. Disponível em: http://www.imsglobal.org/digitalrepositories/index.html. Acesso em:10 jul. 2008.

IMS QTI, **IMS Question and Test Interoperability v2 Final specification**, 2005. Disponível em: http://www.imsglobal.org/question/index.html. Acesso em: 3 jul. 2008.

JORUM team, **E-Learning Repository Systems Research Watch**, 2006. Disponível em: http://www.jorum.ac.uk/docs/pdf/Repository_Watch_final_05012006.pdf. Acesso em: 10 jul. 2008.