Frameworks para as Interfaces de Busca e Catalogação de Conteúdo do *ContentNet*

Milene Selbach Silveira
Carlos José Pereira de Lucena

Departamento de Informática – PUC-Rio
[milene, lucena]@inf.puc-rio.br

PUC-RioInf.MCC13/00 Março, 2000

RESUMO

O uso do padrão IMS, para interoperabilidade de conteúdos educacionais, vem de encontro a necessidade global de intercâmbio de informações nesta área.

Dada a complexidade do padrão, este estudo visa propor *frameworks* para interfaces que privilegiem a utilização das mesmas por pessoas com pouca ou nenhuma experiência no uso deste.

Estes *frameworks* foram construídos para as interfaces de busca/catalogação de conteúdo do *ContentNet* e instanciados para o ambiente *AulaNet*.

Palavras-chave

IMS; meta-informação; WWW; *frameworks*; interface

ABSTRACT

The IMS standard for interoperability of educational contents meets the needs for information exchange in this area.

Due to the standard's complexity, this study proposes frameworks for user interfaces mostly for people with little or no experience in IMS.

These frameworks were built for the user interfaces of search and content cataloguing in *ContentNet*, as instantiated in the *AulaNet* environment.

Keywords

IMS; meta-data; WWW; frameworks; user interface

1 INTRODUÇÃO

Com a disseminação do uso da Internet para fins educacionais, cresce o número de cursos e/ou materiais pedagógicos nela disponibilizados. Este crescimento e o aumento da qualidade dos recursos encontrados, leva a necessidade de possibilitar-se o compartilhamento dos mesmos.

Um problema que dificulta este compartilhamento, é que, primeiro, é preciso conseguir localizar estes recursos e ter acesso aos mesmos. Esta dificuldade advém da grande quantidade de elementos disponíveis na WWW (*World Wide Web*) e na diversidade de locais onde os mesmos podem ser encontrados.

De encontro a estas necessidades, a plataforma IMS (*Instructional Management Systems*) [5], um consórcio de organizações acadêmicas, comerciais e governamentais, objetiva implementar um conjunto de padrões técnicos para software que facilitarão a publicação de conteúdos de aprendizagem distribuídos e o uso destes conteúdos, de diferentes formas, em múltiplos sistemas de aprendizado.

Esta plataforma utiliza, para especificação das características dos conteúdos educacionais, uma analogia feita com o sistema usado em bibliotecas, fazendo com que os recursos educacionais encontrados na Internet sejam catalogados de acordo com suas características, como, por exemplo, título, formato de apresentação e/ou nível educacional pretendido. Esta informação sobre a informação é denominada meta-informação.

Mas não adianta apenas catalogar esta meta-informação segundo um padrão. Necessitam-se, também, mecanismos de busca que facilitem sua localização. Para isto [11,12] propõe o *ContentNet*, que visa resolver os problemas relativos a localizar, avaliar, prover acesso, e manipular as informações disponíveis em servidores de conteúdo, compatíveis com esta plataforma.

O *ContentNet* está instanciado no contexto do *AulaNet* [6], que é um ambiente para a criação, manutenção, assistência e administração de cursos baseados na Web. Dentro deste processo de criação/aprendizado, o *AulaNet* considera três atores: o **Administrador**, o **Aluno** e o **Professor**. Para o *ContentNet*, o **Professor** é especialmente interessante, pois, além de sua função usual de criador de cursos, ele agora tem um novo papel, o de pesquisador de conteúdos.

Além disto, com a necessidade de catalogação de meta-informações dos conteúdos, uma nova figura surge, o **Catalogador**, que – em analogia ao bibliotecário "normal" – deve conhecer o formato a ser utilizado (IMS) na catalogação e cadastrar as meta-informações para cada conteúdo existente em seu servidor.

Com estes novos papéis/atores, surge a necessidade de interfaces que possibilitem esta catalogação/busca, auxiliando o usuário a ter uma interação eficaz e eficiente, conseguindo obter/catalogar os conteúdos desejados, de acordo com seus conhecimentos prévios (se necessário).

Este estudo visa propor um *framework* para estas novas interfaces (busca pelo **Professor** e catalogação pelo **Catalogador**), instanciando-as para o ambiente *AulaNet*.

2 PLATAFORMA EDUCAUSE-IMS

Na plataforma Educause-IMS [5], conforme citado anteriormente, a fim de habilitar a interoperabilidade de produtos educacionais, é proposto um padrão para o catalogação das meta-informações dos conteúdos educacionais.

Este padrão [3] especifica a sintaxe e semântica desta meta-informação, definindo os atributos requeridos para descrevê-la completa e adequadamente. Desta forma, os conteúdos educacionais poderão ser expressos em um formato que seja independente do próprio conteúdo em si; seja simples, porém extensível, a fim de ser fácil e amplamente adotável e aplicável; e que tenha suporte a busca e compartilhamento destes conteúdos.

Este padrão é um modelo hierárquico composto por três componentes:

1. **categoria**: uma categoria pode ter somente elementos de dados. Este componente compõe o nível mais alto da hierarquia;

2. **elementos de dados**: um elemento de dados pode ter valores tanto de elementos de dados quando de tipos abstratos de dados. Cada elemento de dados tem uma definição, descrita no **Dicionário** de elementos de dados;

3. **tipos abstratos de dados**: definem os valores possíveis para os elementos de dados.

Estes elementos estão organizados, conforme especificação da tabela 2.1.

Categoria Elemento de Dados: *Tipo Abstrato De Dados*	Exemplo: Informações Gerais Idioma: String
	Instanciação Específica: Informações_Gerais.Idioma: *pt-BR* significa que o recurso será descrito em português do Brasil

Tabela 2.1: Organização hierárquica das meta-informações

A figura 2.1 ilustra a hierarquia acima. No topo desta hierarquia está o elemento “raiz”. Cada elemento raiz contém muitos sub-elementos. Se um sub-elemento contém sub-elementos adicionais, ele é chamado um “ramo”. Sub-elementos que não contém quaisquer sub-elementos são chamados “folhas”. Este modelo hierárquico inteiro é chamado “estrutura de árvore” de um documento.

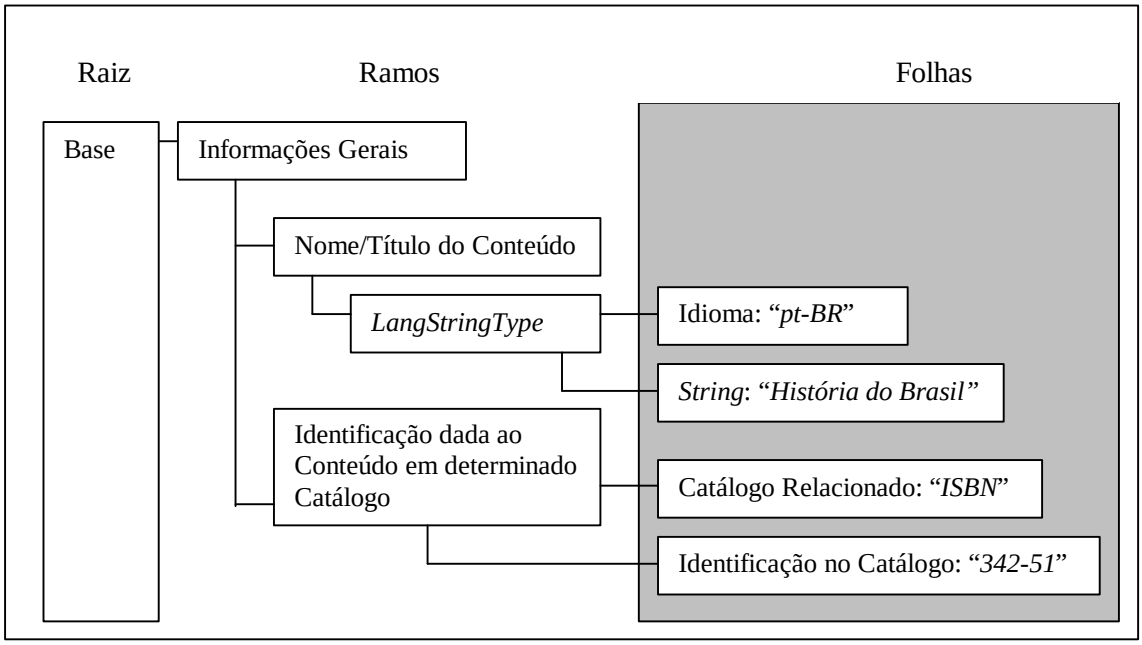


Figura 2.1: Visão Hierárquica: relacionamento entre a raiz, os ramos e as folhas

2.1 Categorias e Elementos de Dados

Para o armazenamento dos elementos de dados (meta-informações), é definido um **Dicionário**, com os termos recomendados para a classificação dos diferentes tipos de conteúdos educacionais.

Cada elemento é descrito pelas seguintes informações [4]:

- ***name***: como o elemento deve ser referenciado;
- ***explanation***: a definição do elemento;
- ***multiplicity* (M)**: quantos elementos são permitidos e se sua ordem é significativa;
- ***domain* (D)**: qual o vocabulário permitido para o elemento (somente é indicado quando há restrição ou sugestão de vocabulário);
- ***type***: indica se os valores dos elementos são textuais, numéricos ou uma data e, também, qualquer restrição a seu tamanho e formato;
- ***note***: informações adicionais sobre o elemento;
- ***example***: exemplo para o elemento.

Na tabela 2.2 tem-se uma ilustração das informações acima, para os elementos do exemplo da figura 2.1.

#	*name*	*explanation*	*M*	*D*	*type*	*note*	*example*
1	**Informações Gerais**	**Características independentes de contexto do conteúdo**	-	-	-	-	-
1.2	Título do Conteúdo	nome dado ao conteúdo	*Single value*	-	*LangString Type (1000 char)*	1. o titulo pode ser um já existente ou pode ser criado pelo indexador *ad hoc*. 2. corresponde ao elemento *Dublin Core DC.Title*	-
1.3	Identificação dada ao Conteúdo em determinado Catálogo	designação dada ao conteúdo	*Multiple unordered instance* (10 itens)	-	-	Uma das entradas do catálogo pode ser gerada automaticamente pela ferramenta	-
1.3.1	Catálogo relacionado	indicação da fonte do valor dado	Single value	-	*String (1000 char)*	Geralmente o nome do catálogo	ISBN, ARIADNE
1.3.2	Identificação no catálogo	valor atual	Single value	-	*LangString Type (1000 char)*	Geralmente o número no catálogo indicado em 1.3.1	2-7342-1, LEAO875

Tabela 2.2: Descrição dos elementos exemplificados na figura 2.1

Atualmente, as categorias e elementos de dados disponíveis, são os seguintes[1] [4]:

(*General*) Informações Gerais
- (*Title*) Nome/Título do Conteúdo
- (*CatalogEntry*) Identificação dada ao Conteúdo em determinado Catálogo
  - (*Catalogue*) Nome do Catálogo
  - (*Entry*) Identificação no Catálogo
- (*Language*) Idioma
- (*Description*) Descrição
- (*Keywords*) Palavras Chave
- (*Coverage*) Características Espaciais e Temporais do Conteúdo
- (*Structure*) Estrutura Organizacional
- (*AgregationLevel*) Tamanho da Informação

(*LyfeCicle*) Características do Desenvolvimento
- (*Version*) Versão
- (*Status*) Status
- (*Contribute*) Desenvolvedores (pessoas/organizações que contribuíram para o desenvolvimento)

  (*Role*) Tipo de Contribuição

[1] Entre parênteses, em itálico, encontram-se os nomes originais. Na primeira tabulação encontram-se as categorias e nas demais os elementos de dados. Ao lado encontra-se a descrição dada aos mesmo em português, para o *ContentNet* [11,12].

(*Entity*) Entidades Envolvidas
(*Date*) Data

(*MetaMetaData*) Dados de Catalogação
(*CatalogEntry*) Designação dada à Catalogação
(*Catalogue*) Catálogo Relacionado
(*Entry*) Identificação no Catálogo
(*Contribute*) Pessoas ou Organizações que contribuem à catalogação
(*Role*) Tipo de Contribuição
(*Entity*) Entidade(s) Envolvida(s)
(*Date*) Data
(*MetaDataScheme*) Nome da Estrutura da Catalogação
(*Language*) Idioma utilizado na Catalogação

(*Technical*) Características Técnicas
(*Format*) Formato
(*Size*) Tamanho Aproximado
(*Location*) Localização
(*Requirements*) Configuração e Plataforma
(*Type*) Tipo
(*Name*) Nome
(*MinimunVersion*) Versão Mínima
(*MaximunVersion*) Versão Máxima
(*InstallationRemarks*) Descrição de como Instalar o Recurso
(*OtherPlatformRequirements*) Informações Adicionais sobre a Plataforma
(*Duration*) Tempo de Duração

(*Educational*) Características Educacionais e/ou Pedagógicas
(*InteractivityType*) Tipo de Interação
(*LearningResourceType*) Tipo de Material
(*InteractivityLevel*) Nível de Interatividade entre Usuários e Recursos
(*SemanticDensity*) Utilidade x Tempo/Duração
(*IntendedEndUserRole*) Usuário Pretendido
(*LearningContext*) Nível Educacional do Aluno
(*TypicalAgeRange*) Idade do Usuário
(*Dificulty*) Grau de Dificuldade
(*TypicalLearningTime*) Tempo Aproximado para trabalhar com o Recurso
(*Description*) Comentários sobre como usar o recurso
(*Language*) Idioma

(*Rights*) Condições de Uso
(*Cost*) Há Custo para possuir o conteúdo?
(*CopyrightAndOtherRestrictions*) Há Restrição ao Uso?
(*Description*) Comentários das Condições de Uso

(*Relation*) Características do conteúdo em relação a outros recursos
(*Kind*) Natureza do Relacionamento
(*Resource*) Conteúdo a que se relaciona
(*Description*) Descrição de Outros Recursos

(*Annotation*) Detalhes sobre o uso Educacional
(*Person*) Autor
(*Date*) Data da Criação
(*Description*) Comentários

(*Classification*) Classificação
(*Purpose*) Características do Conteúdo
(*TaxonPath*) Caminho Taxonômico
(*Source*) Fonte da Classificação
(*Taxon*) Classificação
(*Id*) Identificação no Catálogo
(*Entry*) Nome do Catálogo
(*Description*) Descrição da Característica
(*Keywords*) Palavras Chave

2.2 Tipos Abstratos de Dados

Os tipos abstratos de dados, utilizados neste padrão, dividem-se em: *String*, *LangStringType*, *DateType* e *VocabularyType*. O *String* é o tipo base, formado por caracteres, e os demais baseiam-se nele em sua composição:

- *LangStringType*: indica um *String*, que pode ser escrito em até 10 idiomas diferentes. Este é um tipo muito particular, podendo descrever até 10 itens (10 traduções do item trabalhado), em uma lista desordenada. Estes itens são formados, cada um, por *Language* e *String*, onde *Language* (*string* com 100 char) tem um valor único e indica o idioma em que o item foi escrito como, por exemplo, "pt-BR", e *String*, um valor único, que indica o texto, no idioma relacionado, como, por exemplo, "História do Brasil";

- *DateType*: trabalha com datas e horas, sendo formado por dois itens, *DateTime* e *Description*, onde *Datetime* é um valor único (*string* de 200 char) como, por exemplo, 2000-02-10, e *Description* é do tipo *LangStringType (de 1000 char)*, como, por exemplo, *Fall semester 1999*. O uso de *LangStringType*, aqui, permite que uma mesma data seja descrita em idiomas diversos;

- *VocabularyType*: indica que há um vocabulário pré-definido, especificado em ***domain***. Este tipo é formado por dois itens, *Entry* e *Detail*, onde *Entry* (*string* de 2 char) indica o valor do dado pré-definido, como, por exemplo, um valor único entre 0 e 3 ou uma lista ordenada de 4 itens, com os valores {*Teacher, Author, Learner, Manager*}, e *Detail* (*LangStringType*, 1000 char), detalhes sobre o mesmo.

O *LangStringType* favorece a interoperabilidade de conteúdos, por permitir que as características dos mesmos (meta-informações) sejam descritas em vários idiomas diferentes, o que aumenta a possibilidade de difusão do mesmo.

Além de suas características acima descritas, cada tipo pode possuir tamanhos diferentes de *strings* e, pode diferir, também, sua multiplicidade, ocasionando uma série de possibilidades de combinação, conforme pode ser visto no exemplo da tabela 2.3.

type	***lenght***	***multiplicity***
		Multiple unordered instance (10, 15 ou 30 itens)
String	1, 30, 60, 100 ou 1000 *char*	*Single value* *Ordered list* (10 elementos) *Unordered list* (10 elementos)
LangStringType	50, 500, 1000 ou 2000 *char*	*Single value* *Ordered list* (10 elementos) *Unordered list* (5 ou 10 elementos)
VocabularyType	-	*Ordered list* (4 ou 10 elementos) *Unordered list* (10 elementos)

Tabela 2.3: Descrição das possíveis variações dos tipos abstratos de dados

Além disto, os elementos podem conter sub-elementos (como no caso de *CatalogueEntry*, formado por *Catalogue* e *Entry*) ou, também, ser formados por listas de valores, ao invés de um valor único (*single value*). Estas listas podem ser *ordered*, onde a ordem dos elementos é fator importante ou *desordered*, em caso contrário.

3 INTERFACES PARA A BUSCA/CATALOGAÇÃO DE CONTEÚDO

A diversidade de possibilidades e restrições para cada elemento, vista no capítulo anterior, reflete-se na hora da construção da interface de catalogação/busca do *ContentNet* [11,12].

Para estas interfaces, além deste controle, ainda deve ser considerado que existem dois possíveis usuários, com conhecimentos distintos sobre estes elementos: o **Professor**, que não necessita conhecer o padrão IMS, devendo, desta forma, ser transparente suas implicações e auto-explicativa sua interface, e o **Catalogador**, que deve conhecer o padrão e, nem por isso, sua interação deve ser menos facilitada, principalmente pela quantidade de vezes que terá que fazer a mesma tarefa (catalogação de conteúdo).

3.1 Busca de conteúdo: usuário Professor

O usuário **Professor** fará uso da ferramenta de busca do *ContentNet*, para procurar algum conteúdo desejado. Ele não necessita conhecer IMS, mas, sim, precisa ter acesso a seus elementos para poder informar os dados necessários para a busca.

Estes dados, que o usuário precisa informar, são[2]:

- tipo de conteúdo a pesquisar: Curso, Aula, Tarefa, Webliografia, Auto Avaliação, Livro Texto, Demonstração, Bibliografia ou Questão (figura 3.1);

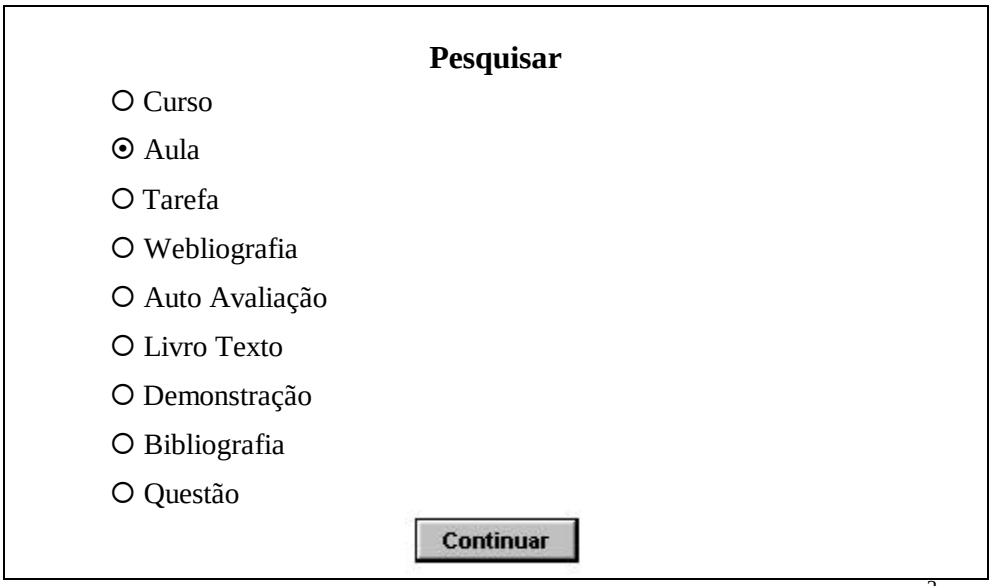


Figura 3.1: Esboço de Interface para Busca por Tipo de Conteúdo[3]

[2] Os elementos, a partir de agora, estão instanciados para o ambiente *AulaNet*, de acordo com o *ContentNet* [11,12].

[3] Estes esboços foram feitos como estudo inicial para a construção da interface, a partir de cenários [1] construídos pelo designer do *ContentNet* [11].

- grupo de atributos pelo qual quer fazer a pesquisa: Detalhes sobre o Uso Educacional, Características do Conteúdo em Relação a outros Conteúdos, Características Educacionais e/ou Pedagógicas, Descrição da Classificação, Condições de Uso, Características Técnicas, Características de Desenvolvimento, Informações Gerais e/ou Características da Descrição dada ao Conteúdo. Estes grupos correspondem as **categorias** no padrão IMS;

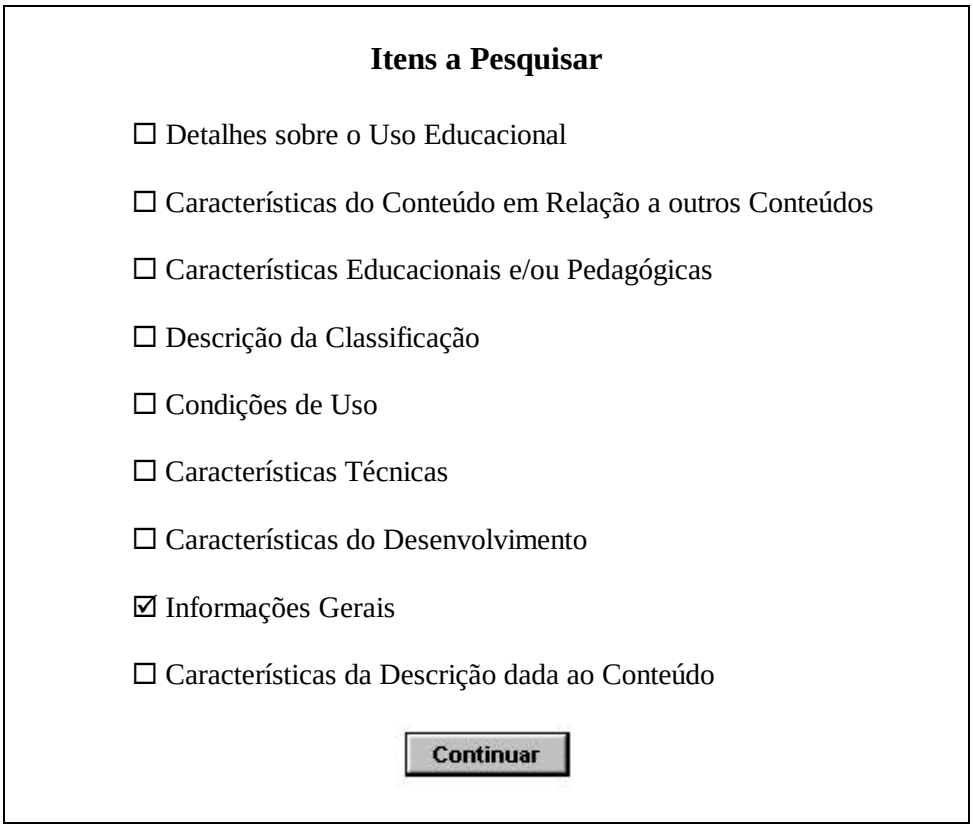


Figura 3.2: Esboço de Interface para Busca por Grupo de Atributos

- os atributos conhecidos ou desejados, de acordo com o grupo acima. Estes atributos correspondem aos **elementos de dados** do padrão, possuindo – cada um deles - um **tipo abstrato** relacionado e, também, conforme o caso, restrições de vocabulário.

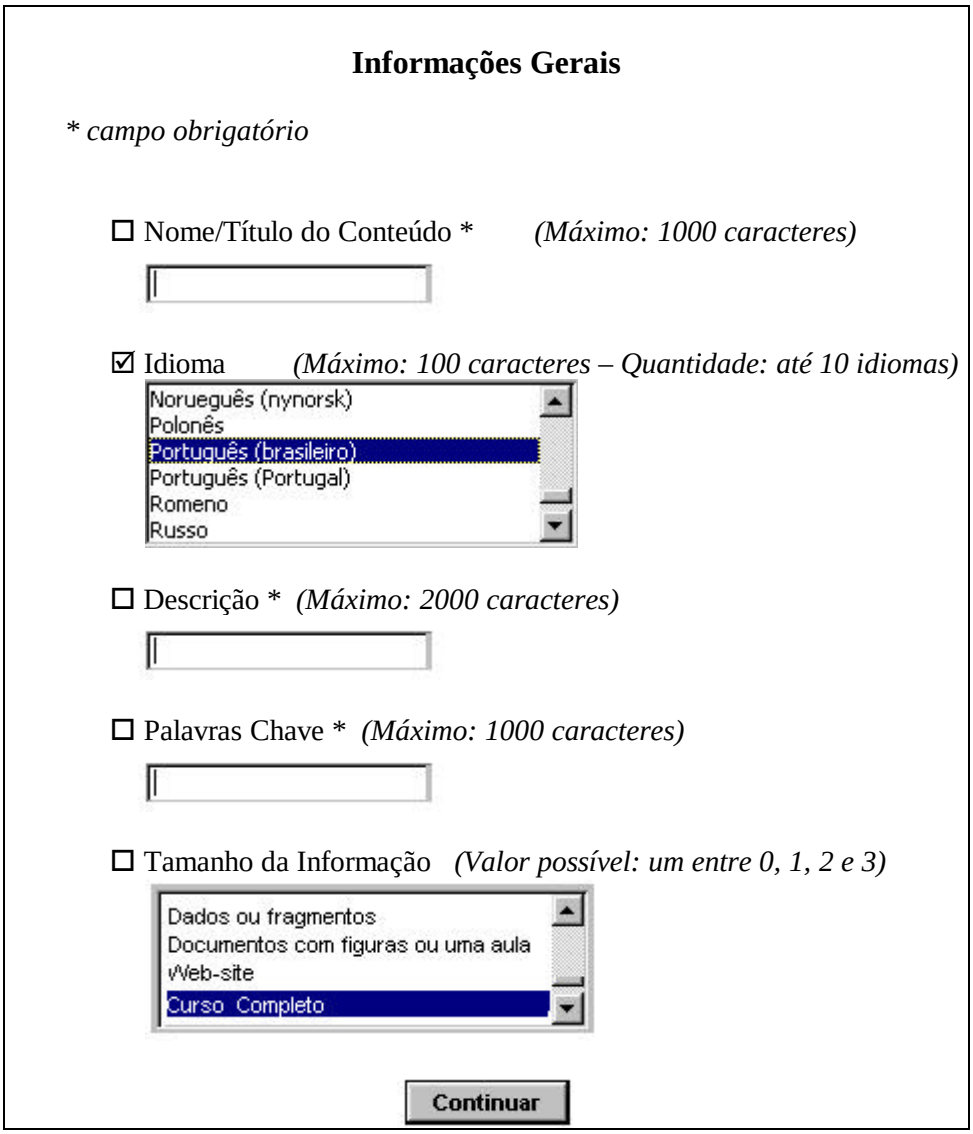


Figura 3.3: Esboço de Interface para Informação dos Atributos para a Busca, com especificação de suas restrições

3.2 Catalogação de conteúdo: usuário Catalogador

O usuário **Catalogador** fará uso da ferramenta de catalogação do *ContentNet*, para catalogar as meta-informações dos conteúdos de seu servidor (há um catalogador para cada servidor de conteúdos).

Para a catalogação destas meta-informações, o usuário deve informar:

- tipo de conteúdo a catalogar: Curso, Aula, Tarefa, Webliografia, Auto Avaliação, Livro Texto, Demonstração, Bibliografia ou Questão (mesmos itens da figura 3.1);

- a partir do tipo de conteúdo, deve ser informado qual especificamente irá ser

catalogado;

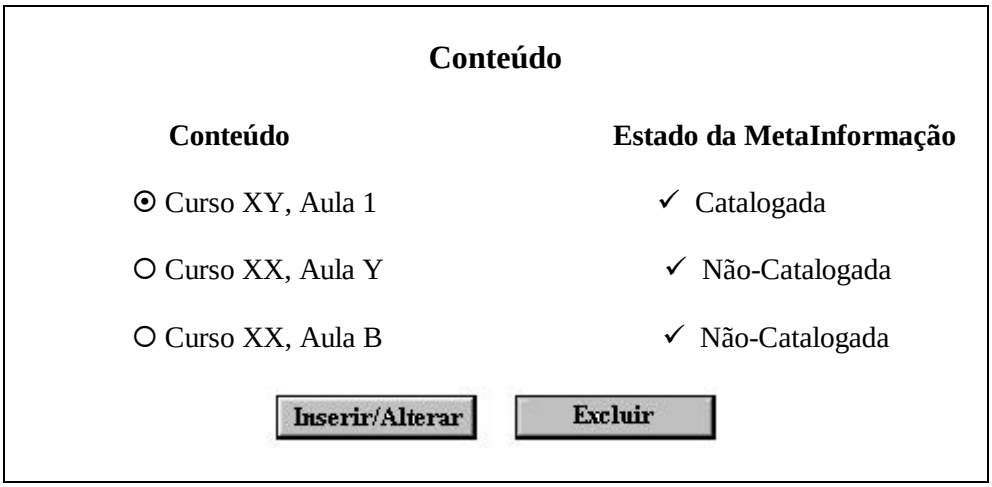


Figura 3.4: Esboço de Interface para Seleção de Conteúdo a Catalogar

Após ser indicado o conteúdo desejado, o usuário deve cadastrar suas meta-informações, onde terá todos os grupos de atributos listados (conforme figura 3.3).

3.3 Desafios da Interface: soluções encontradas

Os tipos abstratos de dados, bem como as restrições de vocabulário, levaram ao estudo sobre a melhor forma de representá-los na interface.

A partir dos esboços elaborados (alguns deles vistos nos exemplos das primeiras figuras deste capítulo), as interfaces para os usuários **Professor** e **Catalogador**, foram evoluindo até chegar as que estão em testes atualmente [11].

Nos itens abaixo, encontram-se os tipos de *widgets* implementados [9,10] e sua relação com as meta-informações que o determinam:

- campos que permitem uma escolha entre várias opções:

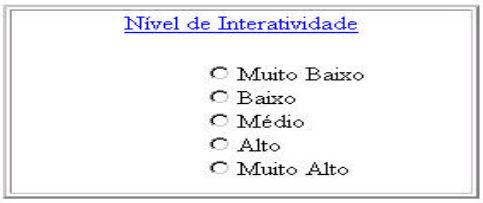
Meta-informação	Item de Interface
type: *String* (1 char) ***multiplicity***: *single value* ***domain***: {0, 1, 2, 3, 4} ***translation***[4]: Muito Baixo | Baixo | Médio | Alto | Muito Alto	

Tabela 3.1: Escolha Simples

[4] Este item foi incluído especialmente para o *ContentNet*, devido ao fato do mesmo ser instanciado para o *AulaNet*, que é em português.

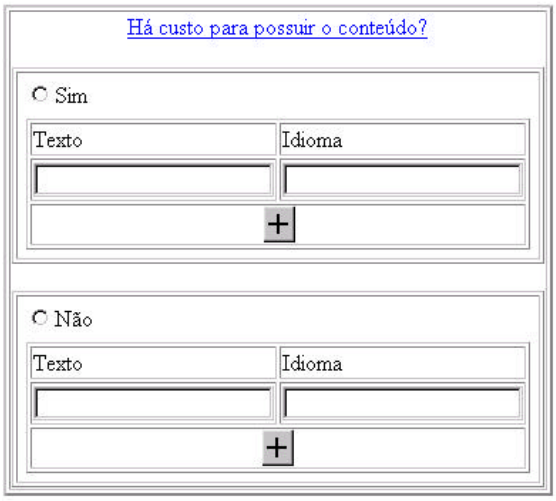
Meta-informação	Item de Interface
type: *Vocabulary Type* ***multiplicity***: *single value* ***domain***: {3=*yes*, 4=*no*} ***translation***: *Sim* | *Não*	

Tabela 3.2: Escolha Simples

- campos para preenchimento de dados:

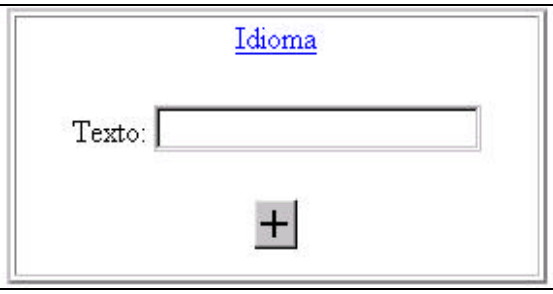
Meta-informação	Item de Interface
type: *String* (*1000 char)* ***multiplicity***: *Unordered (10 itens)* ***domain***: Language ID ***translation***: -	

Tabela 3.3: Preenchimento de Dados

- campos para preenchimento de dados com possibilidade de diversas traduções (elemento com sub-elementos):

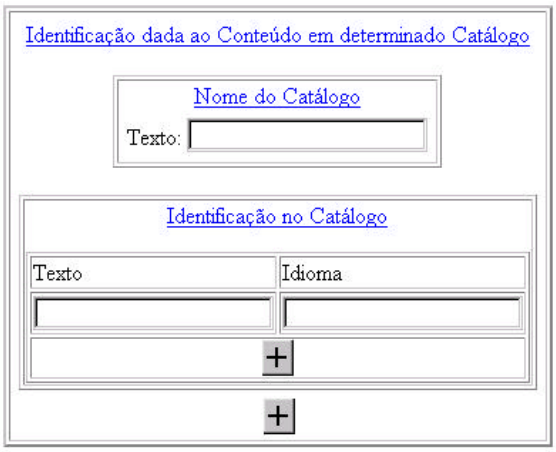
Meta-informação	Item de Interface
Identificação... ***multiplicity***: *multiple unordered instance* (10 itens) Nome do Catálogo ***type***: *String* (1000 char) ***multiplicity***: *single value* ***domain***: - ***translation***: - Identificação no Catálogo ***type***: *LangStringType* (1000 char) ***multiplicity***: *single value* ***domain***: - ***translation***:-	

Tabela 3.4: Sub-elementos

- campos com entrada de data ou período:

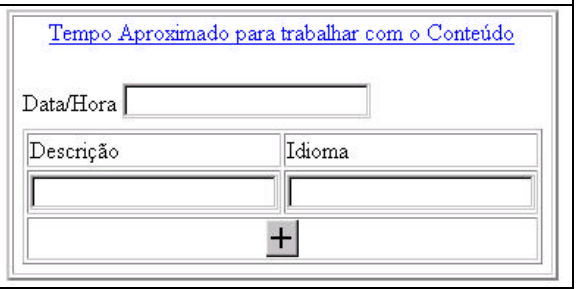
Meta-informação	Item de Interface
type: *DateType* ***multiplicity***: *single value* ***domain***: - ***translation***: -	

Tabela 3.5: Entrada de Data/Período

- campos com entrada de ordem para os dados:

Meta-informação	Item de Interface
type: *VocabularyType* ***multiplicity***: *ordered (4 itens)* ***domain***: *{3=Teacher; 4=Author; 5=Learner; 6=Manager}* ***translation***: Professor| Autor| Aprendiz | Gerente	

Tabela 3.6: Ordenação da Entrada

Em vários dos exemplos acima, é possível notar o uso de um botão com um símbolo de adição. Este botão é utilizado quando é permitida a entrada de mais de um valor para cada item. Isto ocorre, tipicamente, quando o tipo abstrato de dados em questão é o *LangStringType* (que permite a entrada de dez traduções (descrição e idioma) para cada item) ou listas (com influência ou não de ordem) de valores. O uso combinado de listas e de *LangStringType* pode ocasionar dois destes botões em um mesmo item, conforme pôde ser visto na tabela 3.4.

O acionamento deste botão expande as opções restantes do item, como pode ser visto na figura 3.7[5].

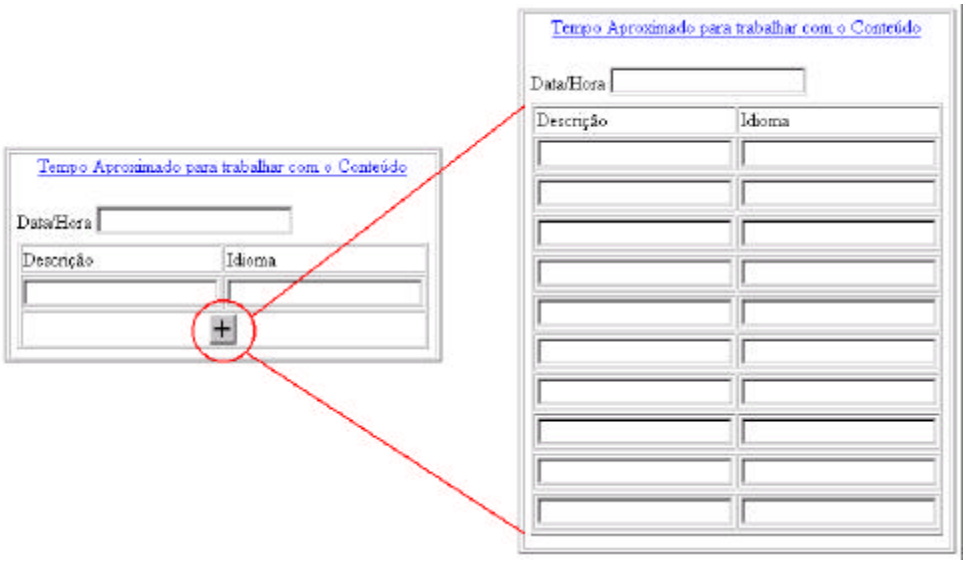


Figura 3.7: Expansão do botão +

[5] Esta forma de expansão de campos (+) foi especificada, mas ainda não implementada e testada neste ambiente, mas já é padrão em outros [13].

Além dos itens tratados anteriormente, outro recurso utilizado foi o uso de mensagens de ajuda locais [13], onde, para cada item a ser preenchido pelo professor e/ou catalogador, é possível aceder a informações sobre o mesmo no ato do preenchimento.

Este recurso foi utilizado, principalmente, com o objetivo de facilitar a interação do usuário **Professor**, visto que o mesmo não precisa conhecer o padrão IMS para utilizar a ferramenta de busca de conteúdos.

Este mecanismo de ajuda é ativado quando o usuário "clica" sobre o título do item a ser preenchido (na forma de um *link* de hipertexto, como visto na figura 3.8). Esta ação abre uma pequena janela no topo da janela corrente, com uma explicação sobre como preencher o campo correspondente, em termos de sua sintaxe e/ou semântica.

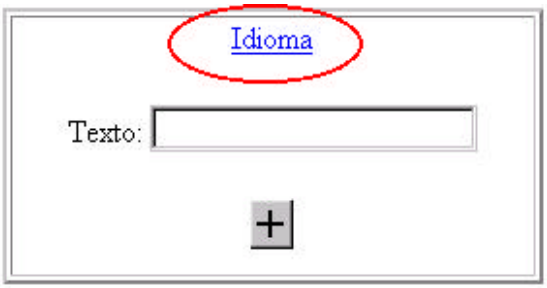


Figura 3.8: *Link* para a ajuda local

Para composição do texto desta ajuda, são usados os elementos de meta-informação ***name***, ***explanation***, ***note*** e ***example***, sendo que estes dois últimos são utilizados como complemento a explicação usual (***explanation***) quando disponíveis. Esta relação pode ser visa com mais clareza nos exemplos das tabelas 3.7, 3.8 e 3.9.

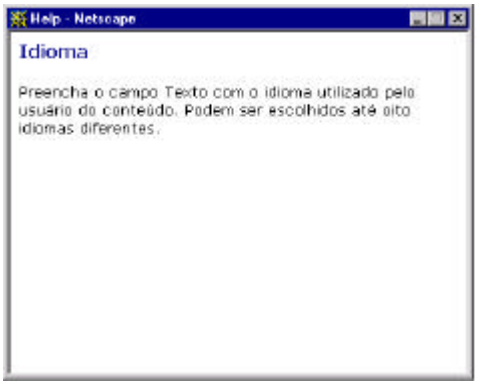
Meta-informação	Ajuda Local
name: Idioma ***explanation***: Preencha o campo Texto com o idioma no qual o conteúdo foi escrito. Podem ser escolhidos até oito idiomas diferentes. ***note***: - ***example***: -	

Tabela 3.7: Ajuda local para o elemento Idioma

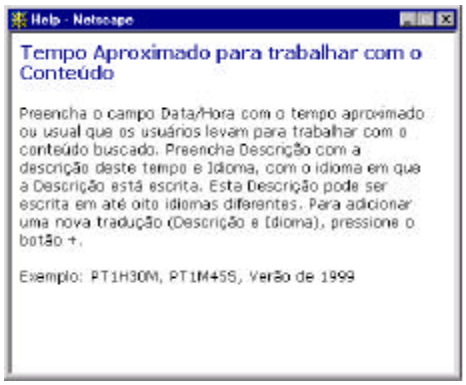
Meta-informação	Ajuda Local
name: Tempo Aproximado para trabalhar com o Conteúdo ***explanation***: Preencha o campo Data/Hora com o tempo aproximado ou usual que os usuários levam para trabalhar com o conteúdo buscado. Preencha Descrição com a descrição deste tempo e Idioma, com o idioma em que a Descrição está escrita. Esta Descrição pode ser escrita em até oito idiomas diferentes. Para adicionar uma nova tradução (Descrição e Idioma), pressione o botão +. ***note***: - ***example:*** PT1H30M, PT1M45S, Verão de 1999	

Tabela 3.8: Ajuda local para o elemento Tempo Aproximado para trabalhar com o Conteúdo

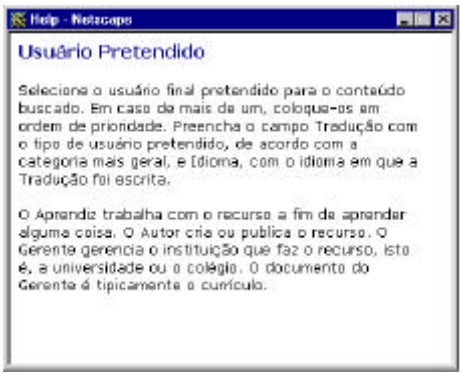
Meta-informação	Ajuda Local
name: Usuário Pretendido ***explanation***: Selecione o usuário final pretendido para o conteúdo buscado. Em caso de mais de um, coloque-os em ordem de prioridade. Preencha o campo Tradução com o tipo de usuário pretendido, de acordo com a categoria mais geral, e Idioma, com o idioma em que a Tradução foi escrita. ***note***: O Aprendiz trabalha com o recurso a fim de aprender alguma coisa. O Autor cria ou publica o recurso. O Gerente gerencia o instituição que faz o recurso, isto é, a universidade ou o colégio. O documento do Gerente é tipicamente o currículo. ***example:*** -	

Tabela 3.9: Ajuda local para o elemento Usuário Pretendido

4 *FRAMEWORKS* PARA AS INTERFACES DE BUSCA/CATALOGAÇÃO DE CONTEÚDO

A partir dos estudos realizados sobre as interfaces, foram propostos *frameworks* para as interfaces de busca e catalogação de conteúdo no *ContentNet*, com sua instanciação para o ambiente *AulaNet*.

4.1 O que são *Frameworks*?

Segundo [2], a definição de *framework* varia. Uma definição usada freqüentemente é "um *framework* é um *design* reusável de todo ou parte de um sistema que é representado por um conjunto de classes abstratas e a forma que suas instâncias interagem". Outra definição comum é "um *framework* é o esqueleto de uma aplicação que pode ser customizado pelo desenvolvedor da aplicação". Estas não são definições conflitantes: a primeira descreve a estrutura de um *framework* enquanto a segunda descreve seu propósito.

Usualmente, vários aspectos de um *framework* de aplicação não podem ser completamente antecipados. Estas partes - genéricas para que possam ser facilmente estendidas e customizadas, ou seja, adaptáveis a necessidades específicas – são denominadas *hot-spots* flexíveis [2,8] (figura 4.1).

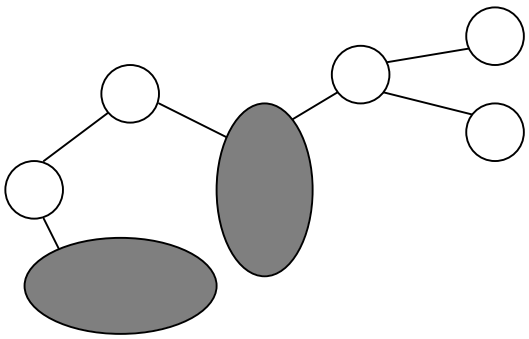


Figura 4.1: *Hot-spots* flexíveis do *framework (*regiões sombreadas)

Estes *hot-spots* são descobertos na análise do domínio ou providos por um especialista no domínio [2]. A dificuldade de um bom *design* orientado a objetos é identificar estes aspectos do domínio de aplicação que têm que ser mantidos flexíveis, tanto que um *framework* é considerado "bem projetado" se ele provê *hot-spots* adequados para adaptações. [8]

Conforme citado anteriormente, esta flexibilização permite que o *framework* possa ser usado como base para o desenvolvimento de diferentes aplicações no domínio de aplicação por ele capturado (figura 4.2).

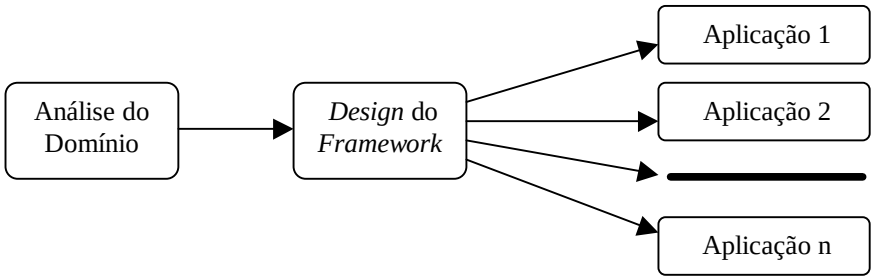


Figura 4.2: Desenvolvimento de Aplicações baseadas em *frameworks* [7]

4.1.1 Pontos Fortes e Fracos de um *Framework*

Segundo [7], o desenvolvimento de *frameworks* orientados a objetos tem potenciais pontos fortes e fracos, conforme pode ser visto na tabela 4.1.

Pontos Fortes	**Pontos Fracos**
• uma diminuição real em linhas de código que tem que ser desenvolvidas se a funcionalidade requerida pela aplicação e a funcionalidade capturada pelo *framework* tem um alto grau de similaridade; • o código do *framework* já está escrito e depurado; • o *framework* oferece reuso de *design* e não somente de código. Este reuso de *design* feito por outros pode transferir conhecimento e experiências de *design* ao usuário do *framework*; • aspectos de eficiência do software podem ser previstos pelos desenvolvedores do *framework* o que normalmente não pode ser feito em um planejamento de projeto de aplicação normal.	• é difícil desenvolver um bom *framework*. Experiência no domínio da aplicação é necessária quando o mesmo for construído; • a documentação do *framework* é crucial a seu usuário. Se os *frameworks* não provêm documentação, eles provavelmente não serão usados; • a compatibilidade pode ser difícil de manter. Isto deve-se ao fato do *framework* se desenvolver e tornar-se mais maduro com o tempo e a aplicação construída sobre ele dever evoluir junto; • o processo de depuração pode ser complicado por que é difícil distinguir problemas no *framework* de problemas no código da aplicação. Se os problemas estão no *framework*, pode ser impossível para seu usuário encontrá-los; • a generalidade e flexibilidade do *framework* podem ir em contra sua eficiência em uma aplicação particular.

Tabela 4.1: Potenciais pontos fortes e fracos de um *framework*

4.2 *Frameworks* para Interface de Busca/Catalogação de Conteúdo

Os *frameworks* aqui propostos objetivam implementar interfaces baseadas no padrão IMS, onde, dados os tipos de conteúdos a serem buscados/catalogados, provê-se uma interface que permite que as meta-informações do mesmo sejam compreensíveis tanto por um usuário inexperiente neste padrão, como o **professor** quanto para um usuário – teoricamente – conhecedor do mesmo, como o **catalogador**.

4.2.1 Descrição do *Framework* para Interface de Busca (**Professor**)

Este *framework* visa oferecer ao professor uma interface para busca de conteúdos, baseado no padrão IMS, onde o mesmo não tenha que – necessariamente – conhecer este padrão para poder efetuar eficazmente sua pesquisa.

A classe *Interface Content_Type* vai montar a interface de **Tipos de Conteúdos** a pesquisar (figura 4.3), a partir da busca dos tipos de conteúdos existentes, pela classe *Select_Content_Type*. Se não for instanciada, esta classe retorna os *itens Tool, Module, Item e Base* – tipos de conteúdo do padrão IMS, caso contrário, deve ser implementada de acordo com a instância do *framework*[6], retornando os tipos existentes.

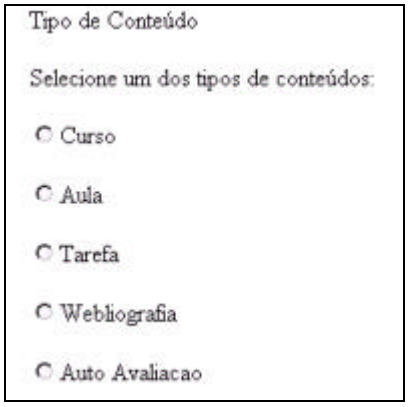


Figura 4.3: Interface de Tipos de Conteúdos a Pesquisar (**Professor**)

Selecionado o conteúdo desejado, a classe *Interface Content_Type_Atrib* vai montar a interface de **Tipos dos Atributos de Pesquisa** (figura 4.4), que contém os grupos pelos quais estão agrupados os conteúdos. Esta interface é montada a partir da busca dos grupos de atributos existentes – de acordo com o tipo selecionado anteriormente (envia-se o tipo de conteúdo e retorna o nome do grupo relacionado) -, pela classe *Select_Content_Type_Atrib*. Se não for instanciada, esta classe retorna os itens do padrão IMS, caso contrário, deve ser implementada de acordo com a instância do *framework*[1], retornando os nomes dos grupos existentes.

[6] Estas classes (*hot-spots* do *framework*) estão instanciadas para o ambiente *AulaNet* e referenciadas, no diagrama de classes (figura 4.8), por identificadores que contém a palavra *AulaNet*.

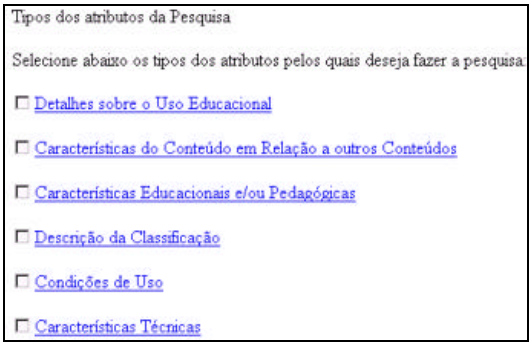


Figura 4.4: Interface de Tipos de Atributos da Pesquisa (**Professor**)

Nesta interface, para cada grupo de atributos encontra-se disponível uma ajuda local (figura 4.5) sobre o mesmo acessível através de um *link* do tipo hipertexto, conforme descrito no capítulo anterior (capítulo 4). As informações que compõe esta ajuda são buscadas através do nome do grupo selecionado, pela classe *Help_Type_Atrib*, que retorna a descrição e atributos do mesmo, informações através das quais a ajuda é formada. Esta classe também é uma classe abstrata e deve ser implementada de acordo com a instância do *framework*[1].

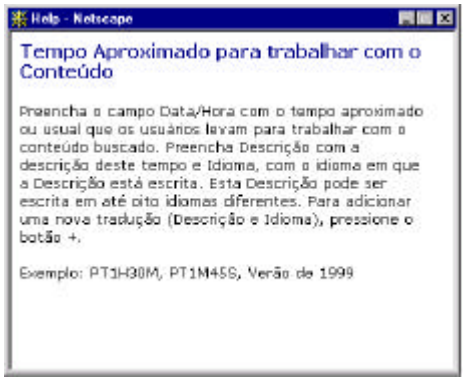


Figura 4.5: Interface da Ajuda para os Tipos de Atributos (**Professor**)

Selecionado o grupo de atributos desejado, a classe *Interface Content_Atribs* vai montar a interface com os atributos correspondentes ao grupo selecionado (figura 4.6). Esta interface é montada a partir da busca destes pela *classe Select_Content_ Atribs* atributos (enviando-se o grupo como parâmetro de entrada). Se não for instanciada, esta classe retorna os itens do padrão IMS, caso contrário, deve ser implementada de acordo com a instância do *framework*[1], retornando os atributos e, para cada atributo, seus tipo, tamanho e multiplicidade (vide capítulo 2), que servem para formar e validar o item de interface referente a cada atributo.

Figura 4.6: Interface de Atributos da Pesquisa (**Professor**)

Nesta interface encontra-se disponível, também, para cada atributo, uma ajuda local sobre o mesmo. As informações que compõe esta ajuda são buscadas através do nome do atributo selecionado, pela classe *Help_Atrib*, que retorna a descrição e exemplos do mesmo, informações através das quais esta ajuda é formada. Esta classe também é uma classe abstrata e deve ser implementada de acordo com a instância do *framework*[1].

O professor preenche os dados que deseja informar para sua pesquisa e a classe *Set_MetaData* gera o arquivo XML correspondente a estas informações e faz a comunicação com o *ContentNet* [11] para a busca do conteúdo desejado, e este, quando efetuada a busca, envia de volta os resultados da busca (arquivos XML com a meta-informação de cada conteúdo).

Após a busca ter sido realizada é montada a interface com os resultados (figura 4.7), através da classe *Interface Content_Result* e, selecionado o item (dentre os conteúdos encontrados) desejado, retorna-se ao *ContentNet* para buscá-lo para o servidor de conteúdos do professor. Também aqui é possível visualizar-se as meta-informações de cada conteúdo resultante da busca, através da classe *Interface Content_MetaData*, que, de acordo com o conteúdo selecionado, mostra o arquivo XML correspondente.

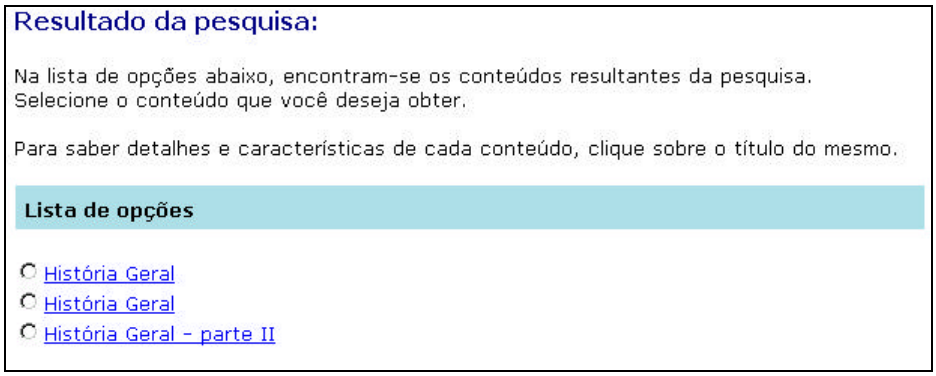


Figura 4.7: Interface de Resultado da Pesquisa (**Professor**)

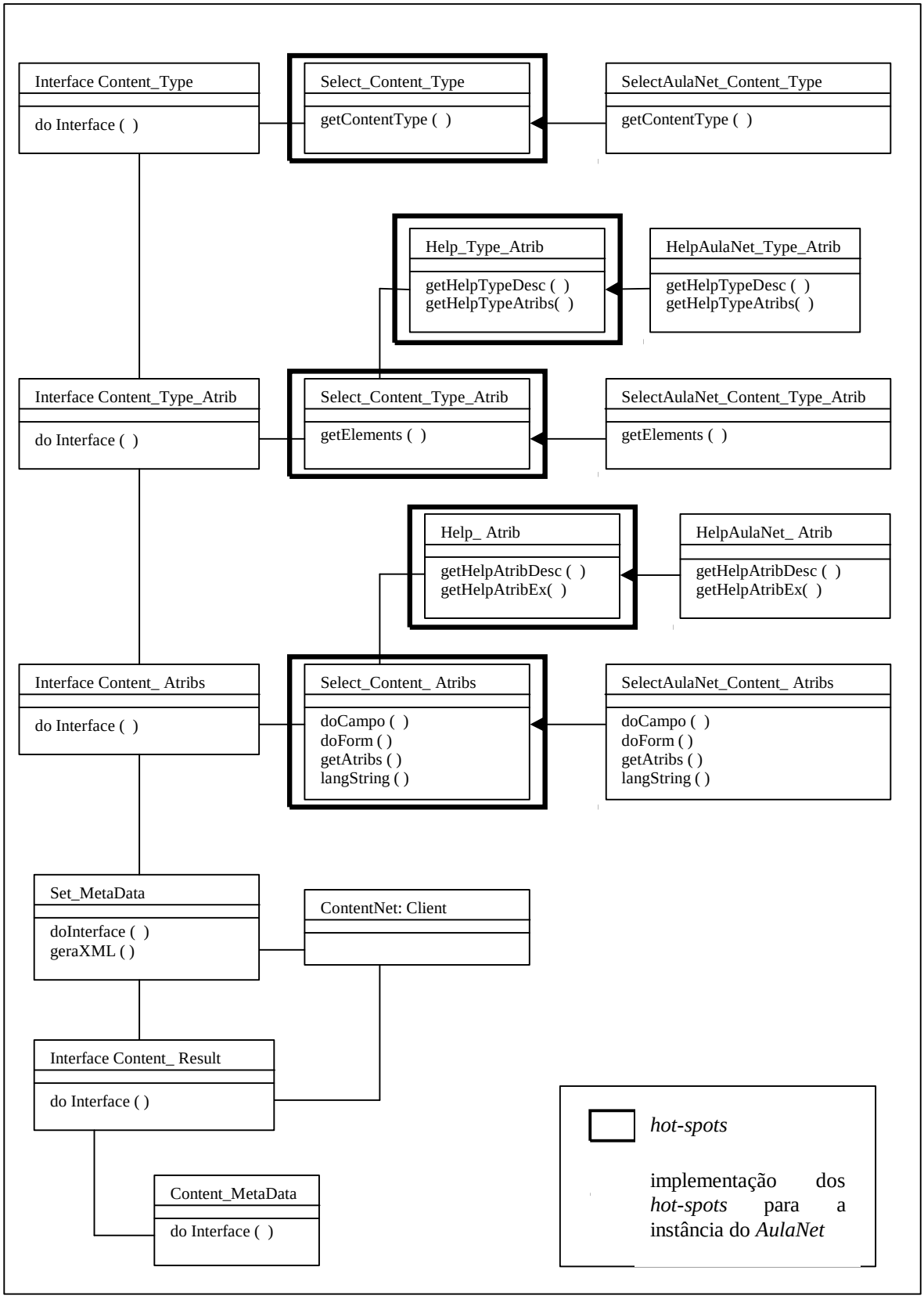


Figura 4.8: Diagrama de Classes (**Professor**)

4.2.2 Descrição do *Framework* para Interface de Catalogação (**Catalogador**)

Este *framework* visa oferecer ao **catalogador** uma interface para catalogação de conteúdo, baseado no padrão IMS.

A classe *Interface Content_Type* vai montar a interface de **Tipos de Conteúdos** a pesquisar (figura 4.3), da mesma forma que no *framework* anterior (item 4.2.1).

Selecionado o conteúdo desejado, a classe *Interface Content_Status* vai montar a interface de seleção de conteúdo a catalogar (figura 4.9), que contém os conteúdos existentes no servidor do **catalogador**, listados de acordo com seu *status* (meta-informações já cadastradas, parcialmente cadastradas ou não cadastradas). Esta interface é montada a partir da busca do conteúdos disponíveis e seus *status* – de acordo com o tipo selecionado anteriormente (envia-se o tipo de conteúdo e retorna os conteúdos e *status* dos mesmos) -, pela classe *Select_Content_Status*. Se não for instanciada, esta classe retorna os itens do padrão IMS, caso contrário, deve ser implementada de acordo com a instância do *framework*[7], retornando os títulos dos conteúdos existentes e seus *status*.

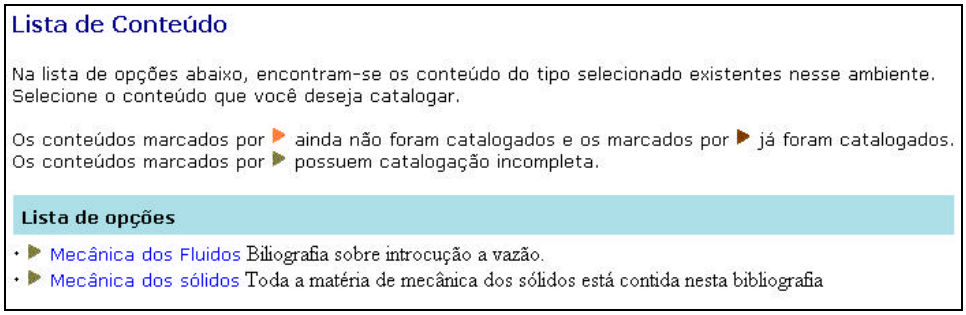


Figura 4.9: Interface de Seleção de Conteúdo a Catalogar (**Catalogador**)

Selecionado o conteúdo que se deseja catalogar, a classe *Interface Content_Atribs* vai montar a interface com todos os atributos, da mesma forma que no framework anterior (item 4.2.1), variando apenas que neste caso não há seleção anterior de grupo e, sim, são mostrados, todos os atributos existentes, para que o **catalogador** preencha os que achar necessários e/ou tiver as meta-informações correspondentes.

O catalogador preenche os dados que deseja catalogar para o conteúdo escolhido e a classe *Set_MetaData* gera o arquivo XML correspondente a estas informações e faz a comunicação com o *ContentNet* [11] para sua catalogação, que também (especificamente para o caso do ambiente *AulaNet*) cadastra estas informações no banco de dados *AulaNet*.

[7] Estas classes (*hot-spots* do *framework*) estão instanciadas para o ambiente *AulaNet* e referenciadas, no diagrama de classes (figura 4.10), por identificadores que contém a palavra *AulaNet*.

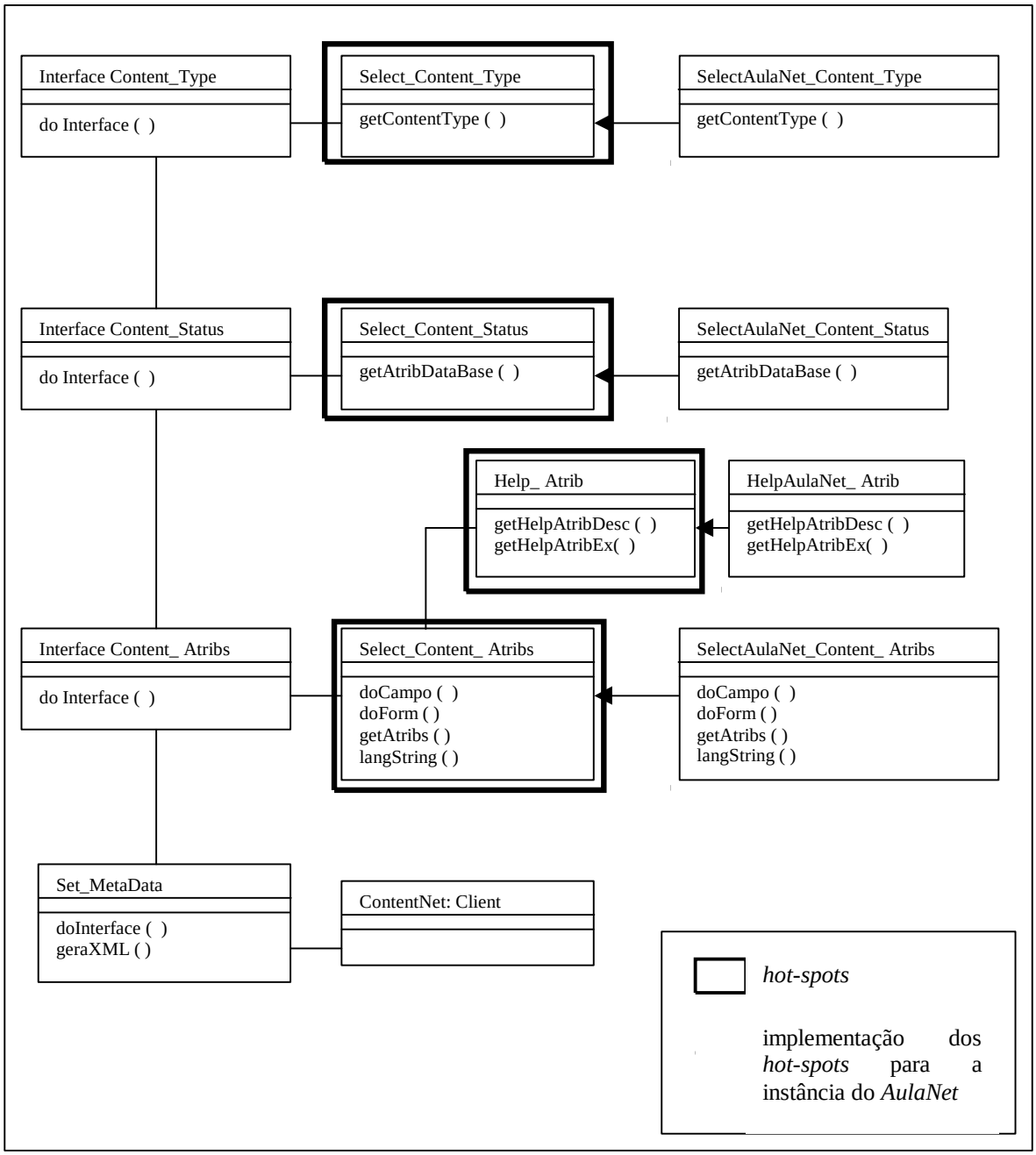


Figura 4.10: Diagrama de Classes (**Catalogador**)

5 CONCLUSÕES/TRABALHOS FUTUROS

O uso do padrão IMS, para interoperabilidade de conteúdos educacionais, vem de encontro a necessidade global de intercâmbio de informações nesta área.

Dada a complexidade do padrão, em termos de seus componentes e formatos, este trabalho propôs *frameworks* para interfaces que visam privilegiar sua utilização por pessoas com pouca ou nenhuma experiência no uso do mesmo, o que amplia sua abrangência.

Dentre as facilidades sugeridas neste trabalho, ressalta-se o uso das ajudas locais, que foram implementadas com a consciência de que – por melhor que fosse a interface - o padrão seria muito complexo mesmo para usuários experientes em seu uso (no caso, o **catalogador**)[8].

Outro item de interface proposto – e não implementado nesta versão do *framework* – foi o botão para adição de campos (item 3.3), para o caso de poder incluir-se várias possibilidades de resposta (atributos com tipo de dado *LangStringType*). Esta alternativa foi muito discutida, principalmente em questões relativas a tempo de acesso/busca (crítica em termos de *Web*) e facilidade de compreensão e utilização destes tipos de dados.

Com a versão atual dos *frameworks* aqui propostos, sua implementação e os testes realizados, foi possível verificar o grau de usabilidade destas interfaces, ficando, como trabalhos futuros, a elaboração de mais testes para aprimoramento destes *frameworks*, principalmente nos itens que não foram abordados neste trabalho (como por exemplo, implementação do botão de adição de campos e ordenação das respostas), a implementação de novas instâncias para validação do mesmo e o acompanhamento da evolução do padrão IMS.

[8] Testes com ambas interfaces foram realizados e encontram-se descritos em [11].

6 REFERÊNCIAS BIBLIOGRÁFICAS

1. Carroll, J. (ed) *Scenario-based Design: Envisioning Work and Technology in System Development*. John Wiley and Sons, New York, NY, 1996.
2. Fayad, M.E.; Schmidt, D.C.; Johnson, R.E. *Building Application Frameworks: object-oriented foundations of framework design.* New York: Wiley Computer Publishing, 1999.
3. IEEE Learning Technology Standards Committee (LTSC). http://ltsc.ieee.org/docs/wg12/scheme.html
4. IEEE Learning Technology Standards Committee (LTSC). *Learning Object Metadata (LOM).* Draft Document v2.1. 25 June 1998 // Document 3 (approved 1999-11-27) . http://ltsc.ieee.org/doc/wg12/LOMdoc2_1.html
5. IMS – Instructional Management Systems. http://www.imsproject.org/
6. Lucena, C. J.; Fuks, H.; Milidiu, R.; Laufer, C.; Blois M.; Choren, R.; Torres, V.; Ferraz, F.; Robichez, G.; Daflon, L. "AulaNet technologies: future trends". *In Proceedings of International Conference on Engineering and Computer Education - ICECE'99*. Rio de Janeiro, 1999.
7. Mattsson, M. *Object-Oriented Frameworks: a survey of methodological issues.* Lund: Department of Computer Science, Lund University, 1996.
8. Pree, W. *Design Patterns for Object-Oriented Software Development*. Reading: Addison-Wesley Publishing Company, 1995.
9. Preece, J. (ed). *Human-Computer Interaction.* Harlow: Addison-Wesley, 1994.
10. Shneidermann, B. *Designing the User interface: strategies for effective human-computer interaction.* Reading: Addison-Wesley, 1998.
11. Silva, V.T. *ContentNet: um framework para interoperabilidade de conteúdos educacionais utilizando a plataforma EDUCAUSE-IMS*. Dissertacao de Mestrado. Rio de Janeiro: DI/PUC-Rio, 2000 (em fase final)
12. Silva, V.T.; Lucena, C.J.P. ContentNet: um framework para interoperabilidade de conteúdos educacionais utilizando a plataforma EDUCAUSE-IMS. *Revista Brasileira de Informática na Educação*. SBC, 2000. (aceito para publicação)
13. Silveira, M.S.; Monteiro, C.C.O.; Barbosa, S.D.J.; de Souza, C.S. The Role of Designer-Generated Scenarios in Developing Web Applications and their Help Systems. *Série Monografias em Ciência da Computação* (MCC09/00). Rio de Janeiro: DI/PUC-Rio, 2000.