4
Metamodelo para Adaptação e Meta-adaptação

4.1.
Modelo Semântico: ASHDM

O metamodelo de referência (seção 3.2) identificou os modelos que devem compor uma arquitetura para adaptação e meta-adaptação e suas principais características. O SHDM possibilita que os esquemas conceituais das aplicações sejam baseados em ontologias definidas para a *Web* Semântica (em OWL ou RDFS). A especificação da função de adaptação ("como"), e de seus parâmetros de entrada ("em função de quê") e saída ("o quê"), em uma linguagem da *Web* Semântica, possibilita a interpretação da própria especificação.

Este trabalho especifica um Modelo Semântico – ASHDM – que sugere uma Arquitetura de Implementação; estende o SHDM, acrescentando características adaptativas aos Modelos de Navegação e de Interface propostos pelo método e agrega Modelos do Usuário (Contexto) e de Adaptação.

Observa-se que em Aroyo et al. (2004), além dos Modelos de Domínio, do Usuário e de Adaptação, distingue-se também o Modelo da Aplicação de modo a separar as decisões sobre por que adaptar. Este modelo contém uma descrição genérica das tarefas do usuário no contexto de uma aplicação em particular. No SHDM, as tarefas são capturadas nos requisitos e não estão representadas explicitamente, sendo utilizadas apenas para derivar os objetos de navegação, ou seja, as visões do DM conforme as tarefas e perfis do usuário.

4.1.1.
Modelo Conceitual

O Modelo do Domínio – que recebe a denominação de Modelo Conceitual no ASHDM, por uma questão de consistência com o SHDM – é uma ontologia qualquer definida para a *Web* Semântica. De acordo com REWERSE (2005), novas ontologias devem ser criadas apenas quando necessário, ou seja, quando ainda não existirem ontologias adequadas para a aplicação que possam ser recuperadas.

A definição do Modelo Conceitual está de acordo com Chepegin et al. (2004) que considera o Modelo do Domínio como uma representação do mundo real, consistindo de conceitos e relações entre eles. A princípio, esta representação é independente da aplicação, ou seja, não diz respeito a problemas, tarefas ou suas soluções.

O Modelo Conceitual em si não sofre nenhum tipo de adaptação. Embora a maneira como o conteúdo é estruturado possa influenciar na adaptação, esta influência apenas se reflete na execução final do sistema. A construção das regras é direcionada pelo modelo proposto, mas a definição das mesmas é feita segundo a ontologia conceitual sendo utilizada. Existem algumas metaregras que podem ser aplicadas sem que o sistema conheça o domínio sendo adaptado.

4.1.1.1. Integração de Domínios

O ASHDM mantém o princípio do SHDM que, assim como o OOHDM, define os objetos navegacionais como visões sobre objetos conceituais, pelo mapeamento da ontologia conceitual para a ontologia navegacional. Conseqüentemente, uma vez que qualquer ontologia definida para a *Web* semântica pode ser utilizada como esquema conceitual, o Modelo Navegacional (descrito na próxima seção) pode ser entendido como um Modelo de Integração, no sentido de se poder criar visões navegacionais sobre ontologias conceituais quaisquer de diversos domínios.

Assim, pode-se dizer que o domínio da aplicação é aberto, pela integração das ontologias no Modelo Navegacional. Observa-se que aqui estamos tratando da questão de modelagem; a forma de acessar estes dados de ontologias distintas, no caso distribuído, não é foco deste trabalho.

4.1.2. Modelo de Navegação

O Modelo de Navegação do ASHDM estende o modelo navegacional para o desenvolvimento de aplicações hipermídia na *Web* Semântica, proposto por Szundy (2004) (seção 2.2.2), pela identificação dos pontos onde a adaptação pode ser incorporada, ou seja, onde podem ser tomadas decisões sobre o que pode ser alterado, para adequar a navegação a determinadas condições decorrentes da utilização do sistema.

Considerando a modelagem navegacional proposta por Szundy (2004), foram identificados os seguintes pontos onde esse processo de decisão pode acontecer:

- Determinação de quais instâncias de uma classe conceitual serão também instâncias da classe navegacional (mapeamento de instâncias);
- Definição dos atributos da classe navegacional;
- Especificação dos valores desses atributos;
- Definição dos elos;
- Seleção dos nós que farão parte de um contexto;
- Determinação do padrão de navegação interna;
- Definição das estruturas de acesso (índices).

Para facilitar o entendimento, a adaptação é especificada e classificada para cada um desses pontos, embora eles não ocorram necessariamente de forma isolada. Alguns exemplos de adaptações combinadas são apresentados. Convém ressaltar que o SHDM já conta com alguns mecanismos – tais como regras de filtragem, regras de seleção e opções de ordenação, criados para possibilitar a definição de visões – que são propícios para a incorporação da adaptação dinâmica.

A classificação é baseada na identificação do quê pode ser adaptado (Tabela 1). A adaptação da apresentação é inerente ao Modelo de Interface (seção 4.1.3). A adaptação da navegação se divide em adaptação da estrutura do conteúdo, relacionada aos nós, e adaptação da topologia do hiperespaço, relacionada aos elos. A adaptação de conteúdo também se dá no mapeamento navegacional, mais especificamente na definição dos atributos, uma vez que é o momento em que se define o objeto de navegação, inclusive o teor do conteúdo.

4.1.2.1. Mapeamento de Instâncias

De acordo com Szundy (2004), a primeira etapa do mapeamento de uma classe navegacional é estabelecer quais instâncias da classe base serão consideradas instâncias da classe navegacional. No caso mais simples, todas as instâncias são mapeadas. Existe a possibilidade de filtragem de instâncias pela utilização de regras que estabelecem as condições para que as instâncias da classe

base sejam mapeadas. A regra de filtragem permite, assim, a definição de nós de navegação (oriundos de classes conceituais) através do mapeamento apenas das instâncias conceituais que satisfaçam determinadas condições.

Em Szundy (2004), estas condições são estáticas, determinadas antes da execução do sistema. Do ponto de vista deste trabalho, interessam as situações em que a filtragem é definida de acordo com a interação que o usuário está tendo com o sistema. Como resultado, tanto a topologia do hiperespaço quanto a estrutura do conteúdo são adaptados, uma vez que o mapeamento condicional ocorre entre os conceitos e o nó como um todo.

Exemplo:

```
Se
      UM.interesse = "IA"
Então
      FILTER: AreaDePesquisa = "IA"
```

4.1.2.2. Definição de Atributos

Em Szundy (2004), já existe um ponto de adaptação implícito na definição de atributos do tipo lista, mais especificamente na escolha do critério de ordenação dos elementos da lista. No caso, a adaptação é da estrutura do conteúdo, uma vez que varia a ordem em que o conteúdo do nó é apresentado.

Exemplo:
No mapeamento da classe navegacional *Professor*, contendo a especificação do atributo *atuação* como uma lista com dois atributos, *tipo* e *detalhe*, pode-se definir que a mesma seja ordenada pelos valores de *tipo*, de acordo com os interesses do usuário.

```
Se
      UM.interesse = <tipo>
Então
      CRITÉRIO_ORDENAÇÃO: <tipo>
```

Identificou-se, porém, que a própria definição de atributos como um todo pode ser adaptada de acordo com a utilização do sistema.

No caso de atributos simples, como o valor do atributo – originado de um atributo da classe conceitual base ou de uma classe conceitual associada – é mapeado em função de um caminho de propriedades conceituais, regras diferentes de mapeamento podem ser utilizadas para promover a adaptação. Além disso, o filtro de exclusão correspondente à lista das propriedades para as quais não se

deseja um mapeamento automático que, originalmente, era especificado antes da execução do sistema também pode ser definido em função de parâmetros determinados durante a utilização do sistema. A adaptação é da estrutura do conteúdo, já que os atributos correspondem, na prática, aos dados recuperados.

Exemplo:
Alguns atributos não são mapeados caso a resolução da tela seja pequena.

Exemplo:
Classe conceitual `Laboratório` possui atributo `aviso_de_segurança`. Este atributo só é mapeado caso seja a primeira vez que o usuário está acessando.

Observação: este é um ponto de meta-adaptação, uma vez que, dependendo do perfil do usuário, pode-se optar por não alterar as condições de exibição do aviso (ou seja, não fazer a adaptação); exibir o aviso de forma esmaecida (adaptação da apresentação / interface do conteúdo) ou, como no exemplo acima, simplesmente não exibir o aviso.

No caso da definição de atributos como âncoras, como ela é feita para a criação de um elo entre instâncias da classe e um índice qualquer ou para representar um elo originado na classe, as adaptações possíveis vão ser tratadas como definição de elos ou como adaptações de apresentação que alteram a aparências das âncoras. Já adaptações passíveis de ocorrer na definição de atributos como índices são consideradas na definição das estruturas de acesso.

4.1.2.3. Especificação dos valores de atributos

O valor de um atributo determina o teor do conteúdo de um nó. Na adaptação dinâmica, o valor do atributo, em vez de ser um mapeamento pré-definido do modelo conceitual, é uma função das características observadas durante a utilização do sistema, promovendo a adaptação da estrutura do conteúdo.

Exemplo (Figura 18):
Para uma classe conceitual `seção`, com atributos `texto_Introdutório`, `texto_Básico`, `texto_Avançado`, existirá uma classe navegacional `seção` com um único atributo `texto`, cujo valor dependerá do conhecimento do usuário. Este valor, por sua vez, pode ser determinado, por exemplo, em função do histórico de navegação e de testes realizados pelo usuário. Caso o usuário esteja acessando o sistema pela primeira vez, ele verá o texto introdutório. Um usuário iniciante verá o texto básico e o especialista, o texto avançado.

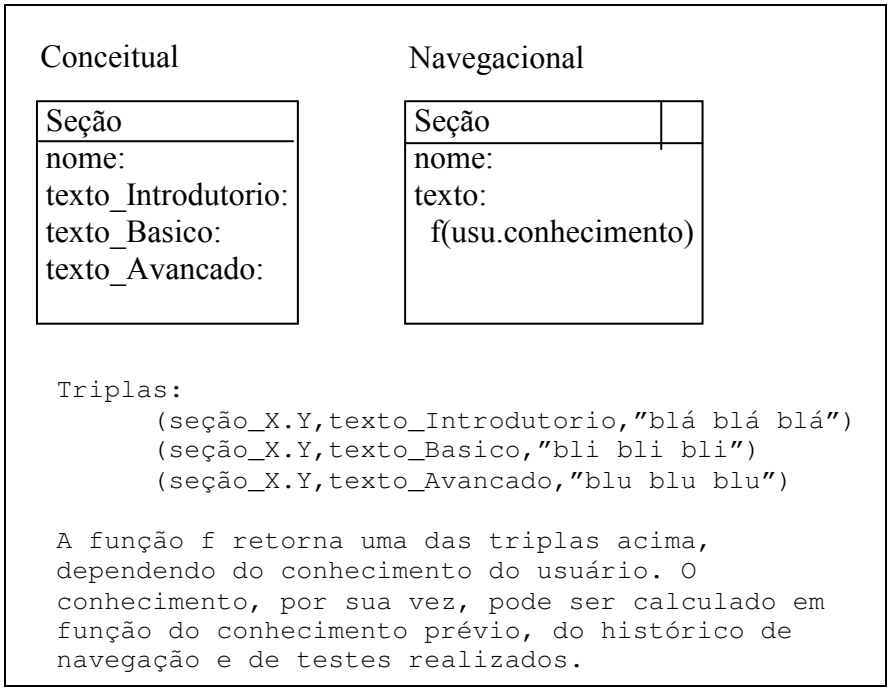


Figura 18 – Exemplo de adaptação dinâmica do conteúdo

4.1.2.4. Definição de elos

Uma das maneiras de se definir um elo é através do mapeamento de propriedades conceituais. O caminho de navegação pode ser alterado pela definição de regras para esse mapeamento, adaptando a topologia do hiperespaço.

Exemplo:
Para a apresentação de um conceito, pode ser definido um elo para um texto explicativo ou para um exemplo, dependendo do estilo de aprendizagem.

Elos também podem ser definidos pela declaração de âncoras em estruturas de acesso ou classes navegacionais e pela definição de padrões intracontextuais. Em todos os casos, como os elos representam as relações disponíveis para navegação, regras para a definição dos mesmos determinam (adaptam) a topologia do hiperespaço.

4.1.2.5. Seleção de nós

Um contexto é o resultado da união de nós escolhidos por um conjunto de regras que determinam o critério de seleção. Cada regra seleciona nós de um determinado tipo (exemplo: todos os alunos) e que, opcionalmente, satisfaçam alguma condição (como, por exemplo, "publicações posteriores a 2003").

Originalmente, o critério de seleção dos nós é estático. Na adaptação dinâmica, a escolha dos elementos do contexto é definida durante a utilização do sistema. A seleção se reflete na navegação como um todo, caracterizando, portanto, uma adaptação da estrutura do conteúdo e da topologia do hiperespaço.

Exemplo:
Caso o estilo de aprendizagem do aluno seja mais teórico, será selecionado o contexto "tópicos por seção"; caso contrário, o contexto escolhido será "tópicos por exercício".

Exemplo:
Considerando-se como publicações recentes aquelas publicadas no último ano, uma regra de seleção dos elementos do contexto poderia ser:

```
Se
      UM.interesse = "Publicações recentes"
Então
      SELEÇÃO:
      tipo: publicação; condição: anoPublicacao >= ano_atual -1
```

Vale notar que, no caso das regras de filtragem (seção 4.1.2.1), define-se quais informações serão mostradas através da criação de uma nova classe navegacional contendo apenas as instâncias da classe conceitual que satisfazem aos critérios estabelecidos. Já a regra de seleção de elementos do contexto está definindo como as informações serão acessadas. No exemplo acima, as publicações serão navegadas dentro de um contexto de publicações recentes. Caso haja interseção com outro contexto (conforme definido em Szundy (2004)), pode-se navegar de um para outro.

4.1.2.6. Determinação do padrão de navegação interna

Um contexto também é definido pela especificação de um padrão de navegação interna que possibilita a exploração de todos os elementos de um contexto, a partir de qualquer um deles.

A definição desse padrão é baseada nos seguintes tipos de navegação intracontextual (aplicados diretamente ou combinados): livre, circular, seqüencial ou por índice (Szundy, 2004). Tal definição pode ser condicional, possibilitando a adaptação da estrutura de navegação, uma vez que altera a maneira como os nós se organizam para a navegação; e da topologia do hiperespaço, por influenciar a definição dos elos.

Exemplo:
Considerando-se um contexto que recupere poucos nós (”alunos com mais de 20 publicações”), a navegação livre pode ser mais adequada.

De acordo com Szundy (2004), a ordem em que os elementos são explorados pode ser determinada pelo usuário, com base nos valores dos atributos dos nós. Ou seja, a ordenação opcional é um ponto natural de adaptação que altera a navegação como um todo: nós e elos.

Exemplo:
Considerando-se a navegação por um contexto com todos os alunos, caso não se conheça os objetivos do usuário, pode-se definir uma ordenação padrão por ordem alfabética. Por outro lado, o usuário pode estar interessado em ver os alunos de acordo com as áreas de pesquisa em que eles atuam:

```
Se
      UM.objetivo = <desconhecido>
Então
      ORDENAÇÃO: <Alfa>

Se
      UM.objetivo = ”acessar alunos por área de pesquisa”
Então
      ORDENAÇÃO: <AreaDePesquisa>
```

4.1.2.7. Definição dos índices

De acordo com Szundy (2004), “o método SHDM permite a definição de quatro tipos de índices que se distinguem em função da origem de seus dados”. São eles: índice derivado de consulta, índice derivado de contexto; índice arbitrário e índice facetado.

A adaptação depende do tipo do índice, podendo acontecer na definição da consulta, na escolha dos dados do contexto, na declaração explícita das entradas ou nos valores selecionados na composição de facetas. Em todos os casos, a topologia do hiperespaço é alterada.

4.1.2.8. Outros exemplos

- Mapeamento de instâncias e definição de atributos:

Classe conceitual `Aluno` possui atributo `Dependências`. Este atributo só será mostrado se o usuário que estiver acessando for o orientador do aluno.

- Mapeamento de instâncias e definição de elos:

`Exercício possui Gabarito`, mas este só é navegável se `Exercício` já tiver sido resolvido pelo usuário. Ou seja, o *status* do exercício no UM determinará se será feito o mapeamento da propriedade `possui` entre a instância de `exercício` e a instância de `gabarito`, resultando em um elo que permitirá a navegação pelo `gabarito`.

- Seleção de elementos do contexto navegacional e determinação do padrão de navegação.

Considerando-se o contexto como "`todos os capítulos de um determinado módulo`" e o padrão de navegação seqüencial, se o aluno tiver terminado o estudo do `capitulo_1`, o botão "Próximo" pode levá-lo ao nó `capitulo_1_extendido` ou ao nó `capitulo_2`, conforme o nível de conhecimento adquirido. Este nível de conhecimento, por sua vez, pode ser calculado em função do histórico de navegação e da funcionalidade da aplicação.

- Definição de elos

Um elo para a solução de um problema só será criado caso o mesmo esteja sendo utilizado como exemplo

- Definição de índices

Elos para exercícios vêm antes ou depois de elos para a teoria de acordo com o estilo de aprendizagem do usuário.

4.1.3. Modelo de Interface

O Modelo de Apresentação no ASHDM é denominado Modelo de Interface, também por uma questão de coerência com o SHDM. Ele continua sendo definido pela Ontologia de *Widgets* abstratos, pelo mapeamento do abstrato para o concreto e pelos *widgets* concretos, como no SHDM (seção 2.2.3). Entretanto, como será visto, o mapeamento e a definição do *layout* são estendidos para a inclusão de adaptação no sistema; já uma extensão para a incorporação de adaptação na interface abstrata é uma questão em aberto, objeto de trabalhos futuros.

4.1.3.1. Ontologia de *Widgets* Abstratos

A essência da arquitetura de *widgets* abstratos é a definição dos elementos "ativadores" (reagem a interações do usuário), "exibidores" (exibem saídas do sistema) e "capturadores" (capturam entradas do usuário).

A interface abstrata – instância dessa ontologia – é definida pela composição dos elementos acima e não é passível de adaptação, já que a decisão sobre se o elemento é um "ativador", um "exibidor" ou um "capturador" e a determinação de qual o tipo de informação capturada não é função das características do usuário, ou da utilização do sistema, mas é feita pelo projetista que se baseia na análise de requisitos e no modelo navegacional. A forma como os elementos são agrupados, por sua vez, poderia ser adaptada. Embora sendo possível, este tipo de adaptação não parece ser muito útil.

Assim, assume-se que, de uma maneira geral, não há adaptação na Ontologia de *Widgets* Abstratos, mas essa questão é de fato vista como objeto de estudos futuros.

4.1.3.2. Mapeamento

No SHDM, a escolha de para qual elemento concreto o elemento abstrato será mapeado, dentre os que satisfazem à regra de consistência, é uma decisão do projetista da interface. Propõe-se que regras de adaptação sejam incluídas no mapeamento para que esta escolha possa ser feita dinamicamente, em função de parâmetros de utilização do sistema.

Dentre todos os mapeamentos possíveis, algumas escolhas são unicamente em função do projeto do sistema, não sendo adaptáveis. De uma maneira geral, as possibilidades de adaptação no mapeamento são as seguintes:

- `ElementExhibitor` pode ser mapeado para `Image` ou `Label`;

Exemplo: as instruções de montagem de uma peça podem aparecer como um diagrama ou como um texto explicativo, de acordo com o perfil do usuário

- `DiscreteGroup` pode ser mapeado para `CheckBox`, `CheckBoxAction`; `ComboBox`, `ComboBoxAction`, `ComboBoxTarget`; `RadioButton`, `RadioButtonAction`, `RadioButtonTarget` ou para um elemento composto por `Link`;

Exemplo: para a seleção de publicações por aluno, o mapeamento poderia ser para uma estrutura tipo `ComboBox`, caso o perfil do usuário fosse leitura *on-line* e o aluno tivesse mais de 10 publicações; para um elemento `Link`, se o número de publicações fosse menor ou para uma estrutura `RadioButton`, se o objetivo fosse imprimir. Se a preferência do usuário fosse por salvar as publicações selecionadas, o mapeamento poderia ser para uma estrutura tipo `CheckBox`

- `SingleChoice` pode ser mapeado para `RadioButton`, `RadioButtonAction`, `RadioButtonTarget`; `ComboBox`; `ComboBoxAction`; `ComboBoxTarget` ou para um elemento composto por `Link`.

Exemplo: a lista de orientandos de um determinado professor pode ser mapeada para `RadioButtonTarget`, caso o professor possua menos de 5 orientandos ou para `ComboBoxTarget`, caso contrário. O mapeamento é feito para o *widget* concreto que já embute um *link* na definição de cada item da lista porque se deseja habilitar a navegação para o orientando. Outra possibilidade é o mapeamento para um elemento composto por `Link`, no caso do professor possuir entre 5 e 10 orientandos.

Embora `SimpleActivator` possa ser mapeado para `Link` ou `Button`, a escolha entre estas duas alternativas parece estar relacionada ao projeto do sistema, não sendo considerada como adaptação. Analogamente, a decisão entre mapear `ArbitraryValue` para `TextArea` ou para `TextBox` parece estar relacionada ao tipo de entrada que se quer capturar, o que não depende das preferências do usuário ou utilização do sistema. Entretanto, como o modelo proposto deverá ser extensível, caso se deseje representar a entrada de dados como `TextArea` quando, hipoteticamente, o resultado esteja sendo enviado para um celular, ou como `TextBox`, caso contrário, bastaria incluir uma condição associada ao mapeamento de `ArbitraryValue`.

4.1.3.3. *Layout*

As seguintes propriedades da Ontologia Abstrata, definidas em Moura (2004), podem ter condições associadas que adaptem o *layout* da interface, de acordo com parâmetros de utilização do sistema:

- `blockElement`: indica as *tags* HTML e classes CSS que serão usadas para a tradução de um elemento específico. Esta propriedade é opcional para todas as subclasses de `AbstractInterfaceElement`;

Exemplo:
De acordo com a resolução da tela, pode-se utilizar a tag <h1> ou a tag <h6> para exibir um título da página

- `compositionTag`: indica uma *tag* HTML. Esta propriedade pertence à classe `CompositeInterfaceElement`. Utilizada quando a propriedade `isRepeated` possui valor *true*, sua função é indicar qual a *tag* HTML que irá separar os

elementos que se repetem.

Exemplo:
Dependendo do espaço disponível para exibição, os elementos de repetição podem ser separados pelo uso das tags
 ou <p>.

Além disso, as classes CSS podem ser definidas de acordo com o *layout* desejado. A implementação da metáfora do "sinal de trânsito" para a anotação de elos, por exemplo, pode ser feita com CSS, com uma classe para cada valor do sinal e usando CSS para definir que quando o sinal for verde, o elo aparece em verde e assim por diante. Desta forma, na adaptação muda o CSS, mas não a categoria semântica. Ou seja, a categoria semântica fica refletida no *layout*.

Os atributos CSS permitem formatar diversas características de interface, entre elas, o fundo (*background*), as bordas, as cores, a fonte (tamanho, estilo, peso, etc.), espaçamento entre letras, altura das linhas, estilo do texto. Assim, fica fácil disponibilizar diferentes estilos para os usuários de acordo com suas características. É necessário apenas que as regras de adaptação alterem as propriedades CSS.

A utilização de CSS permite adaptações ainda mais complexas de *layout* que fogem ao escopo deste trabalho, mas seria possível, por exemplo, posicionar alertas de segurança na parte superior ou inferior da tela caso fosse a primeira vez ou não que o alerta estivesse sendo mostrado.

4.1.4. Modelo do Contexto de Adaptação

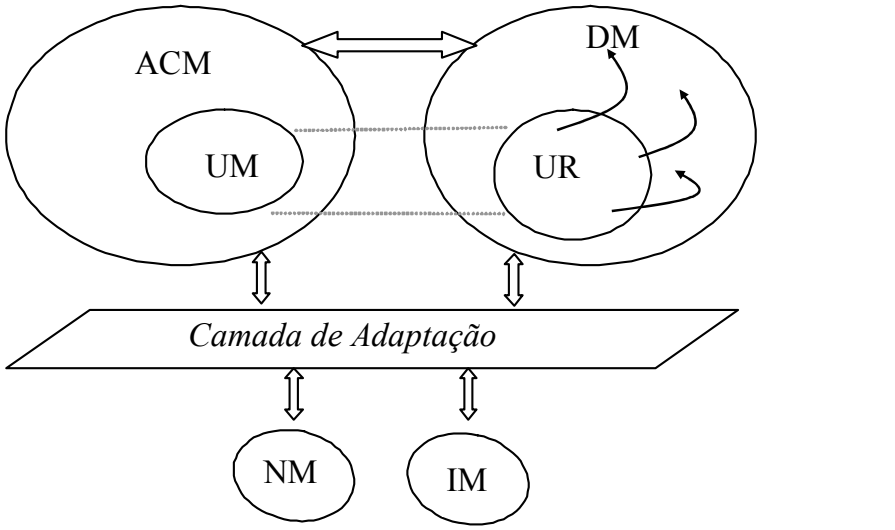
O ASHDM inova em relação aos modelos tradicionais para hipermídia adaptativa propondo que o Modelo do Usuário seja considerado como um componente de um Modelo do Contexto de Adaptação (*Adaptation Context Model*, ACM).

Em geral, considera-se que o propósito de uma aplicação hipermídia adaptativa é refletir algumas características do usuário em um Modelo de Usuário e utilizar este modelo para adaptar diversos aspectos do sistema ao seu usuário (Kuruc, 2005). Entretanto, embora este seja o enfoque habitual, na realidade o usuário é apenas parte – em geral fundamental, é verdade – de todo um contexto de adaptação.

Por conta das aplicações que armazenam informações sobre seus usuários,

mesmo não sendo adaptativas, sugere-se a existência de uma Representação do Usuário (*User Representation*, UR) como parte do Modelo do Domínio. Por exemplo, no domínio de comércio eletrônico, é típico encontrar-se uma classe "Cliente" no modelo de domínio da aplicação. Essa representação pode ser entendida como uma subclasse do Modelo do Usuário.

A Figura 19 mostra a relação entre esses modelos e a influência deles nos Modelos de Navegação e de Interface.



ACM (*Adaptation Context Model*): Modelo do Contexto de Adaptação
UM (*User Model*): Modelo do Usuário
DM (*Domain Model*): Modelo do Domínio
UR (*User Representation*): Representação do usuário
NM (*Navigation Model*): Modelo de Navegação
IM (*Interface Model*): Modelo de Interface

Figura 19 – Relação do usuário com o contexto, o domínio e outros modelos

Para uma melhor compreensão da figura acima, a interação entre seus elementos é analisada a partir de exemplos em um cenário de um museu virtual que, além de exposições permanentes, promove diversos eventos. Inicialmente, aborda-se a situação em que é necessário acrescentar adaptação à aplicação.

Primeiramente, considera-se uma adaptação que não é sensível ao usuário, ou seja, o comportamento do sistema independe de quem o está usando, mudando apenas de acordo com o ambiente de uso. No cenário proposto, as alas do museu, incluindo eventos que possam estar acontecendo, são mostradas em tempo real em diversos tipos de dispositivo, alguns com banda passante variável. Assim, a informação pode ser mostrada como um vídeo ou como imagens estáticas, conforme a banda disponível no momento. A apresentação também é adaptada ao

tipo de vídeo, variando a quantidade de informação visual e textual de acordo com o tamanho do mesmo. Um dos dispositivos poderia, inclusive, ser um celular, mas note-se que a adaptação não é influenciada por quem está vendo as informações.

Por conseguinte, para incluir tais características adaptativas, é necessário estender o modelo da aplicação com um Modelo do Contexto de Adaptação capaz de representar os elementos do contexto aos quais se deseja ser sensível como, por exemplo, informações sobre dispositivos e banda passante. O usuário não precisa ser identificado nem modelado. A partir das informações do contexto e do domínio, e das inter-relações, se existirem, são instanciados os modelos (adaptados) de navegação e de interface, produzindo-se o resultado final que vai ser mostrado.

Na situação mais usual, a adaptação é feita também em função do usuário. Os dados sobre o usuário podem ser obtidos através de formulários eletrônicos ou recuperados de algum tipo de base de dados, inclusive de outras aplicações. Essas informações podem ser independentes do domínio ou relacionadas a ele.

No cenário considerado, caso o dispositivo seja um celular, um computador ou algum outro que permita a interação do usuário com a aplicação, a seleção das alas a serem mostradas pode levar em consideração características do usuário independentes do domínio, tais como, sua idade e suas preferências em relação à forma de apresentação dos dados; ou dependentes do domínio como, por exemplo, o interesse dele pelas categorias dos objetos do museu e as alas que ele se interessou em explorar.

Em vista disso, a aplicação precisa considerar informações sobre o usuário. Quando a adaptação acontece unicamente a cada sessão, tais informações não necessitam ser armazenadas. Este seria o caso de um transeunte acessando a partir de um centro de turismo. Entretanto, a situação mais comum – e interessante – envolve a utilização de um Modelo do Usuário para armazenar as informações sobre a pessoa que está utilizando a aplicação e, ainda, as relações dela com o domínio. Como visto, esse modelo é usual em aplicações adaptativas.

Assim, a adaptação é obtida ainda da mesma forma, mas informações sobre o usuário fazem, agora, parte dos elementos aos quais a adaptação é sensível, sendo incluídos no ACM usado para estender a aplicação.

A incorporação da adaptação pela simples extensão do modelo da aplicação através de um ACM e de suas relações com o domínio é possível quando a

aplicação não possui a noção de usuário. Existem casos, porém, em que, mesmo não havendo adaptação, o domínio tem informações sobre o usuário, ou seja, o domínio possui uma Representação do Usuário. Pode-se usar como exemplo uma aplicação que permita ao usuário registrar seus "favoritos" (*bookmark*) ou que use dados cadastrais do usuário para fins promocionais. Como no esquema proposto os parâmetros de adaptação são modelados no contexto, incluídos aí os relativos ao usuário (UM), é necessário fazer uma integração entre o UM e o UR, para evitar problemas de duplicação da informação e possíveis inconsistências que poderiam ocorrer, caso os dois modelos fossem tratados independentemente.

Tem-se, portanto, um Modelo de Domínio com um conjunto de classes que representam o usuário e deseja-se que o Modelo do Usuário para a adaptação seja uma junção entre as informações do UR e as do próprio UM. Os dados armazenados no UR podem coincidir no todo ou em parte com os do UM, ou podem ser necessários para estendê-lo. Logo, a integração de ambos deve ser feita como uma unificação do UM e do UR que resulta em uma composição das informações sobre o usuário e das relações destas com o domínio. A integração destes dados com as demais informações do ACM e do DM é obtida como uma "visão" (*view*) sobre os modelos, Essa visão é análoga à visão navegacional em que a partir de atributos conceituais cria-se um nó virtual. Aqui é criado um contexto de adaptação virtual. A Figura 20 esquematiza a unificação das informações sobre o usuário.

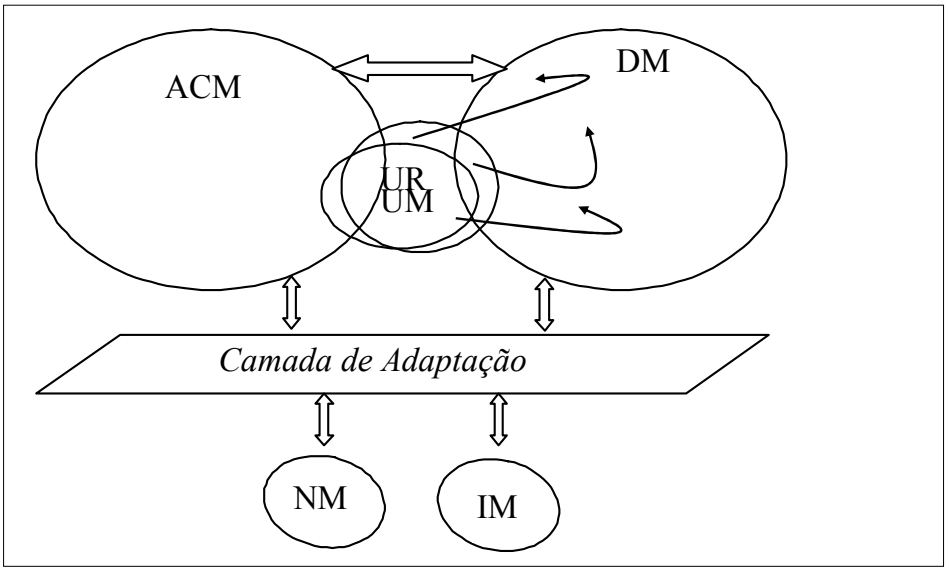


Figura 20 – Unificação do UM e da Representação do Usuário

No cenário-exemplo, o domínio da aplicação tem uma classe `usuário` com os atributos: `ID (identificador)`, `nome`, `endereço`, `data_nascimento` e

`favoritos` (utilizado para registrar as alas destacadas pelo usuário). O ACM contém as informações sobre o tipo de vídeo (*display*) e um componente UM que armazena as preferências de apresentação. A visão do contexto de adaptação possui os atributos `idade`, `favoritos`, `preferências` e `display`.

Quando as aplicações já são adaptativas, é usual a utilização de um Modelo do Usuário, cuja função é representar diversas características do usuário como conhecimento, preferências, interesses, tarefas e objetivos (Kuruc, 2005). As relações que existem entre o usuário e o domínio são, em geral, representadas em um modelo de sobreposição (*overlay model*) onde para cada conceito do DM existe um conceito correspondente no UM. Em aplicações educativas, por exemplo, pode-se representar o quanto o usuário sabe de determinado assunto. No comércio eletrônico, o quanto o usuário conhece sobre certo produto ou se ele já comprou este produto. A representação do perfil do usuário em relação ao conhecimento prévio sobre o domínio (exemplo: novato, intermediário, experiente), sobre outras áreas ou relativo às suas preferências, estilo de aprendizagem, etc. muitas vezes é armazenada em um modelo de estereótipos (*stereotype model*), que também pode ser usado para classificar um novo usuário, estabelecendo os valores iniciais para o modelo de sobreposição. Pela sua importância, o UM será estudado mais detalhadamente na próxima seção

A Figura 21 apresenta o último exemplo descrito, com a inclusão no domínio da aplicação das informações sobre quais alas do museu já foram visitadas pelo usuário (`histórico`). O UM, existente previamente, também contém a identificação e o nome do usuário, além de seu estilo cognitivo e preferências. Considerando-se uma adaptação feita em função da idade do usuário, do interesse dele pelas categorias dos objetos do museu, das alas já visitadas e das características do vídeo, os atributos da visão unificada do contexto de adaptação, mostrada na Figura 22, são `idade`, `categorias_de_interesse`, `histórico` e `display`. As categorias de interesse poderiam, por exemplo, ser inferidas a partir da idade do usuário, ou ser informadas por ele explicitamente..

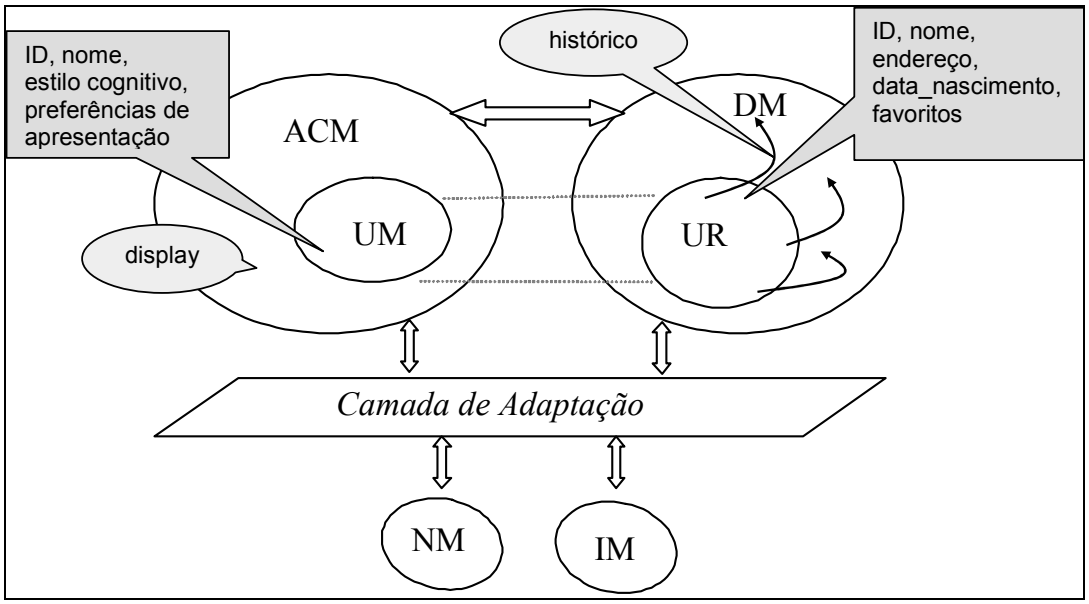


Figura 21 – Exemplos de informações armazenadas nos diversos modelos

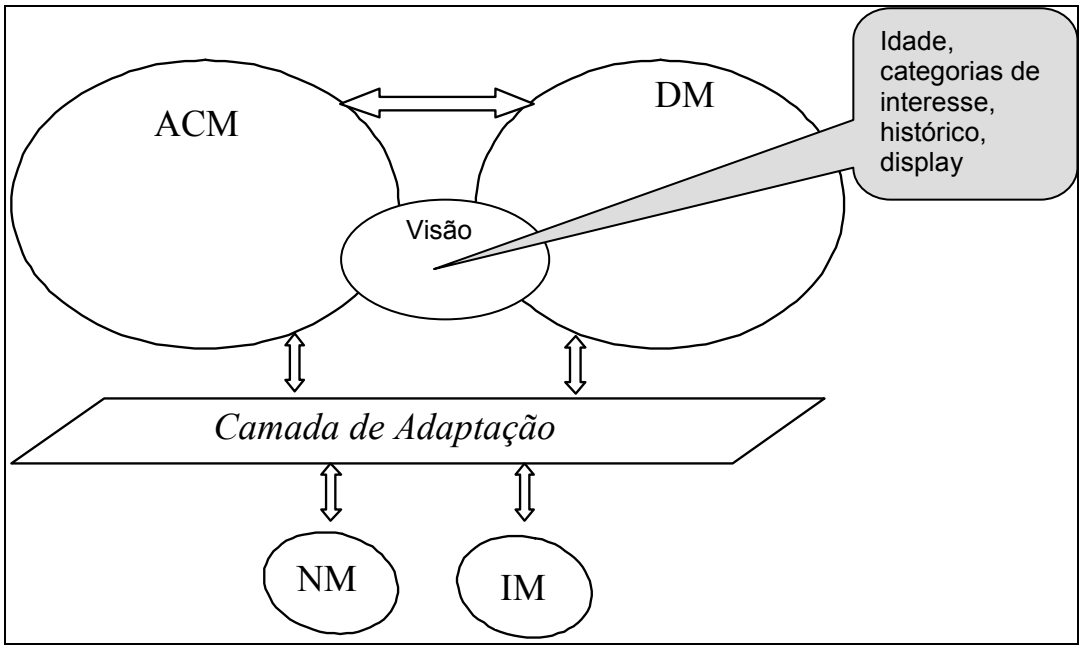


Figura 22 – Exemplo das informações como uma visão sobre os modelos

Em resumo, o contexto de adaptação engloba todos os elementos aos quais a adaptação é sensível como, por exemplo, características de dispositivos (vídeo, processador, teclado, mouse, etc.); localização; largura de banda; sistema operacional; navegador e, inclusive, o próprio usuário. Pelo seu tradicional destaque no processo de adaptação, o usuário é modelado de forma diferenciada.

4.1.4.1. Esquematização do Modelo do Usuário

Potencialmente, o Modelo do Usuário pode conter centenas ou até mesmo milhares de elementos. A utilização de uma ontologia ajuda a estruturá-lo, mas

persiste a questão sobre quais informações vão fazer parte dela.

Para que o UM possa ser compartilhado por diversas aplicações hipermídia adaptativas (seção 2.3.1), é necessário que sua semântica – ou seja, o significado dos atributos e a interpretação de seus valores – esteja explicitada. É preciso, ainda, fazer a distinção entre a representação da parte do UM que é dependente do domínio e da parte independente, isto é, comum a todos os domínios (Kuruc, 2005). Entretanto, a própria definição de quais informações devem fazer parte do UM (e quais fazem parte do DM) é uma questão em aberto.

Radicalizando, pode-se considerar que, à exceção da identificação do usuário, todas as outras informações fazem parte do domínio. Afinal, o endereço pode influenciar na sugestão de lojas mais próximas; a idade na oferta de produtos de interesse; o local de nascimento na escolha de um vocabulário mais adequado e assim por diante. Já outras informações, como aquelas sobre realizações acadêmicas propostas por modelos tipo o IMS LIP (seção 2.3.3), já são, por natureza, pertinentes apenas ao domínio (no caso, o de educação).

Sob o ponto de vista da adaptação propriamente dita, pouco importa quais dados estão armazenados em um ou em outro modelo, já que a função de adaptação considera o conjunto dos parâmetros de entrada. Entretanto, como algumas informações relativas ao usuário são sempre as mesmas, independentemente do domínio, a estruturação delas em um modelo próprio, modular, permite o reuso em diversos domínios, evitando que o UM tenha que ser repetidamente inicializado e alimentado.

Grosso modo, considera-se que informações úteis para qualquer aplicação, independentemente da semântica específica do domínio, devem fazer parte do núcleo do UM e as demais, do DM. Todavia, uma vez que a ontologia do domínio é qualquer, além da ontologia básica, que representa as características do usuário independentes do domínio, o UM precisa ser complementado com as informações do usuário que estejam embutidas na ontologia do domínio sendo usada. Como já visto, são construídas visões para a integração dessas informações.

Considerando-se o exposto até aqui, sugere-se a seguinte estrutura modular para o UM, de modo facilitar o reuso:

1. Informações Independentes do Domínio (Figura 23);
2. Modelo de Sobreposição;
3. Informações Dependentes do Domínio.

A proposta baseia-se na ontologia GUMO que é bastante interessante do ponto de vista de estruturação dos aspectos básicos do usuário. Contudo, a falta de uma política para inclusão de novas classes e instâncias acaba por comprometer sua eficiência. Observa-se que, assim como nem todas as informações precisam ser consideradas por algumas aplicações, outros dados deverão poder ser adicionados para atender a particularidades de outras.

1. Informações Independentes do Domínio
   1.1. Estáticas
      1.1.1. Contato
         Nome Completo
            Nome de Família
            Primeiro Nome
            Nomes Intermediários
         Endereço
            Logradouro
            Número
            Complemento
            Bairro
            CEP
            Cidade
            Estado
            País
         Telefone
            Residencial
            Comercial
            Celular
            Fax
         Web
            E-mail
            Homepage
      1.1.2. Demográficas
         Sexo
         Data de nascimento
         Naturalidade
         Nacionalidade
      1.1.3. Personalidade
      1.1.4. Deficiências
   1.2. Dinâmicas
      1.2.1. Preferências
         Dispositivo Preferencial de Entrada
         Dispositivo Preferencial de Saída
         Idioma principal
         Idioma alternativo
      1.2.2. Estado Emocional
      1.2.3. Estado Fisiológico
      1.2.4. Estado Mental
      1.2.5. Habilidades
         Digitação
         Leitura
         Escrita
         Oral
2. Modelo de Sobreposição
3. Informações Dependentes do Domínio

Figura 23 – Estrutura do UM: Informações Independentes do Domínio

As informações estáticas são relativas à customização, já que não mudam durante a utilização do sistema. Informações de contato, em geral, são usadas apenas pela parte "administrativa" do sistema. Informações demográficas e personalidade podem interessar a todos os domínios, embora nem todas as aplicações tenham mecanismos para lidar com elas.

Diferentemente do GUMO (seção 2.3.5) que as modela, as capacidades referentes aos sentidos (visão, audição, tato, olfato, paladar) e mais a capacidade de andar e de falar são assumidas como verdadeiras. A falta de uma ou mais delas é caracterizada como "Deficiências".

As informações sobre o estado emocional (alegria, ansiedade, medo, etc.), o fisiológico (temperatura, pressão sanguínea, dilatação da pupila, etc.) e o mental (estresse, trauma, depressão, etc.), além das habilidades de digitação, leitura, escrita e oral que influenciam na utilização do sistema, interessam, teoricamente, a todos os domínios, embora apenas aplicações específicas as usem. Poder-se-ia argumentar que a habilidade oral deveria ficar, preferencialmente, como dependente do domínio já que depende do idioma. Entretanto, esse tipo de discussão não cabe no escopo da tese e, como já observado, em termos de adaptação é indiferente em que categoria a informação está representada.

O Modelo do Usuário está, em geral, relacionado ao Modelo do Domínio, através de uma relação com o sentido de "conhecer", ou seja, o usuário tem algum grau de experiência prévia (que pode até ser nulo) com o domínio. Essa informação é principalmente utilizada na área educacional, como forma de adequar o conteúdo ao conhecimento do aluno. Contudo, em outros domínios essa relação também se aplica. Por exemplo, o sistema pode avaliar se o conhecimento do usuário sobre algum produto é técnico ou leigo para decidir o vocabulário a ser utilizado; em uma visita virtual a um museu, a sugestão do caminho a percorrer pode ser feita em função de caminhos percorridos em visitas anteriores e assim por diante. Esse tipo de informação pode ser armazenado em um Modelo de Sobreposição. Cabe lembrar que o uso deste modelo não é obrigatório. Em uma loja virtual, a informação sobre se um produto já foi comprado pode ser ignorada por se considerar que uma nova compra para presentear pode ser feita, por exemplo. Até mesmo em aplicações educativas o projetista pode julgar mais adequado que o conteúdo seja sempre percorrido, baseando a adaptação em outros parâmetros.

Informações dependentes do domínio são do tipo "avaliação de desempenho", "estilo musical preferido", "salário" e outras características, preferências e informações do usuário que não interessam a todos os domínios. Elas podem ser modeladas, por exemplo, extraindo-se de ontologias do tipo IMS LIP (seção 2.3.3) todas as classes que dizem respeito ao usuário, mas que não sejam do tipo "conhecer". Isto é possível porque a utilização de metamodelos e ontologias viabiliza a definição de um tipo de metaregra que atua sobre os esquemas e não sobre as instâncias, ou seja, regras do tipo "recupere todos os recursos que têm propriedade diferente de 'conhece' cujo domínio é da classe usuário".

A estrutura proposta não pretende ser completa, mas apenas sugerir uma maneira de organizar o UM. Requisitos de modelagem de usuário ligados à gerência de contexto e à computação móvel (Rolins, 2003) não foram explorados neste trabalho. Embora muitas questões permaneçam em aberto, demonstra-se que, com a estrutura apresentada, um servidor de UM[17] poderia disponibilizar para as aplicações o núcleo do UM e mais outros módulos, de acordo com o domínio e o tipo de aplicação, possibilitando a reutilização de informações, sem a necessidade de inicializar o modelo a cada vez. Além disto, a estrutura proposta permite a definição de metaregras que organizam as regras de adaptação de acordo com as propriedades do modelo (exemplo: se UM tiver a propriedade "nacionalidade", inclua notícias sobre o "país de origem").

Requisitos propostos em Kay et al. (2002) são atendidos por essa estrutura modular do UM: (a) diferentes visões do UM podem estar disponíveis para cada aplicação adaptativa. Assim, uma aplicação vai ter definidos para ela apenas os componentes do UM dos quais ela necessita; (b) um usuário pode determinar que partes do UM estão acessíveis para que aplicações; (c) usuários podem querer manter a maior parte do seu modelo pessoal sob controle em um sistema local, escolhendo quais informações serão armazenadas em servidores externos, fora do controle direto do próprio usuário.

A possibilidade de integração entre as ontologias é exemplificada na Figura 24 (Dolog & Nejdl, 2003) que mostra que tecnologias da *Web* semântica como RDF ou RDFS proporcionam possibilidades interessantes, já que não há

[17] A utilização de servidores de UM é referenciada na seção 2.3.1.

restrições ao uso de diferentes esquemas em um único arquivo ou modelo RDF. Assim, pode-se aproveitar o atributo `performance_coding` do padrão PAPI ao mesmo tempo em que se usa o atributo `language_preference` do IMS para a representação das informações dependentes do domínio. A sintaxe é a seguinte:

```
<rdf:Description rdf:ID="record_1">
...
    <papi:performance_coding
          xmlns:papi="http://learninglab.de/papi#">
          number
    </papi:performance_coding>
...
</rdf:description>
<rdf:Description rdf:ID="record_2">
...
    <ims:language_preference
          xmlns:papi="http://learninglab.de/ims#">
          english
    </ims:language_preference>
...
</rdf:description>
```

Figura 24 – Exemplo de Integração entre Ontologias (Dolog&Nejdl, 2003, p.5)

O perfil CC/PP poderia ser utilizado para descrever as capacidades dos dispositivos.

Com a possibilidade de integração entre as ontologias, as Informações Independentes do Domínio podem ser obtidas de um UM centralizado e ontologias tipo PAPI e IMS LIP (seção 2.3) tanto podem ser utilizadas no Modelo de Sobreposição, no que diz respeito à experiência prévia do usuário (o que ele conhece), quanto podem ser usadas para representar as informações dependentes do domínio.

Como se pode observar pelo código acima, sob o ponto de vista de efetuar a adaptação não interessa de onde as informações vêm, pois elas serão usadas de maneira uniforme. As vantagens da estrutura proposta estão na conveniência da *"*modularização*"* e reuso.

A seguir, descreve-se, em linhas gerais, como a estrutura proposta pode ser utilizada em dois tipos de aplicação: Comércio Eletrônico e Ambiente Virtual de Aprendizagem:

4.1.4.1.1. Exemplo: Comércio Eletrônico

Informações demográficas e sobre a personalidade podem ser utilizadas como indicadores das preferências do usuário por produtos. Caso o acesso esteja sendo feito por banda larga e a partir de um *desktop*, o ambiente de uma loja pode

ser reproduzido e imagens do produto podem ser mostradas.

O Modelo de Sobreposição é capaz de informar sobre produtos comprados anteriormente e o grau de satisfação proporcionado pelos mesmos. Uma vez que este modelo pode ser compartilhado entre domínios, é possível ao sistema considerar compras feitas em outras lojas (observa-se que estão sendo ignoradas questões relativas à privacidade, armazenamento e outras pertinentes a um servidor de UM).

Uma livraria virtual pode utilizar informações sobre interesses literários e poder aquisitivo; já para uma loja de produtos esportivos os dados relevantes serão sobre o interesse por esportes e a capacidade de praticá-los.

Esses aspectos são observados do ponto de vista do comprador. Se o usuário fosse um fornecedor, outras informações seriam levadas em consideração. Ainda no caso de um comprador, o objetivo poderia ser o de efetivamente comprar ou fazer um levantamento de preços. A compra poderia também ser para uso próprio ou para presentear.

4.1.4.1.2. Exemplo: Ambiente Virtual de Aprendizagem

As informações estáticas e de contexto podem ser usadas para adaptar a apresentação, mas as preferências do aluno por uma ou outra matéria, ainda que sejam identificadas, não necessariamente são fatores relevantes para a indicação das mesmas, uma vez que existe um objetivo de aprendizagem a ser alcançado e pré-requisitos entre os conceitos.

O Modelo de Sobreposição representa, em geral, o que o usuário já sabe sobre os conceitos do domínio. Certificados já obtidos e estilos de aprendizagem são informações típicas da área educacional.

A seleção dos dados que precisam ser recuperados do UM pode ser influenciada por aspectos, tais como: o papel do usuário (professor ou aluno, por exemplo), quais objetivos ele precisa alcançar e que tarefas ele terá que desempenhar.

O domínio da educação e de aprendizado *on-line* é peculiar por dois motivos principais: (1) sempre existe um usuário e ele sempre é central à aplicação. Isto difere, por exemplo, de uma aplicação cultural que pode ser modelada considerando-se a utilização por um transeunte, ou de um *site* de comércio

eletrônico que pode ser utilizado para presentear terceiros. O aprendizado é contínuo e não se pode aprender por outros; (2) a modelagem do próprio domínio se constitui em uma ontologia relevante para a adaptação. Informações sobre artefatos de um museu ou sobre mercadorias de uma loja não são tão relacionadas ao usuário quanto um assunto sendo estudado no caso do domínio de aprendizado.

Essa diferenciação é bem definida por Chen & Riichiro (2004) que classificam ontologias como de tarefas (*task ontology*) e de domínio (*domain ontology*). Segundo eles, a primeira define um vocabulário para a modelagem de uma estrutura inerente à solução de problemas para todas as tarefas, de forma independente do domínio, enquanto que o vocabulário definido para a segunda especifica classes de objetos e relações que existem em certos domínios.

O sistema de suporte à aprendizagem multi-agente descrito em Chen & Riichiro (2004), define uma ontologia do modelo do aluno (*Learner model ontology*, Figura 25) como um tipo de ontologia de tarefa enquanto que a ontologia de domínio é o vocabulário do assunto sendo estudado (por exemplo: matemática, geometria, química). O trabalho argumenta que as informações sobre o usuário independentes do assunto podem ser reutilizadas por múltiplos sistemas, enquanto que as dependentes podem ser reusadas por sistemas do mesmo domínio.

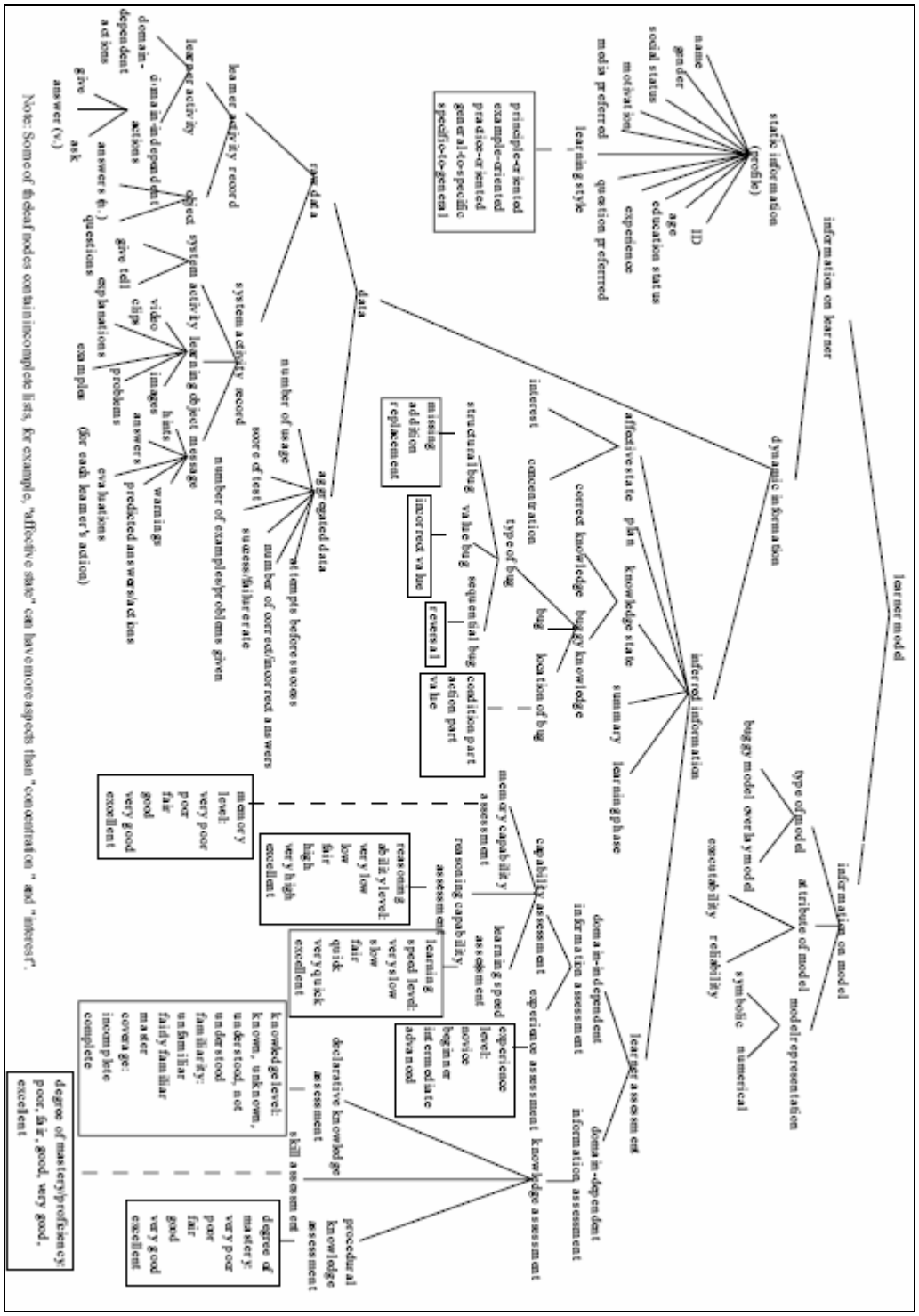


Figura 25 – Ontologia do Modelo do Aluno (Chen & Riichiro, 2004, p.7)

A proposta aqui apresentada, por sua vez, considera que o Modelo de Usuário deve ser geral para todos os domínios e, portanto, deve ser uma ontologia de tarefas capaz de se integrar à ontologia do domínio da aplicação.

O mecanismo de criação de visões de contexto proposto está de acordo com o exemplificado em Chen & Riichiro (2004), onde os modelos de aluno são montados a partir da ontologia proposta.

4.1.5.
Modelo de Adaptação

O Modelo de Adaptação do ASHDM usa as informações dos modelos semânticos descritos. Assim, é oportuno recapitular resumidamente cada um deles:

- O Modelo Conceitual ou Modelo do Domínio pode ser qualquer modelo baseado em uma linguagem da *Web* Semântica, que contenha uma descrição conceitual do conteúdo de uma aplicação. Considera-se que ele vai ser constituído por triplas RDF. O Modelo do Domínio pode incluir uma representação do usuário (UR), utilizada em conjunto com o UM (ver abaixo) para a construção de uma visão do contexto de adaptação;
- O Modelo de Navegação propicia a customização – através da criação de visões navegacionais – e a adaptação dinâmica de conteúdo e de navegação. Seus principais elementos são:
    - As instâncias das classes navegacionais, obtidas pelo mapeamento de classes conceituais base;
    - Os atributos das classes navegacionais, que podem ser originados da classe conceitual base ou de uma classe conceitual associada a ela;
    - Elos originados pelo mapeamento de propriedades conceituais; pela declaração de âncoras de estruturas de acesso ou classes navegacionais e pela definição do padrão de navegação;
    - Contextos navegacionais que apresentam um conjunto de nós com alguma característica em comum e um padrão de navegação, que pode apresentar alguma ordem;
    - Estruturas de acesso (índices), ou seja, um conjunto de âncoras que apontam para outros nós ou para outras estruturas dentro do mesmo contexto e que podem ser ordenadas.
- O Modelo do Contexto de Adaptação contém os elementos aos quais a adaptação é sensível, incluindo o UM, esquematizado da seguinte forma:
    - Informações Independentes do Domínio: vão ser obtidas de algum servidor de UM;
    - Modelo de Sobreposição: originado de alguma ontologia de usuário específica do domínio ou definido para a aplicação em questão;

- Informações Dependentes do Domínio: idem acima;

➢ O Modelo de Interface representa as interações entre usuário e sistema. Essencialmente, tal interação acontece através de eventos que podem ser "ativadores" (*Button* ou *Link*), "exibidores" (*Image* ou *Label*) ou "capturadores" (*widgets* do tipo *CheckBox, ComboBox, RadioButton* ou elemento composto por *Link*). A adaptação da apresentação ocorre, basicamente, no mapeamento entre a ontologia de *widgets* abstratos e a de *widgets* concretos e na definição do *layout* da interface.

O Modelo de Adaptação utiliza a estrutura da linguagem de regras PRML (*Personalization Rules Modeling Language*) proposta por Garrigós (2005). Segundo o referido trabalho, a separação entre o gerenciamento de regras e o desenvolvimento de aplicações proporciona maior flexibilidade e facilita a reutilização das regras. A PRML é uma linguagem de regras de alto nível que pode ser utilizada pelo projetista para especificar a adaptação a ser realizada em tempo de execução. A estrutura básica das regras definidas por essa linguagem é[18]:

```
When event do
        <body>
endWhen
```

onde o corpo da regra (<*body*>) pode representar simplesmente uma ação:

```
When  event  do
        action
endWhen
```

ou uma ação associada a uma condição:

```
When event do
        If condition then action endif
endWhen
```

No ASHDM, um evento (*event*) é uma interação do usuário com a aplicação, representada no Modelo de Interface. Uma ação (*action*) tanto pode adaptar o conteúdo, a navegação ou a apresentação quanto pode atualizar o Modelo do Contexto de Adaptação, incluindo o UM. Por fim, a condição se refere aos aspectos em função dos quais a aplicação é adaptada.

[18] A estrutura completa das regras permite associar algumas características, tais como prioridade e data de expiração. Além disso, pode-se definir uma periodicidade para o corpo da regra, ou seja, o intervalo de tempo em que a condição precisa ser checada.

4.1.5.1. Modelo de Execução

O processo de adaptação dinâmica é disparado a partir de eventos na interface. O Modelo do Contexto de Adaptação e as informações sobre a utilização do sistema (histórico de navegação e de interação, utilização da funcionalidade da aplicação)[19] são consultados para a escolha do conjunto de regras a ser aplicado. A execução das regras pode se dar, basicamente, de duas maneiras: aplicando as ações a todos os modelos pertinentes, interpretando o evento da interface nos modelos atualizados e recomeçando o ciclo ou de maneira encadeada, isto é, atualizações em um modelo ativam adaptações em outros, sendo que a cada ativação do Modelo de Interface o processo de adaptação é disparado.

A Figura 26 mostra um possível fluxo do Modelo de Execução do ASHDM. Nota-se que este fluxo representa a máxima granularidade de uma adaptação encadeada, ou seja, o processo de adaptação é disparado a cada clique e a adaptação é feita passo a passo. A adaptação poderia ser feita a cada sessão, isto é, uma única vez após a identificação do usuário (customização), ou a intervalos / processos pré-determinados. Além disto, as regras de adaptação de conteúdo, navegação e apresentação poderiam ser parte de um único módulo e a página seria adaptada em um único passo. A discussão sobre a adequação de cada procedimento é objeto de trabalhos futuros. O Modelo de Execução pode ser melhor entendido com o exemplo a seguir.

[19] Observa-se que estas informações podem atualizar o ACM, mas podem, também, ser utilizadas como condições das regras.

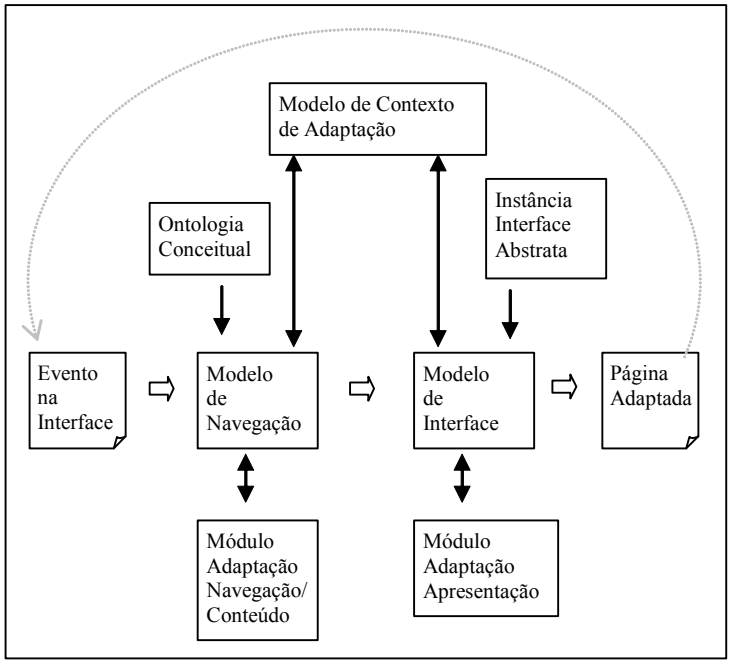


Figura 26 – Um possível fluxo do Modelo de Execução

4.1.5.1.1. Exemplo

O funcionamento de uma aplicação desenvolvida com o ASHDM é demonstrado a partir de um exemplo simples de adaptação em um *website* de um departamento acadêmico, cujo esquema conceitual é apresentado na Figura 27, extraída de Szundy (2004).

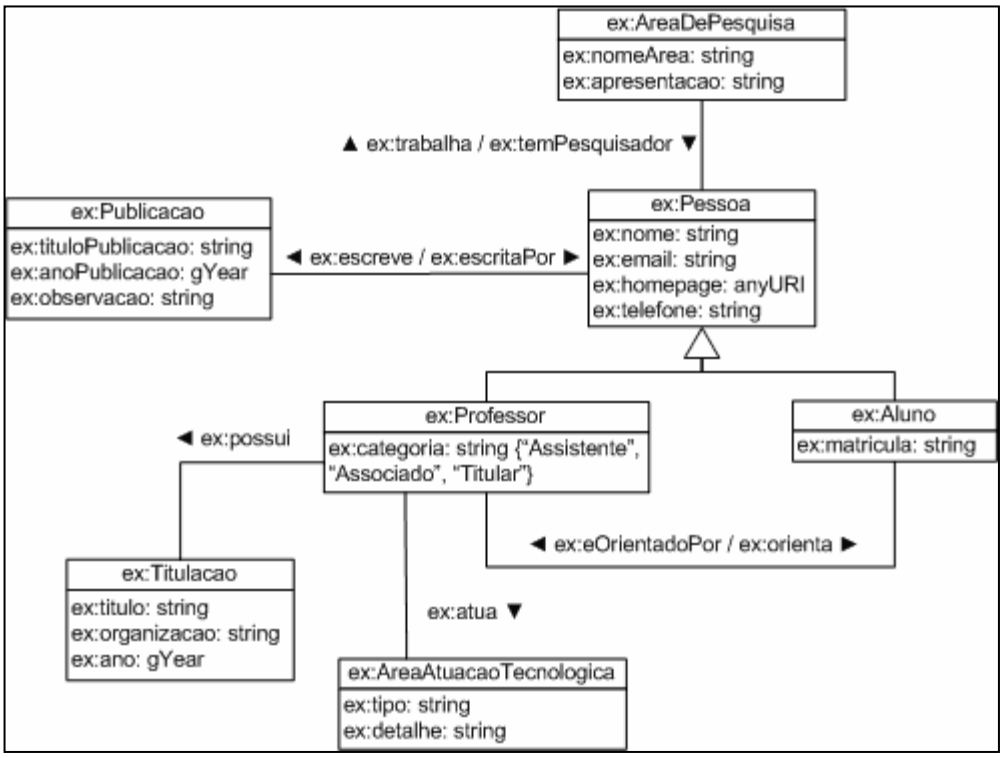


Figura 27 – Esquema conceitual para o *website* de um depto. acadêmico (Szundy, 2004, p.87)

Na situação em questão, os professores aparecem listados por ordem alfabética e a adaptação é disparada no momento em que o usuário clica no nome de um professor (evento na interface) (Figura 28).

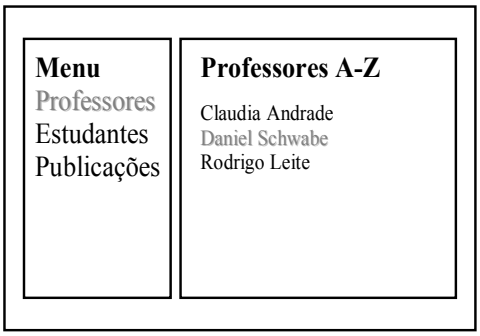


Figura 28 – Usuário escolhe professor

O sistema solicita, então, informações sobre o professor escolhido. Estas vão ser obtidas através de um mapeamento navegacional (adaptativo) sobre a ontologia conceitual, definindo quais são os dados e como eles vão ser apresentados.

A informação sobre o tipo de interesse do usuário pelo professor[20] é usada para determinar se o esquema navegacional utilizado será do tipo "agenda" – com informações sobre titulação, *e-mail*, telefone e *homepage* – ou do tipo "*portfolio*", contendo as publicações que podem ser contextualizadas por data ou por área, ainda de acordo com regras de adaptação. Também é definida a instância da interface abstrata a ser utilizada em cada caso. O módulo de interface é, então, acionado, juntamente com a máquina de adaptação.

No caso do esquema navegacional "agenda", nenhuma regra de adaptação é aplicável. A interface concreta gerada é mostrada na Figura 29.

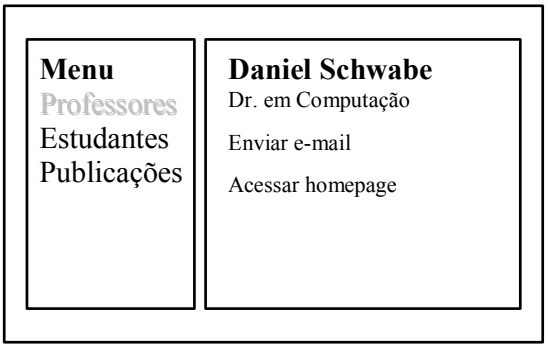


Figura 29 – Página gerada como agenda

[20] Observa-se que tal informação pode ser fornecida pelo usuário ou ser deduzida a partir da interação do usuário com o sistema. Os mecanismos para a obtenção de informações fogem ao escopo do trabalho.

No caso das publicações, a exibição de um resumo por página (Figura 30) ou dos títulos de várias publicações, com elos para os resumos (Figura 31), pode estar condicionada ao perfil do usuário.

Figura 30 – Página gerada como *portfolio*, com um resumo por página

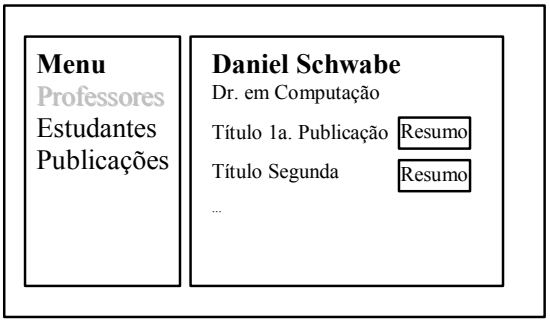


Figura 31 – Página gerada como *portfolio*, com vários títulos por página

Este exemplo é bastante simples, pois pretende apenas passar uma idéia concreta do processo de adaptação.

4.2. Implementação do ASHDM no AHA!

A finalidade desta seção é evidenciar que a proposta apresentada é adaptativa e geral, ou seja, pode ser usada para qualquer aplicação e domínio. Para tanto, demonstra-se o funcionamento no sistema proposto de um exemplo utilizado em um modelo correlato.

O AHA! já foi apresentado na seção 3.4. O exemplo é referente à Secretaria de Cultura (ProCultura) da Prefeitura Municipal de Nova Friburgo (PMNF). O objetivo aqui é exemplificar o processo de adaptação e, posteriormente, mostrar como o ASHDM poderia ser implementado no AHA!.

Toda aplicação desenvolvida com o AHA! é acessada por um formulário de entrada (*login form*, Figura 32) que identifica cada usuário individualmente. Caso seja o primeiro acesso ao servidor do AHA!, é necessário que o usuário se registre através de um formulário de registro, que pode ser usado para inicializar as

preferências do usuário (como opção por imagem em vez de texto; ou por áudio ou vídeo, além de texto). Um servidor AHA! pode conter várias aplicações, possibilitando o reuso da identidade do usuário entre elas.

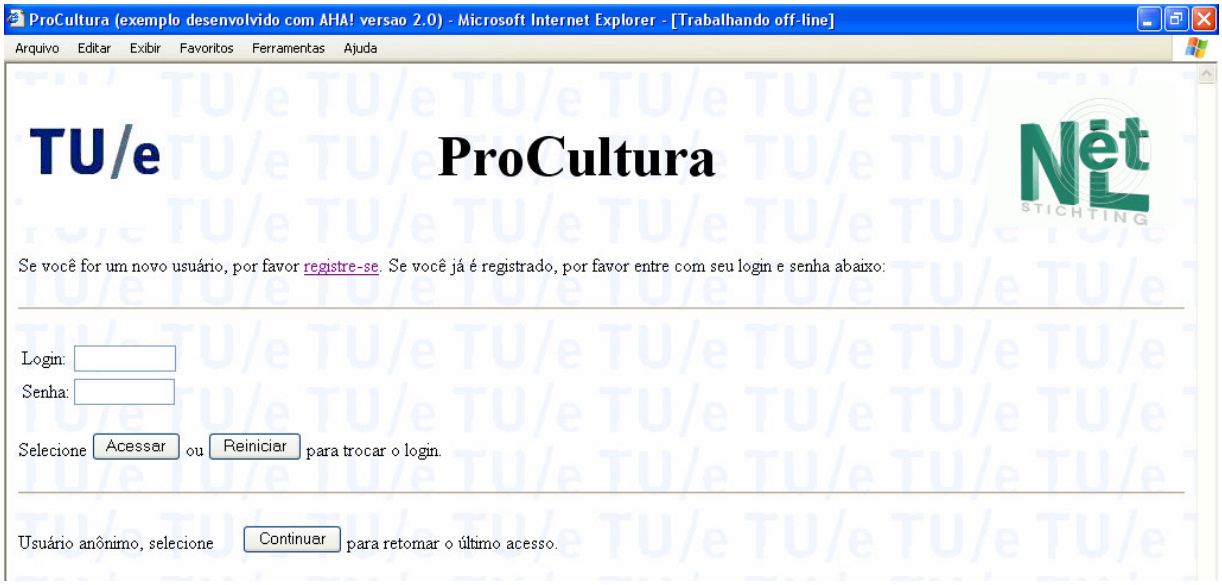


Figura 32 – Formulário de Entrada

Essencialmente, pode-se dizer que usuários requisitam páginas clicando em elos em um navegador e o AHA! exibe as páginas que correspondem a estes elos. Para gerar as páginas, o AHA! utiliza modelos do Domínio, do Usuário e de Adaptação.

O Modelo do Domínio contém a descrição conceitual do conteúdo da aplicação e consiste de conceitos e relações entre eles. O Modelo do Usuário consiste de conceitos com atributos e valores de atributos. Ele contém um Modelo de Sobreposição, uma vez que para cada conceito no DM tem um no UM. O Modelo de Adaptação consiste de regras que definem como as ações do usuário são traduzidas nas atualizações do UM e na geração de uma apresentação adaptada da página solicitada.

Pode-se considerar que o mecanismo de adaptação é disparado quando o usuário clica em um elo de uma página servida pelo AHA!. São executadas as regras associadas ao atributo “acesso” do conceito requisitado (ou do conceito correspondente à página requisitada). Este atributo é definido pelo sistema e é usado especificamente para começar a execução da regra. Cada regra pode atualizar o valor de atributo(s) de conceito(s) disparando as regras associadas a estes atributos. Basicamente, os valores dos atributos podem ser utilizados para determinar destinos de elos, inclusões condicionais de fragmentos e anotações dos

elos.

No exemplo, após o *login*, o usuário é direcionado para uma tela de introdução da aplicação (Figura 33), caso seja o primeiro acesso (isto é, o valor do atributo `visitado` relativo ao conceito `introdução` é `falso`), ou para uma tela de navegação por menu (Figura 34), caso a página de introdução já tenha sido visitada. Ou seja, existe uma adaptação do destino do elo.

Figura 33 – Tela de introdução

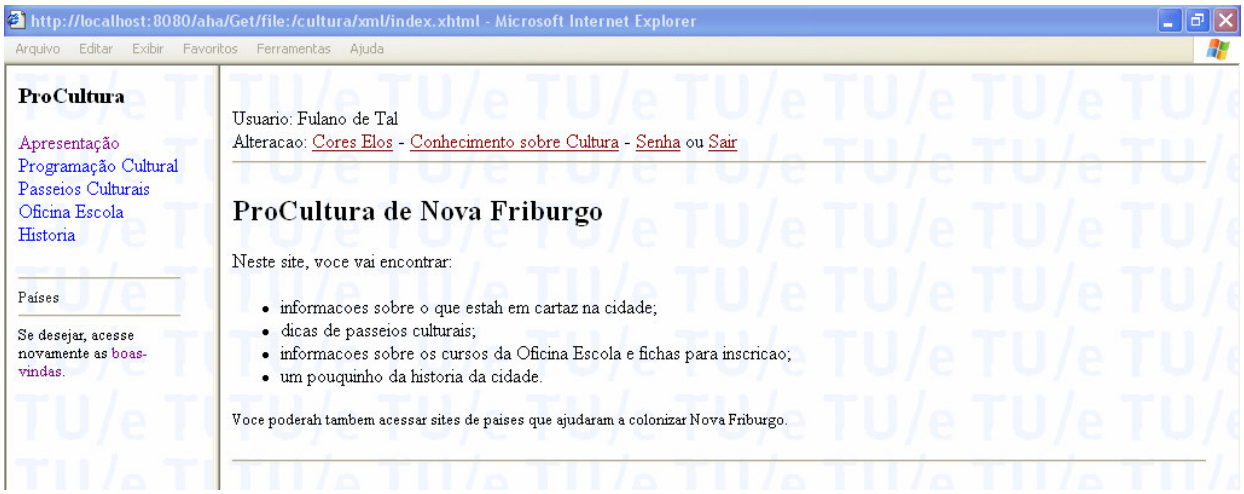


Figura 34 – Tela de navegação por menu

A inclusão condicional de fragmentos é exemplificada na Figura 35: o elo para um módulo de teste só é inserido caso o valor do atributo `conhecimento` do conceito `cultura` seja igual a 100. A implementação é feita por código xhtml como o exemplificado abaixo:

```
<if expr="cultura.conhecimento==100">
<block>
Agora que você já navegou pelo site,
<a href="avaliacao.xml" class="conditional">
teste seus conhecimentos</a>
</block>
</if>
```

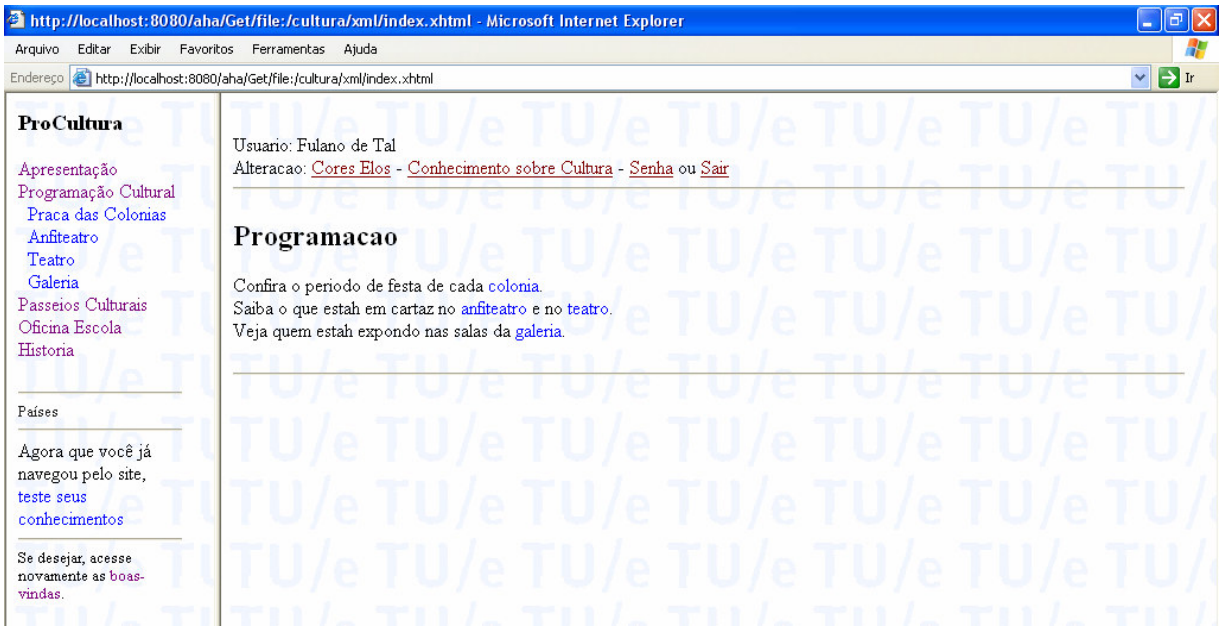


Figura 35 – Exemplo de inclusão condicional de fragmento

A anotação condicional dos elos (âncoras) é feita utilizando-se o seguinte esquema de cores: elos que apontam para páginas recomendadas aparecem em azul (elos "bons"), se elas ainda não foram visitadas e em roxo (elos "neutros"), caso contrário. Elos para páginas não recomendadas (ou seja, que ainda não devem ser visitadas) aparecem da cor do texto (preto: elos "ruins"). No exemplo, `Paises` aparece como um elo oculto por considerar-se que a sugestão para seguir tal elo só deverá ser feita caso o usuário tenha demonstrado interesse pelo conceito `colonia`.

No ASHDM, o Modelo do Domínio deve representar os conceitos e suas relações por triplas (`conceito_1, propriedade, conceito_2`), como as exemplificadas abaixo:

```
(colonia, prerequisito, paises).
(anfiteatro, interesse, teatro).
(teatro, interesse, anfiteatro).
```

O Modelo de Sobreposição precisa ter, para cada `conceito_1` do DM, uma tripla (`conceito_1, atributo, valor`) no UM. Os valores iniciais podem ser iguais a zero ou a algum valor pré-determinado, como no exemplo abaixo:

```
(apresentacao, visitado, falso).
(colonia, conhecimento, 0).
(colonia, interesse, 35).
(paises, interesse, 0).
```

Além do Modelo de Sobreposição, a visão do modelo de contexto adaptado deve incluir as preferências do usuário por texto ou imagem; por recursos adicionais como áudio e vídeo e por um esquema de cores para os elos. Informações sobre largura de banda e dispositivo, embora não explicitadas no AHA!, podem ser úteis.

Os dados obtidos através do mapeamento navegacional sobre a ontologia conceitual do domínio em questão, junto com os elementos definidos na interface abstrata, são utilizados para montar a interface concreta da página inicial da aplicação. A partir daí, neste exemplo, precisam ser monitorados apenas os eventos de interface do tipo *Link*.

Quando o usuário clica em um elo, o Modelo Navegacional é consultado para determinar quais nós e estruturas navegacionais serão apresentados. Neste momento, a máquina de adaptação é disparada para adequar esses nós e estruturas ao contexto de adaptação.

Para a adaptação do destino dos elos exemplificada, a regra poderia ser:

```
Se introdução.visitado == falso
      Ambiente = tela_introducao
Senão
      Ambiente = tela_menu
```

Aqui, `tela_introducao` selecionará um conjunto de regras de mapeamento navegacional cujos dados gerados, ao serem passados para o Modelo de Interface junto com a instância da interface abstrata selecionada, gerarão a interface abstrata correspondente à Figura 33. Analogamente, em relação ao parâmetro `tela_menu` e à Figura 34.

Para a inclusão condicional de fragmentos demonstrada, poderia ter-se a regra:

```
Se cultura.conhecimento == 100
      Mapeia_elo (teste, parâmetros)
```

Os dados e a instância da interface abstrata obtidos pelo mapeamento adaptado são então passados para o Modelo de Interface. A máquina de adaptação é, novamente, disparada para adaptar não só o mapeamento dos *widgets* abstratos em concretos, como também a "renderização" para a interface concreta. Classes CSS definiriam as cores dos elos "bons", "ruins" e "neutros", de acordo com as preferências do usuário.

Observa-se que a estruturação hierárquica dos conceitos, os relacionamentos

pré-definidos entre eles, bem como as regras embutidas no AHA! são típicos do domínio de educação. Aplicações de outros domínios demandam um esforço muito grande por parte dos autores para serem implementadas.

Assim como no caso do AHA!, o ASHDM pode ser implementado em outros *frameworks* de aplicações adaptativas. Os ambientes estudados neste trabalho (capítulo 3) são mais detalhados na modelagem do aprendizado e na atualização do UM, mas não necessariamente no aspecto de adaptação propriamente dito. O ASHDM proporciona a meta-adaptação e é capaz de levar em consideração aspectos do contexto mais variados.

4.3. Meta-adaptação

Um tipo de meta-adaptação possível consiste na seleção do Modelo de Adaptação desejado – incluído aí o Modelo de Execução – de acordo com o papel do usuário, seus objetivos e tarefas. Assim, por exemplo, ao detectar determinado padrão de navegação do usuário (como repetidas consultas a uma mesma página sobre um conceito), o sistema pode adotar uma outra estratégia de adaptação. Ou se o usuário sempre ampliar determinada imagem, ela poderá já aparecer ampliada em visitas futuras.

A adaptação dos modelos é um outro tipo de meta-adaptação propiciado pelo modelo aqui proposto. Uma vez que as ontologias incluem tanto o vocabulário quanto as instâncias propriamente ditas, ambos podendo ser representados através de triplas RDF, a máquina de adaptação que, essencialmente, consiste em testar um conjunto de condições sobre um conjunto de triplas, pode ser a mesma utilizada para acrescentar ou excluir triplas de qualquer um destes modelos. Em particular, uma vez que o próprio modelo pode ser representado em RDF, as regras são capazes de facilmente alterá-lo, da mesma forma como são capazes de alterar os valores dos dados.

O objetivo aqui é mostrar que esse tipo de procedimento é possível de ser expressado. A melhor forma de utilização tanto de regras quanto de metaregras, do ponto de vista da sua aplicabilidade em diversos domínios, é objeto de estudos futuros.

4.4. Arquitetura de Implementação

A arquitetura HyperDE (Nunes, 2005), acrescida de mecanismos de adaptação, é utilizada para validar o modelo proposto; a Figura 36 mostra esta arquitetura de forma esquemática.

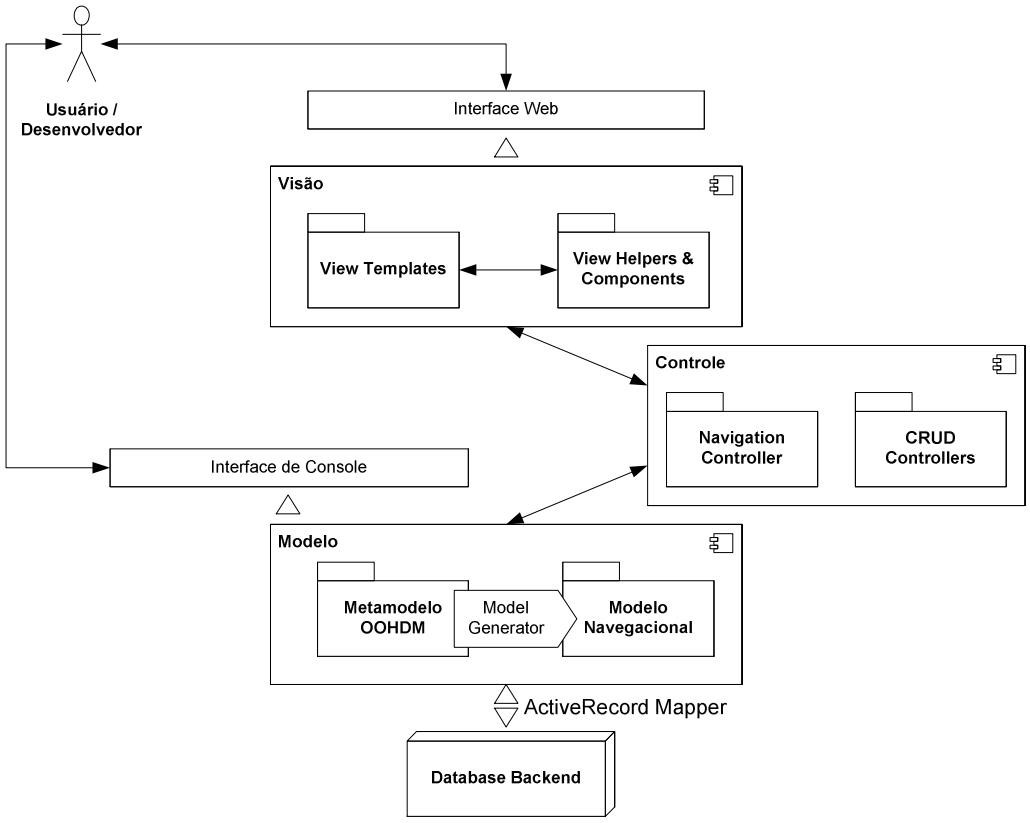


Figura 36 – Componentes da Arquitetura do HyperDE (Nunes, 2005, p.48)

Tal escolha é justificada por ser o HyperDE um *framework* e ambiente de desenvolvimento para aplicações hipermídia, dirigido por ontologias, que facilita a construção de aplicações modeladas segundo os métodos OOHDM ou SHDM. Sua camada de modelo navegacional disponibiliza as primitivas do metamodelo definidas para esses métodos: classes navegacionais, seus atributos e operações; elos; contextos navegacionais; estruturas de acesso (índices) e seus atributos; *landmarks*; nós e seus relacionamentos. A manipulação de tais primitivas é feita através de classes de objetos definidas primariamente pelo *framework*. A instanciação dessas classes permite, então, a construção do modelo navegacional da aplicação propriamente dita. As classes de objetos que representam o modelo navegacional da aplicação são construídas dinamicamente e podem ser manipuladas da mesma forma que as classes do metamodelo.

Com relação à interface, o HyperDE provê visões genéricas para os elementos de navegação disponíveis no *framework*, incluindo visões para suporte à navegação contextual e por índice. Entretanto, visões personalizadas (*templates*) podem ser definidas e associadas a contextos, classes navegacionais e índices, sobrepondo-se às visões genéricas. Em tempo de execução, a visão (*view*) apropriada é selecionada. A camada de interface do HyperDE disponibiliza uma biblioteca de funções auxiliares para a construção dessas visões. O código-fonte dos *templates*, por utilizar folhas de estilo para a formatação de seus elementos e as funções auxiliares para o desenho das primitivas do ambiente, aproxima-se bastante da definição abstrata da interface.

O ambiente HyperDE usa uma base semântica baseada em RDF e RDFS que armazena não só os dados da aplicação, como também os metadados que definem a ontologia do ambiente (metamodelo OOHDM/SHDM) e, ainda, os metadados que descrevem o modelo navegacional da aplicação desenvolvida. Esta característica favorece a meta-adaptação por tratar da mesma forma tanto a definição do vocabulário quanto as suas instâncias.

O objetivo desta seção é mostrar como o Modelo de Adaptação do ASHDM pode ser implementado no HyperDE estendido. Observa-se que a adaptação no mapeamento do Modelo Conceitual para o Navegacional e a adaptação da interface, como propostas pelo ASHDM, não foram implementadas nesta versão. Entretanto, a arquitetura possibilita essas adaptações a partir do momento em que os metamodelos forem atualizados no HyperDE. Cabe, ainda, observar que algumas adaptações obtidas por estes mecanismos também podem, do ponto de vista do usuário, ser alcançadas através de outros tipos de adaptação.

4.4.1. Modelo de Adaptação do HyperDE segundo o ASHDM

No HyperDE, a modelagem navegacional é implementada através da instanciação de metaclasses que representam as seguintes primitivas do modelo OOHDM / SHDM:

- Classes navegacionais, seus atributos e operações;
- Elos (relacionamentos entre as classes);
- Contextos navegacionais;
- Índices e seus atributos;

- *Landmarks*;
- Nós e seus relacionamentos.

Para que se possa discutir a adaptação, é necessário ressaltar alguns aspectos relativos à definição das instâncias para essas metaclasses. O HyperDE suporta diversos tipos de atributos, entre os quais o atributo computado, onde o valor do atributo é calculado em tempo de execução. Em relação ao contexto navegacional, no HyperDE ele é especificado basicamente por uma expressão de consulta que seleciona os nós que fazem parte do contexto sendo definido. Os índices podem ser derivados de contexto ou de consulta. No primeiro caso, a consulta que define quais os elementos do índice é dada pela consulta que define o contexto correspondente; no segundo caso, a expressão de consulta precisa ser especificada. A definição das instâncias dos nós e de seus relacionamentos é feita através da entrada de dados.

Na seção 4.1.2, foram identificados os seguintes pontos onde a adaptação pode ser implementada:

- Mapeamento de instâncias;
- Definição dos atributos da classe navegacional;
- Especificação dos valores desses atributos;
- Definição dos elos;
- Seleção dos nós que fazem parte de um contexto;
- Determinação do padrão de navegação interna;
- Definição das estruturas de acesso (índices).

Relacionando-se a instanciação proposta pelo HyperDE com os pontos de adaptação identificados acima, evidenciam-se alguns aspectos relativos à inserção de características adaptativas no HyperDE. Nota-se que o mapeamento de instâncias ainda não é suportado pelo HyperDE. Porém, o tipo de adaptação proporcionado por este mapeamento é mais relevante para a criação de uma visão navegacional, antes da utilização do sistema. A definição dos atributos, obtida pelo mapeamento de propriedades conceituais, também não é implementada pelo HyperDE. Contudo, o tipo de adaptação resultante da utilização de regras de mapeamento pode ser obtido, embora de forma menos consistente, por regras que atuam nos atributos definidos, por exemplo. Este seria o caso de se ter atributos definidos e não exibidos, já que a condição de não exibi-los poderia ser alterada

pelo usuário. Quanto à especificação dos valores de atributos, uma forma de se inserir a adaptação é através de atributos do tipo computado, pois tais valores, calculados em tempo de execução, podem, a princípio, ter condições incluídas no código definido para o cálculo. Entretanto, esse mecanismo só funciona para atributos simples. Já para a seleção de nós e definição de índices derivados de consulta, as expressões de consulta precisam ser modificadas para incluírem condições. O padrão de navegação é, por definição, seqüencial. As visões genéricas precisam ser modificadas para a seleção condicional do tipo de navegação intracontextual desejado.

Todavia, outros fatores precisam ser considerados. Não só a seleção dos nós, como a própria escolha do contexto deveria ser condicional. A definição de elos precisa ser adaptativa também. Na verdade, a inclusão efetiva de adaptação no ambiente HyperDE envolve modificações em sua própria modelagem. A Figura 37 mostra como fica o modelo de adaptação do HyperDE. Nota-se que este modelo não reflete todas as possibilidades de adaptação discutidas no modelo teórico do ASHDM. O objetivo, aqui, é fazer apenas uma prova de conceito, mostrando que é possível estender o metamodelo do HyperDE de uma forma consistente para utilizá-lo como arquitetura de implementação.

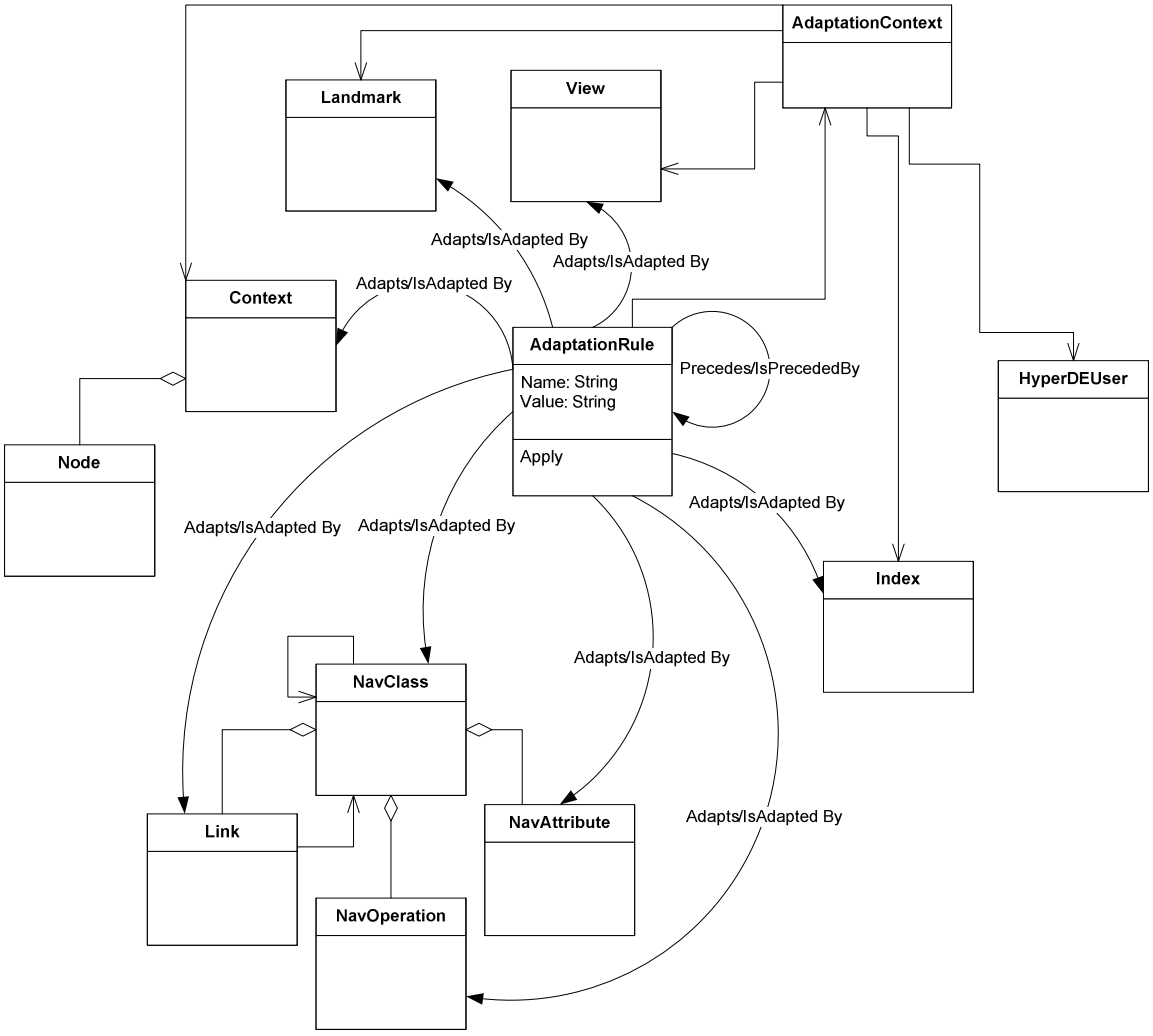


Figura 37 – Metamodelo estendido do HyperDE, incluindo o Modelo de Adaptação segundo o ASHDM

A metaclasse `AdaptationContext` implementa o Modelo do Contexto de Adaptação. `HyperDEUser` é um de seus componentes. Caso a aplicação já possua uma representação do usuário, basta declarar a classe correspondente como subclasse de `HyperDEUser`. Assim, pode-se informar que "aluno" ou "cliente" é o usuário da aplicação em questão. Um contexto (`Context`) é formado por um conjunto de nós (`Node`), ou seja, existe uma relação de agregação entre essas duas metaclasses. Uma classe navegacional (`NavClass`) é composta por atributos (`NavAttribute`), elos (`Link`) entre classes e operações (`NavOperation`). Uma `NavClass` pode ser subclasse de outra.

`AdaptationContext` fornece os dados solicitados pelas regras de adaptação para a geração do modelo navegacional adaptado da aplicação, a partir da instanciação das metaclasses `Node`, `NodeAttribute`, `NavOperation`, `Link`,

`Landmark`, `Index` e `Context`. Observa-se que, como a adaptação ocorre em tempo de execução, `NavClass` e `NavAttribute` não são adaptadas diretamente. A adaptação ocorre na instanciação de suas subclasses, respectivamente, `Node` e `NodeAttribute`. Já a adaptação da `View` pode ocorrer na seleção da mesma ou em sua edição. Pode existir uma relação de precedência entre as regras.

4.4.2. Modelo de Execução

O Modelo de Adaptação do ASHDM (seção 4.1.5) propõe regras que, de uma maneira geral, são do tipo:

```
When event do
      If condition then action endif
endWhen
```

No HyperDe, os eventos (*event*) produzidos pela interação do usuário com a interface da aplicação são capturados pela camada de controle. Esta é responsável por disparar os eventos na camada de modelo, construída sobre as primitivas definidas nos métodos OOHDM e SHDM, com os parâmetros necessários para a instanciação dos modelos, ou seja, para a geração do modelo navegacional da aplicação. A camada de controle é responsável, então, pela ativação da visão que resulta na construção da visualização da aplicação. Conseqüentemente, dependendo do tipo de adaptação, as regras podem estar associadas à camada de controle, à de modelo ou à de visão. Esta associação é feita pela inclusão de ganchos (*hooks*) para adaptação, possibilitada porque o ambiente HyperDE funciona com o conceito de "*closure*", ou seja, permite a passagem de código de programação como parâmetro de uma função.

As condições (*condition*), em geral, testam valores de atributos, relativos aos aspectos em função dos quais a adaptação é executada, e as ações consistem em alterar o conteúdo, a navegação ou a apresentação podendo, ainda, atualizar o modelo de dados, efetuando a adaptação desejada.

Assim, quando o usuário dispara um evento (clicando em uma âncora) é acionada, primeiramente, a camada de controle. Em geral, esse evento representa a seleção de uma âncora apontando para um índice ou para um elemento em um contexto. Se o índice ou contexto selecionados possuírem regras de adaptação associadas, estas serão executadas. A camada de modelo é chamada, então, com os respectivos parâmetros: que metaclasse será instanciada e com que valores.

Caso existam regras associadas à metaclasse, elas serão, por sua vez, executadas, condicionando a instanciação. Posteriormente, a camada de controle seleciona a *view* e apresenta o resultado ao usuário. Não só esta seleção pode ser condicional, como também o código CSS associado à *view* pode ser alterado segundo regras associadas, resultando na adaptação da interface.

4.4.3. Exemplo

Um exemplo simples de adaptação da especificação do valor de um atributo é apresentado com o objetivo de ilustrar a proposta de implementação. Considera-se que no *website* do departamento acadêmico utilizado na seção 4.1.5.1.1, a classe `Publicação` tem mais dois atributos: `resumo` e `abstract`. Associa-se uma regra à consulta de valor de atributos que define o valor de um atributo `resumoOuabstract`, como sendo o valor do `resumo` ou do `abstract`, de acordo com o idioma preferencial do usuário. A Figura 38 mostra a exibição do resumo para um usuário cujo idioma preferencial é o português, enquanto que a Figura 39 mostra o *abstract* quando o idioma preferencial é o inglês.

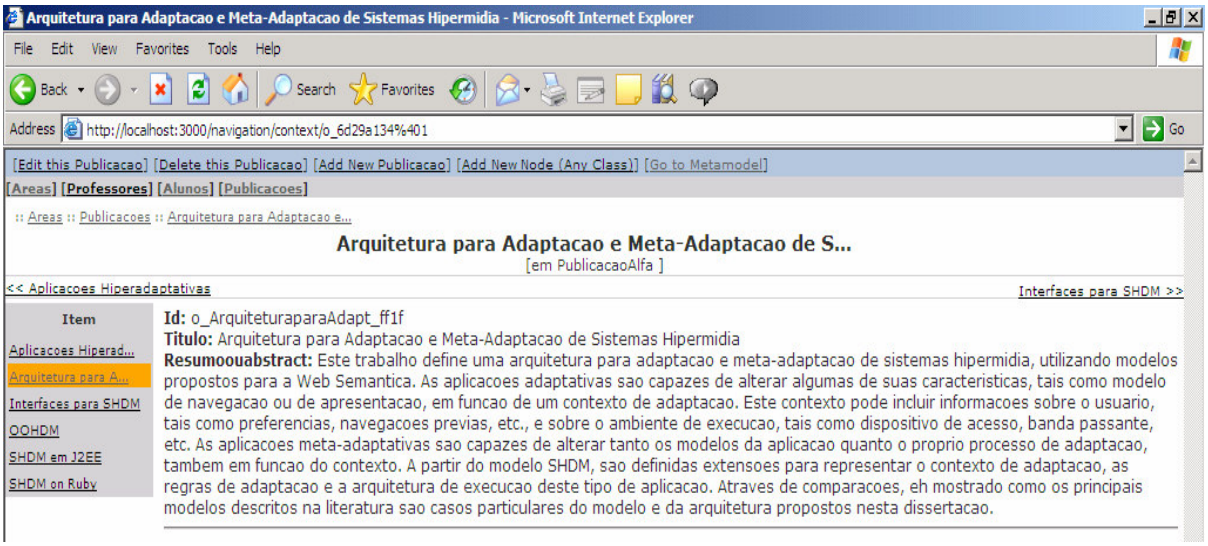


Figura 38 – Exibição de resumo para um usuário cujo idioma preferencial é o português

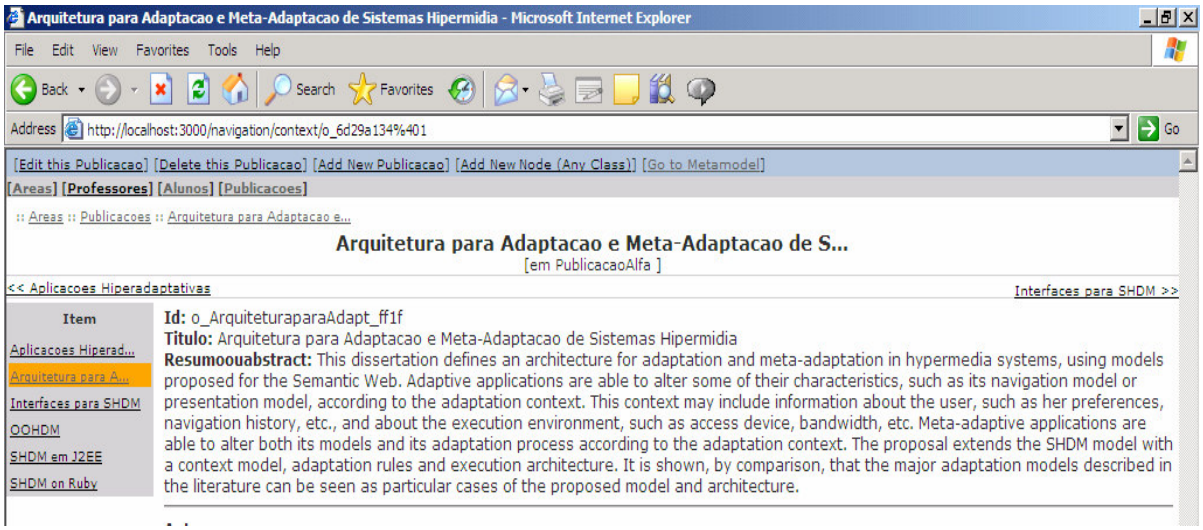


Figura 39 – Exibição de *abstract* para um usuário cujo idioma preferencial é o inglês

A regra de adaptação na DSL (*Domain Specific Language)* utilizada no A-HyperDE (*Adaptive-HyperDE*) tem a seguinte forma:

```
Classes:
 Publicacao:
  attrs:
   resumoOUabstract:
    rule1:
     hook: pre
     value: >
      if AdaptationContext.current.user.idiom=="english"
       attr_name = "abstract"
      else
       attr_name = "resumo"
      end
```

Esta regra é entendida assim: o rótulo `Classes:` indica que a regra se aplica à definição de uma Classe, que no caso é `Publicação`. Nesta classe, o que é adaptado é o atributo `resumoOuabstract` (rótulos `attrs:` e `resumoOuabstract:`). A regra propriamente dita segue-se ao rótulo `rule1:`. Toda regra pode ser executada antes ou depois de ser avaliada a definição do objeto sendo adaptado; isto é indicado pelo valor do rótulo `hook:` – o valor `pre` indica que a regra é executada antes e o rótulo `post` indica que a execução se dá após a avaliação da definição do item.

O código a ser executado pela regra segue-se ao rótulo `value:`. Tipicamente, este código é uma expressão condicional, que verifica alguma propriedade (ou propriedades) do contexto de adaptação. No exemplo, `AdaptationContext.current` retorna o contexto de adaptação (que na verdade é a única instância – "*singleton*" – nesta aplicação). O `AdaptationContext` pode ter os atributos "*view*", "*user*", "*index*", "*context*" e "*landmarks*", cujos valores são os respectivos elementos sendo adaptados. O usuário atual é armazenado no

atributo `user`. No exemplo, a condição verifica se o atributo `idiom` do usuário corrente na aplicação (i.e., o usuário "logado") tem valor `english`. Caso tenha, o atributo `resumoOuabstract` é substituído pelo atributo `abstract`; caso contrário, pelo atributo `resumo`.

A titulo de ilustração, a regra abaixo realiza a mesma adaptação, porém alterando o valor final do atributo, em vez de sua definição. Note que esta regra é do tipo `post`, significando que seu código é executado após a definição do atributo.

```
Classes:
 Publicacao:
  attrs:
   resumoOUabstract:
    rule1:
     hook: post
     value: >
      if AdaptationContext.current.user.idiom=="english"
       val = self.abstract
      else
       val = self.resumo
      end
```

Por ser executado após esta definição, o valor que foi obtido na avaliação normal do atributo é efetivamente ignorado e alterado conforme o texto da regra.

Outros dois exemplos de regras de adaptação são apresentados: o *landmark* `Alunos` leva ao contexto `AlunoPorArea` ou `AlunoPorProfessor`, dependendo do interesse do usuário e o *landmark* `Professores` mostra elos para `Alunos` ou para `Publicacoes`, ainda de acordo com o interesse:

```
Landmarks:
 Alunos:
  attrs:
   contextoSelecionado:
    rule1:
     hook: pre
     value: >
      if AdaptationContext.current.user.interest=="area"
       ctx_anchor = "AlunoPorArea"
      else
       ctx_anchor = "AlunoPorProfessor"
      end
```

```
Landmarks:
 Professores:
  attrs:
   elo:
    rule1:
     hook: pre
     value: >
      if AdaptationContext.current.user.interest=="alunos"
       landmark_link = "Alunos"
      else
       landmark_link = "Publicacoes"
      end
```