2
Descrevendo comportamento na *web*

2.1.
Modelos de *Design*

Métodos de *design* para aplicações hipermídia como o OOHDM [Schwabe e Rossi, 1998], UWA [UWA Consortium, 2002], UWE [Koch e Kraus, 2002], WebML [Ceri et al., 2000] e WSDM [Troyer e Leune, 1998], modelam e especificam uma aplicação *web* usando o conceito de "*separation of concerns*". Comumente possuem modelos para conteúdo, navegação e apresentação.

A estrutura navegacional é descrita por meio de primitivas como os nós, *links*, estruturas de acesso, entre outras. Aspectos das aplicações *web* modernas que estão além da base navegacional hipermídia são modeladas através de operações. As operações são úteis para descrever uma ação que é relacionada e acionada a partir de um nó único, como por exemplo, adicionar um produto numa lista de compras.

A seguir são apresentados alguns métodos e como estes modelam operações.

2.1.1.
OOHDM e SHDM

OOHDM (*Object Oriented Hypermedia Design Method*) é um método orientado a objetos para design de aplicações *web* [Rossi et al., 2003]. Este método compreende cinco atividades distintas, sendo estas: Levantamento de Requisitos, Modelagem Conceitual, Modelagem Navegacional, Projeto de Interface Abstrata e Implementação [Schwabe et al., 1999]. O processo se dá de forma iterativa e incremental seguindo o desenvolvimento de cada fase e produzindo novos modelos ou enriquecendo os já existentes.

O SHDM (*Semantic Hypermedia Design Method*) [Lima, 2003] é uma evolução do OOHDM, e manteve, portanto, os seus fundamentos, além de acrescentar no método os formalismos e primitivas introduzidos pela *Web* Semântica.

A etapa de Levantamento de Requisitos tem as seguintes subfases: identificação de atores e tarefas, especificação de cenários, especificação de casos de uso (*use cases*), validação dos casos de uso e especificação dos diagramas de interação com o usuário (UID). O objetivo principal dessa fase é identificar os requisitos da aplicação. É nesta etapa que as operações são identificadas e modeladas nos UIDs.

Na próxima fase, a de Modelagem Conceitual, é produzido o modelo conceitual da aplicação que é elaborado usando o diagrama de classes da UML. Neste diagrama são definidos todos os conceitos (tipos) e suas propriedades (atributos, operações e relacionamentos com outros conceitos). OOHDM/SHDM provê operações como métodos dos objetos de domínio da aplicação. A figura 1 ilustra o modelo conceitual para o domínio de loja virtual de CDs.

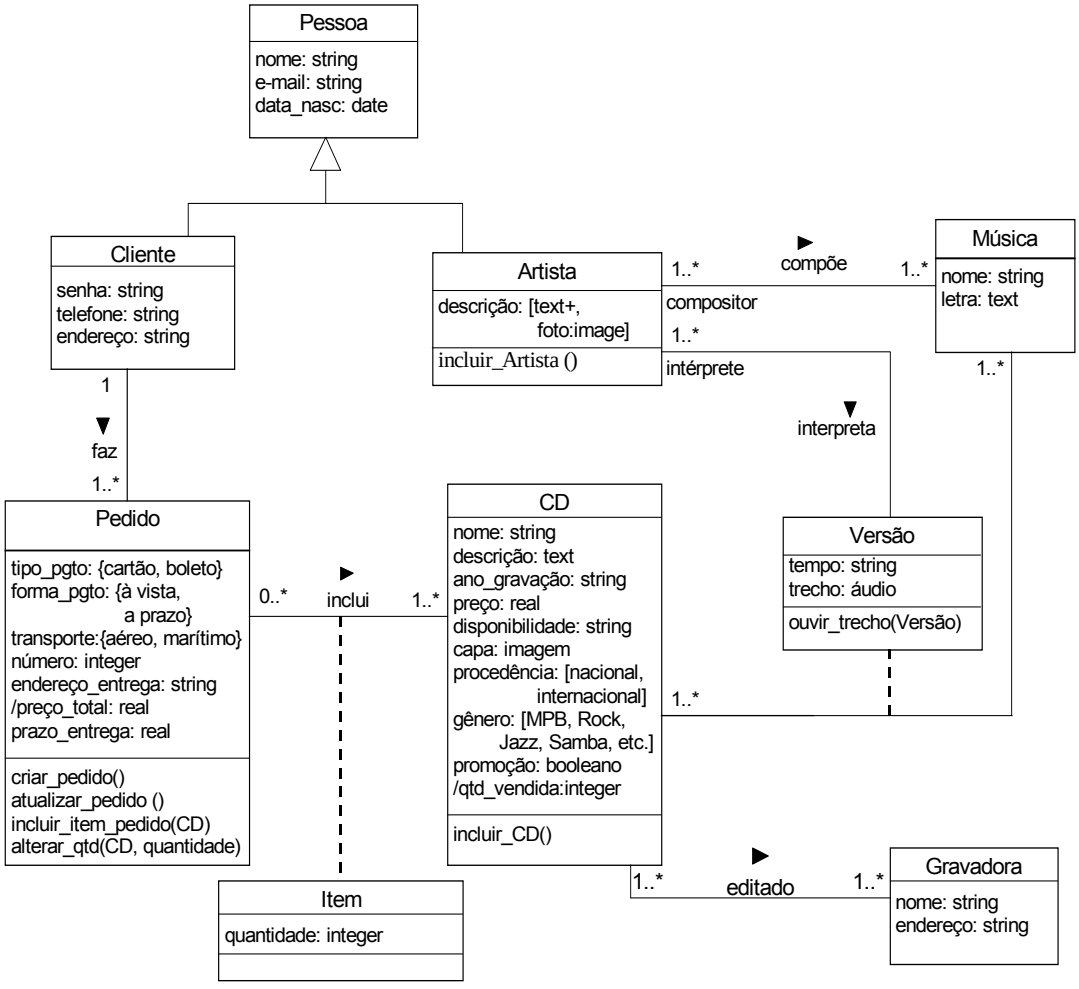

Figura 1 – Modelo Conceitual para o Domínio de Loja Virtual de CDs [Jacyntho, 2001]

O modelo navegacional é representado por um diagrama de classes que especifica os objetos que serão perceptíveis para o usuário durante a

navegação. Este modelo é composto de nós (ou classes navegacionais), elos (ou relacionamentos navegacionais) e estruturas de acesso (ou índices).

Uma classe navegacional é composta por atributos, operações e elos e suas instâncias são chamadas de nós. Os nós representam visões sobre as classes conceituais, definindo as propriedades que serão percebidas pelo usuário. Quanto aos elos, estes definem os *links* que permitem ao usuário navegar de uma instância de classe navegacional para outra. Pode-se dizer que um elo é uma visão sobre um ou mais relacionamentos conceituais (caminho), pois para estabelecer um elo entre duas classes navegacionais, tem que existir um caminho entre as correspondentes classes conceituais.

As operações definidas como métodos dos objetos de domínio da aplicação são chamadas a partir de métodos definidos nos nós de navegação. Elas são especificadas por um cabeçalho e um bloco de código que é executado quando ativado. As operações são úteis para descrever e modelar uma ação vinculada a um nó.

A figura 2 ilustra um modelo navegacional para a loja virtual de CDs.

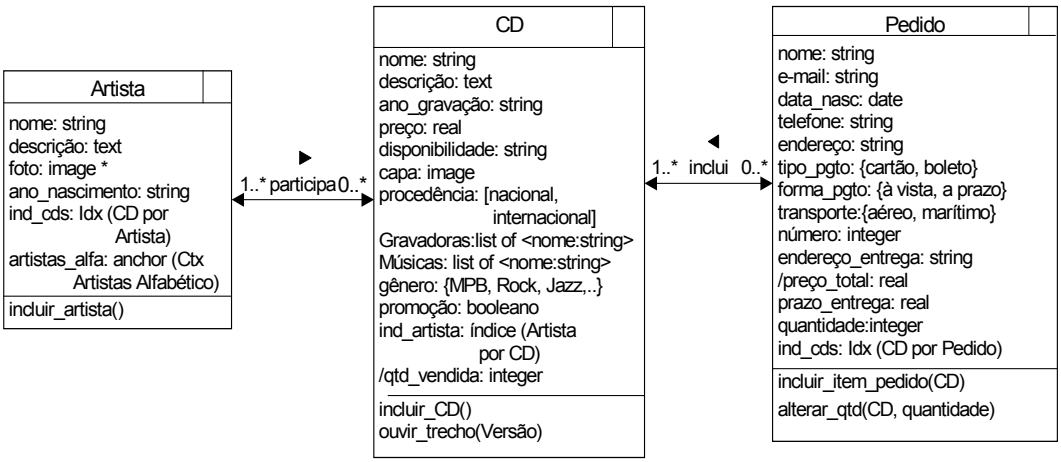

Figura 2 – Modelo Navegacional da Loja Virtual de CDs [Jacyntho, 2001]

Na fase de Projeto de Interface Abstrata é produzido o modelo abstrato de interface da aplicação, no qual serão especificados os objetos perceptíveis que estarão disponíveis para o usuário. Cada objeto da interface final do usuário é representado por um objeto abstrato (independente de tecnologia), que por sua vez, tem um mapeamento com um objeto navegacional.

Implementação é a última fase prevista pelos métodos OOHDM/SHDM. Nesta fase os modelos produzidos pelas etapas anteriores são mapeados e implementados no ambiente de execução escolhido, com o objetivo de gerar uma aplicação hipermídia.

2.1.2. UWA

O framework UWA (*Ubiquitous Web Applications*) [UWA Consortium, 2002] provê uma metodologia de design completa, um conjunto de metamodelos e ferramentas para o design de aplicações *web* que são multi-canal, multi-usuário, e geralmente, independente de contexto, ou seja, ubíquas [Distante e Tilley, 2005].

O ponto forte do UWA é a descrição de operações através da adoção da separação do *design* de "*concerns*" em quatro atividades principais [Distante, Tilley, Canfora e Huang, 2006]:

- Elicitação de Requisitos: definir o conjunto de *stakeholders* da aplicação e seus objetivos, e identificar os requisitos funcionais e não-funcionais;
- *Design* de Hipermídia e de Operações:
  - *Design* de Informação: identificar o conteúdo da aplicação e sua estrutura;
  - *Design* de Navegação: definir os nós de navegação que serão apresentados ao usuário e organizá-los em contextos (*Clusters* de Navegação), para especificar os possíveis caminhos de navegação;
  - *Design* de Operação: definir as operações que a aplicação provê para o usuário;
  - *Design* de Publicação: definir como a aplicação será organizada em páginas.
- *Design* de Transações: identificar as transações *Web* e modelar suas atividades;
- *Design* de Customização: definir como a aplicação será customizada dependendo do contexto de uso (dispositivo, usuário, localização, etc.).

O método UWA permite que operações (serviços) sejam invocadas para determinados propósitos enquanto ocorre a navegação em um site. As operações podem mudar o estado da parte hipermídia e do negócio da aplicação, além de afetar os sistemas subjacentes, controlar ou serem controladas por elementos externos, serem explicitamente iniciadas por usuários ou invocadas implicitamente em situações particulares. No UWA, as operações podem ser dos seguintes tipos:

- Operações simples: são atômicas e devem ser consideradas como um componente caixa-preta, respeitando o ponto de vista do usuário;
- Operações de múltiplos passos: preservam a essência de serem atômicas, porém não são mais caixas-pretas;
- Atividades: não são mais atômicas; podem ser vistas como transações de negócio e/ou *containers* para operações.

As operações atômicas são realizadas pelo usuário ou sistema em uma única interação, como por exemplo, inserir um item em um carrinho de compras ou mudar a ordem na lista de itens. Através das operações atômicas várias categorias de efeitos podem ser especificadas:

- Efeitos de informação: mudar a moeda de real para dólar;
- Efeitos de navegação (modificam a posição corrente do usuário): uma vez que o efeito de informação é obtido, o usuário é automaticamente levado a um nó diferente;
- Efeitos de apresentação (afetam a exibição): mudar a ordem visual dos itens de uma lista.

Já as atividades são um complexo fluxo de operações atômicas ou subatividades com a intenção de cumprir um determinado objetivo, tais como, comprar um produto ou selecionar artigos para uma conferência.

2.1.3. UWE

A metodologia UWE (*UML-based Web Engineering*) [Koch e Kraus, 2002] é baseada no enfoque orientado a objetos e interativo da *Unified Modeling Language* [UML, 2009] para especificação de aplicações *web*. O principal foco do UWE é a sistematização, personalização e geração semi-automática de aplicações *web*.

O processo de autoria do UWE é composto de quatro passos, sendo estes a Análise de Requisitos, o Design Conceitual, o Design Navegacional e o Design de Apresentação, que produzem os seguintes artefatos [Koch et al., 2001]: modelo de caso de uso, modelo conceitual, modelo do espaço navegacional e modelo da estrutura navegacional, e modelo de apresentação. Estes modelos são refinados em sucessivas iterações no processo de desenvolvimento do UWE.

O objetivo da Análise de Requisitos é definir os requisitos funcionais da aplicação *web* e modelar esses requisitos como casos de uso.

Na fase de Design Conceitual deve ser construído um modelo conceitual do domínio da aplicação a partir dos requisitos capturados nos casos de uso. Este modelo é representado por um modelo de classes da UML e possui como seus principais elementos: classe, associação e pacote. Se o modelo conceitual é composto de muitas classes, é recomendado que estas sejam agrupadas em pacotes UML. Uma classe é descrita através de seu nome, atributos, operações e variantes. As classes definidas no modelo conceitual são usadas durante a etapa de *Design* Navegacional para obter os nós da estrutura hipermídia. Uma representação gráfica de uma classe é ilustrada na figura 3.

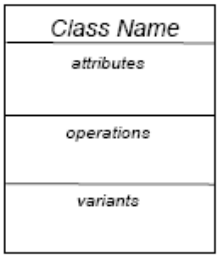

Figura 3 – Descrição de uma classe e seus componentes [Koch et al., 2001]

O modelo conceitual é uma visão estática de uma coleção de elementos do domínio. Os principais elementos utilizados neste modelo do UWE são *class* (classe) e *association* (associação). A figura 4 mostra um exemplo de modelo conceitual para uma biblioteca *online*.

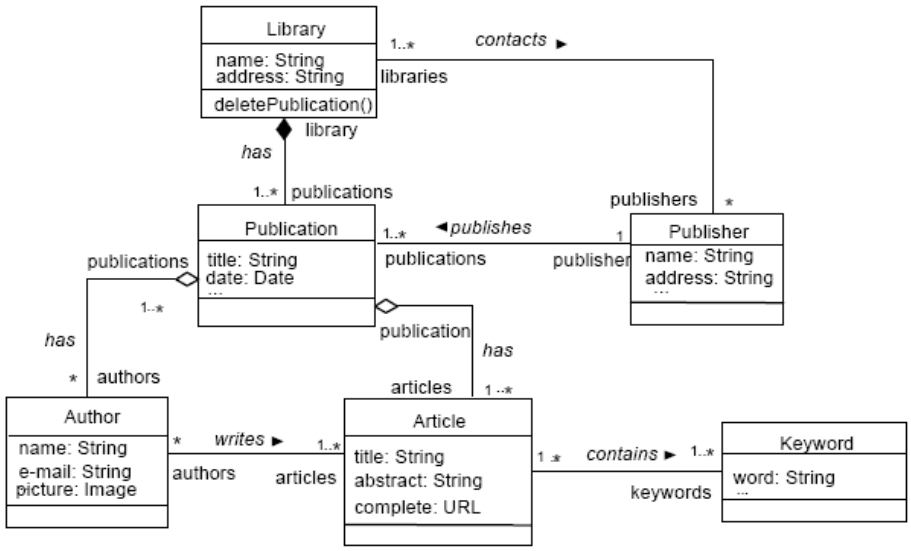

Figura 4 – Modelo Conceitual do UWE para Biblioteca *Online* [Koch e Kraus, 2002]

Baseado no modelo conceitual é construído um modelo navegacional que representa o espaço e a estrutura navegacional ao adicionar elementos de acesso que podem ser úteis para navegação. O método inclui uma série de elementos estereotipados da UML para *design* da navegação, como índices, visitas guiadas, *queries* e menus.

Classes navegacionais são usadas para construir o modelo do espaço navegacional e modelam as classes cujas instâncias são visitadas pelo usuário durante a navegação. A estas classes são dadas o mesmo nome correspondente às suas classes conceituais. Para sua representação é utilizado o estereótipo <<*navigation class*>>, como mostra a figura 5. Estas classes, assim como as do modelo conceitual, são compostas de atributos, operações e variantes.

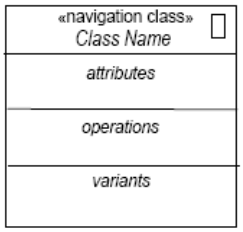

Figura 5 – Classe Navegacional [Koch et al., 2001]

A última fase, a de *Design* de Apresentação, visa o *design* de interface abstrata de usuários e o de interações do usuário com a aplicação *web*.

2.1.4. WebML

WebML (*Web Modeling Language*) [Ceri et al., 2000] é um modelo para *design* de aplicações suportado por uma ferramenta denominada WebRatio [Ceri et al., 2000]. Possui três modelos básicos: Modelo de Dados (correspondente ao modelo conceitual), Modelo de Hipertexto (análogo ao modelo navegacional) e Modelo de Apresentação.

O Modelo de Dados usa uma notação simplificada do modelo de Entidade-Relacionamento, no qual as entidades são representadas como retângulos e conectadas por relacionamentos binários, representados por linhas retas com o nome do relacionamento.

Os principais componentes do Modelo de Hipertexto do WebML são as *views*, áreas, páginas, unidades, operações, *links* e variáveis de sessão/aplicação. Uma *view* é um grafo de páginas, possivelmente agrupado em

áreas, permitindo aos usuários de um determinado grupo que realizem atividades específicas. As páginas contêm unidades conectadas por *links*, que representam partes atômicas de informação a serem publicadas. As unidades são elementos que representam a informação descrita no modelo de dados. É no modelo de hipertexto, ou modelo navegacional, que a navegação de *links* entre as páginas e entre as unidades é definida.

O método WebML permite que operações de atualização sejam especificadas para compor aplicações *web*. As operações básicas de atualização são: criação, modificação e exclusão de instâncias de uma entidade. É permitida também a criação ou exclusão de instâncias de um relacionamento. Algumas operações podem enviar e-mail ou invocar *web services*. Como operações não disponibilizam dados para o usuário, elas não são incluídas em páginas.

No Modelo de Apresentação são especificadas o posicionamento das unidades nas páginas e a aparência gráfica da aplicação.

2.1.5. WSDM

O WSDM (*Web Site Design Method*) foi criado em 1998 por Troyer e Leune [Troyer e Leune, 1998] e tinha como objetivo separar o *design* de "*concerns*" através de um enfoque sistemático e de multi-fases. É um método centrado no usuário e divido em quatro diferentes fases: Definição da Missão, Modelagem da Audiência, *Design* Conceitual (separado em Modelagem de Tarefas e Modelo Navegacional) e *Design* de Implementação [Plesser et al., 2005].

Na primeira fase do WSDM, a declaração da missão do *website* é formulada, tendo como principais objetivos a identificação do sujeito, do propósito e dos usuários alvos do *website*. Em seguida, na fase de Modelagem da Audiência, os usuários identificados são classificados em classes de audiência. Para cada classe de audiência devem-se definir, informalmente, suas características e requisitos (funcionais, não-funcionais, de informação e navegação). O resultado é uma hierarquia de classes de audiência.

A terceira fase é o *Design* Conceitual, na qual o conteúdo, a funcionalidade e a estrutura do *website* são especificados. O conteúdo e a funcionalidade são definidos durante a Modelagem de Tarefas e a estrutura navegacional é definida durante o Modelo Navegacional. O objetivo da Modelagem de Tarefas é analisar

as diferentes tarefas que as classes de audiência têm que realizar, e formalmente, descrever os requisitos necessários para estas tarefas. Para tal, são usados dois modelos: Modelo de Tarefa (*Task Model*) e Modelo de Objeto (*Object Chunk*).

Durante a Modelagem de Tarefas, é definida uma tarefa para cada requisito funcional. Em seguida, esta tarefa é descomposta em tarefas mais simples até chegar a tarefas elementares, formando uma hierarquia de tarefas. Nesta hierarquia, os relacionamentos temporais entre as tarefas são expressos por meio de operadores adequados. Para este propósito, a notação utilizada é *ConcurTaskTrees* (CTT) [Paterno et al., 1997], uma notação utilizada no campo de Interação Humano-Computador (IHC) para descrever atividades. Para facilitar o entendimento, foram criadas as seguintes categorias de tarefas [Troyer e Casteleyn, 2003]:

- Tarefas da aplicação (*Application tasks*): tarefas desempenhadas pelo sistema;
- Tarefas de interação (*Interaction tasks*): tarefas desempenhadas pelo usuário ao interagir com o sistema;
- Tarefas abstratas (*Abstraction tasks*): tarefas complexas que, portanto, são decompostas em tarefas mais simples.

A representação gráfica das categorias de tarefas é mostrada na figura 6.

Figura 6 – Categorias de Tarefas do WSDM [Troyer e Casteleyn, 2003]

A notação CTT utiliza uma série de operadores para expressar relacionamentos temporais entre as tarefas em uma hierarquia de tarefas. A lista completa de operadores pode ser encontrada em [Troyer e Casteleyn, 2003].

- Independente de ordem (T1 |=| T2): as tarefas podem ser executadas em qualquer ordem;
- Escolha (T1 [] T2): uma tarefa pode ser escolhida e apenas a tarefa escolhida pode ser executada;
- Desativação (T1 [> T2): a primeira tarefa é desativada assim que a segunda começa;
- Habilitação com troca de informação (T1 []>> T2): a segunda tarefa é habilitada quando a primeira termina e alguma informação é passada da primeira para a segunda tarefa;

- Suspensão-retomada (T1 |> T2): T1 pode ser suspensa para executar T2, e assim que T2 terminar, T1 é retomada no exato ponto onde parou;
- Iteração (T*): a tarefa pode ser executada várias vezes;
- Opcional ([T]): tarefa opcional;
- Recursão: ocorre quando a subárvore que modela a tarefa contém a própria tarefa;
- Transação (-> T <-): a tarefa e suas subtarefas têm que ser executadas como uma transação (atomicamente).

Para exemplificar como funciona a modelagem de tarefas, a figura 7 ilustra um modelo de tarefas para “Comprar Itens” (“*Buy Items*”) usando a notação CTT.

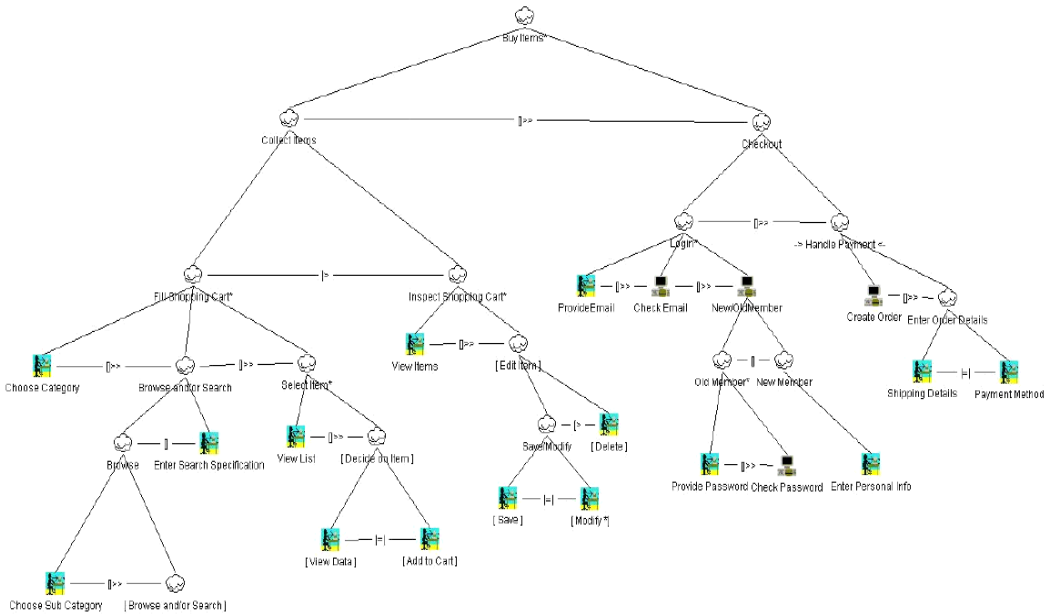

Figura 7 – Modelo de Tarefas para “Comprar Itens” [Troyer e Casteleyn, 2003]

Depois que um Modelo de Tarefas é criado para uma tarefa, deve-se modelar a informação através do Modelo de Objeto para cada tarefa elementar que existir.

A segunda etapa do *Design* Conceitual é o Modelo Navegacional, que tem como objetivo descrever como o usuário navega entre as tarefas modeladas anteriormente. Isto é feito definindo um componente para cada tarefa elementar de interação do Modelo de Tarefas, e criando elos (*process logic links*) entre estes componentes, de acordo com os relacionamentos temporais entre as tarefas. Para completar, cada componente é ligado ao Modelo de Objeto criado para a respectiva tarefa.

A última fase é a de *Design* de Implementação, na qual os modelos especificados na fase de *Design* Conceitual são enriquecidos com informações necessárias para a implementação da aplicação [Plessers et al., 2005].

2.1.6. *Design* de operações

Em [Jakob, Schwarz, Kaiser e Mitschang, 2006a] é apresentado um novo enfoque para o desenvolvimento de aplicações *web* que dependem de uma sofisticada lógica de aplicação, tais como, *sites* de *e-commerce* e sistemas de leilão *online*. Para estes autores, apenas os modelos de conteúdo, navegação e apresentação existentes nas principais metodologias não são suficientes para expressar a complexidade das operações que acessam ou modificam o conteúdo de uma aplicação *web*. Portanto, foi definido um modelo adicional, denominado Modelo de Operações, no qual a lógica de operações das aplicações *web* é especificada.

No Modelo de Operações são definidas as operações que provêm acesso de leitura e escrita aos objetos e relações do modelo de conteúdo. Essas operações determinam qual conteúdo pode ser acessado por qual componente da interface do usuário da aplicação *web*. A figura 8 mostra um exemplo de Modelo de Operação para a entidade "*Employee*".

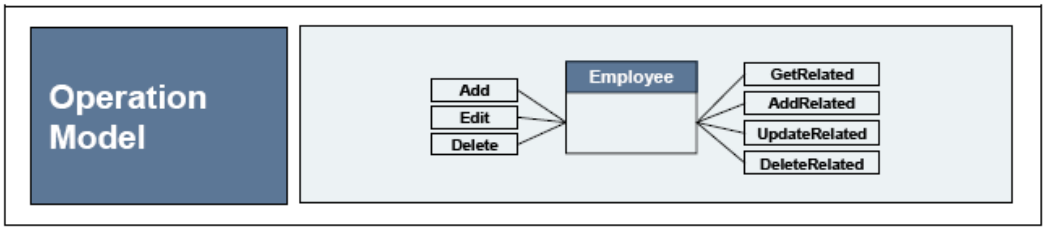

Figura 8 – Modelo de Operações [Jakob, Schwarz, Kaiser e Mitschang, 2006b]

No lado esquerdo da entidade, são listadas as operações básicas que provêm acesso aos seus dados. Estas operações só podem ocorrer uma vez e são auto-explicativas. Já no lado direito, são listadas os tipos de operações que provêm acesso aos relacionamentos entre a entidade "*Employee*" e outras entidades. Cada tipo de operação pode ocorrer inúmeras vezes e a entidade pode participar em qualquer relacionamento.

Este Modelo de Operações é interligado aos modelos de conteúdo e navegação/composição, servindo como mediador entre eles.

2.2. *Web services* semânticos

De acordo com o W3C (*World Wide Web Consortium*), tecnologias de *web services* são um conjunto de padrões que promovem interoperação entre diferentes aplicações de software, que podem estar em várias plataformas e ambientes. As operações são descritas através da WSDL *(Web Services Description Language*), que fornece meios para invocá-las junto com suas entradas e saídas, seus nomes e seus pontos de ligação.

A funcionalidade de um componente é modelada atribuindo à sua operação um identificador e aos seus dados um tipo definido por um esquema XML (*eXtensible Markup Language*). Para computadores, a utilização desta forma de descrição dos componentes de *web services* tem uma interpretação limitada. Sendo assim, é necessário que aspectos semânticos adicionais sejam especificados para que haja uma maior automação no uso de *web services* e mais facilidade na integração em processos *web*.

Algumas propostas surgiram com o intuito de implementar os *web services* semânticos. Dentre elas podemos destacar o WSMO (*Web Service Modeling Ontology*) e OWL-S, sendo estas recomendadas pela W3C. São também as mais maduras, por possuírem implementação real de qualidade.

A OWL-S, anteriormente DAML-S, é baseada na OWL (*Ontology Web Language*), cuja ontologia modela *web services* e provê um conjunto de construtores com o intuito de descrever as propriedades e capacidades dos mesmos, de forma não ambígua e interpretável automaticamente [OWL-S, 2008]. OWL-S é composta por três subontologias: perfil de serviço (*service profile*), que descreve o que o serviço faz; modelo de processo (*process model*), que especifica como o serviço é usado; e fundamentos (*grounding*), que mostra como interagir com o serviço.

Para um resultado satisfatório, as funcionalidades que o serviço provê e a especificação das condições que devem ser satisfeitas, precisam estar modeladas no perfil do serviço. Dois aspectos funcionais do serviço são representados: a transformação da informação (representada pelas entradas e saídas – *inputs* e *outputs*) e a mudança do estado produzida pela execução do serviço (representada pelas pré-condições e efeitos – *preconditions* e *effects*).

Não existe um esquema para descrever as instâncias de IOPR (*Input, Output, Precondition, Result*) na ontologia do perfil de serviço. Esse esquema é especificado na ontologia de processo. Espera-se que o IOPR publicado no perfil

seja um subconjunto daqueles publicados pelo processo. Sendo assim, a parte de descrição do processo cria todas as instâncias de IOPR que são apontadas pelas instâncias do perfil.

Os critérios de IOPR em OWL-S têm as seguintes propriedades da classe perfil de serviço na ontologia [OWL-S, 2008]:

- *hasParameter*: é o contra-domínio da classe *Parameter* da ontologia de processo, não sendo utilizada diretamente, pois serve apenas como uma superpropriedade das demais;
- *hasInput*: é o contra-domínio da classe de *Input* na forma como foi definida na ontologia de processo;
- *hasOutput*: é o contra-domínio da classe de *Output* na forma como foi definida na ontologia de processo;
- *hasPrecondition*: especifica uma das pré-condições do serviço e tem como contra-domínio a classe de *Precondition*, definida de acordo com o esquema da ontologia de processo;
- *hasResult*: especifica um dos resultados do serviço, de acordo com a definição da classe *Result* da ontologia de processo, e sobre quais condições as saídas são geradas.

Um serviço pode ser visto como um processo. Um processo não é um programa executável e sim uma especificação de como os clientes podem interagir com um serviço ou com um conjunto de serviços. Um processo atômico é a descrição de um serviço que espera uma mensagem e retorna uma mensagem como resposta.

Processos podem receber diferentes tamanhos de parâmetros:

- Qualquer número de entradas (*inputs*), inclusive nenhuma. As entradas representam a informação que deve ser especificada para o serviço, de forma a produzir um resultado;
- Qualquer número de saída (*outputs*), sendo a informação retornada ao solicitante pelo serviço;
- Qualquer número de pré-condições (*preconditions*). Todas as pré-condições devem ser mantidas em ordem para garantir a correta invocação do serviço;
- Qualquer número de efeitos (*effects*). No momento que um processo é executado, saídas e efeitos podem precisar que certas condições se mantenham verdadeiras. Quando se consegue

manter essas condições verdadeiras, e um conjunto de saídas e efeitos é produzido, temos um resultado.

Na especificação da figura 9 [OWL-S, 2008], podemos ver um trecho do exemplo da descrição de um processo de pagamento com cartão de crédito. A cobrança é autorizada se o limite do cartão não foi excedido. Se o limite foi ultrapassado, a única saída é uma notificação de falha, que não será apresentada no presente trabalho.

```
<process:AtomicProcess rdf:ID="Purchase">
   <process:hasInput>
      <process:Input rdf:ID="ObjectPurchased"/>
   </process:hasInput>
   <process:hasInput>
      <process:Input rdf:ID="PurchaseAmt"/>
   </process:hasInput>
   <process:hasInput>
      <process:Input rdf:ID="CreditCard"/>
   </process:hasInput>
   <process:hasOutput>
      <process:Output rdf:ID="ConfirmationNum"/>
   </process:hasOutput>
   <process:hasResult>
     <process:Result>
        <process:hasResultVar>
           <process:ResultVar rdf:ID="CreditLimH">
              <process:parameterType rdf:resource="&ecom;#Dollars"/>
           </process:ResultVar>
        </process:hasResultVar>
        <process:inCondition>
          <expr:KIF-Condition>
            <expr:expressionBody>
              (and (current-value (credit-limit ?CreditCard)
                                  ?CreditLimH)
                   (>= ?CreditLimH ?purchaseAmt))
            </expr:expressionBody>
          </expr:KIF-Condition>
        </process:inCondition>
        <process:withOutput>
           <process:OutputBinding>
              <process:toVar rdf:resource="#ConfirmationNum"/>
              <process:valueFunction rdf:parseType="Literal">
                 <cc:ConfirmationNum xsd:datatype="&xsd;#string"/>
              </process:valueFunction>
           </process:OutputBinding>
        </process:withOutput>
        <process:hasEffect>
          <expr:KIF-Condition>
            <expr:expressionBody>
              (and (confirmed (purchase ?purchaseAmt) ?ConfirmationNum)
                   (own ?objectPurchased)
                   (decrease (credit-limit ?CreditCard)
                             ?purchaseAmt))
            </expr:expressionBody>
          </expr:KIF-Condition>
        </process:hasEffect>
     </process:Result>
        …
   </process:hasResult>
</process:AtomicProcess>
```

Figura 9 – Processo de Pagamento com Cartão de Crédito [OWL-S, 2008]

O resultado da execução do processo será o débito no cartão de crédito e a redução do dinheiro na conta vinculada.

Outra proposta para descrição de *web services* semânticos é o WSMO. É uma ontologia para descrever semanticamente *web services*, definida pelo *Digital Enterprise Research Institute*. O WSMO é composto de quatro elementos principais [Domingue, Roman e Stollberg, 2005]: ontologias, que fornecem os conceitos e relacionamentos usados por outros elementos; objetivos, que definem os problemas a serem resolvidos; descrições de *web services*, para definir os vários aspectos de um *web service;* e mediadores, para resolver problemas de interoperabilidade.

O elemento de descrição de *web services* no WSMO especifica os aspectos comportamentais do *web service* através dos seguintes atributos: importação de ontologias, uso de mediadores, propriedades não funcionais, capacidades e interfaces.

A funcionalidade de um *web service* é definida pela sua capacidade, compreendendo os seguintes conceitos [WSMO, 2006]:

- Importação de ontologia: utilizado para importar ontologias caso não haja conflitos a serem resolvidos;
- Uso de mediadores: a importação de ontologias pode ser realizada através de mediadores que se encarregarão dos passos de alinhamento, fusão e transformação necessários;
- Propriedades não-funcionais: conjunto de propriedades pertencentes à capacidade, utilizando o padrão recomendado pelo WSMO para interoperabilidade;
- Variáveis compartilhadas: variáveis que são compartilhadas pelas pré-condições, pós-condições, suposições e efeitos;
- Pré-condição (*precondition*): especifica o espaço de informação que precisa ser válido antes da execução do serviço;
- Suposição (*assumption*): condição do mundo real que precisa ser válida antes da execução do serviço;
- Pós-condição (*poscondition*): descreve o espaço de informação do serviço após a sua execução;
- Efeito (*effect*): situação do mundo real após a execução do serviço.

A capacidade pode ser obtida através da descrição de como a interface obtém a funcionalidade do serviço, ou seja, como se comunicar com ele (coreografia) e como fazê-lo cooperar com outros serviços (orquestração).

2.3. Análise dos Modelos

Do ponto de vista de um projetista é importante que um modelo detenha determinadas características que atendam às necessidades especificadas pelo usuário. Um modelo de operações deve possuir as seguintes características:

- Descrever a assinatura da operação: o modelo deve permitir que um conjunto de informações que identificam a operação seja definido. Esse conjunto de informações é denominado assinatura da operação, sendo composto pelo nome, objetos que são aceitos como parâmetro de entrada e o tipo de valor retornado pela operação.
- Permitir o entendimento de qual é o objetivo da execução da operação sem analisar o código: através das pré e pós-condições. Estas diretivas estabelecem como será o contrato da operação, ao descrever suas expectativas e compromissos.
- Estar integrado com os outros modelos do método de desenvolvimento de aplicações *web*, como os modelos de conteúdo, navegação e apresentação.

Os métodos de *design* para aplicações hipermídia descritos no início do capítulo propõem diferentes mecanismos para lidar com os aspectos relacionados à especificação de operações. Porém, esses métodos falham ao não dar suporte à representação de todas as características que um modelo de operações deve possuir.

Os métodos OOHDM/SHDM e UWE modelam operações como métodos de classe, o que não é viável em um modelo de descrição de dados como RDF, que não possui o conceito de classe. UWA, WebML e WSDM possuem um metamodelo para descrever operações baseado em objetos, que não é compatível com o modelo de descrição de dados utilizados na *web* semântica, o RDF. O WSDM enfatiza a interação entre tarefas, em como estas estariam organizadas em um processo e não se preocupa em detalhar a especificação da operação que realizaria a tarefa. Ou seja, este método não possui modelo especializado para descrever operações. Nenhum dos métodos referenciados possui um vocabulário que contemple a descrição das diretivas de pré e pós-condições de uma operação.

O trabalho proposto em [Jakob, Schwarz, Kaiser e Mitschang, 2006a] visa à criação de um novo Modelo de Operações para enunciar os conceitos que não

são suportados pelos modelos comumente utilizados (conteúdo, navegação e apresentação). Este modelo falha em não prover uma estrutura genérica o suficiente que permita a criação de qualquer tipo de operação e por não fornecer meios de especificar pré e pós-condições de uma operação.

As propostas de descrição de *web services* semânticos, OWL-S e WSMO, propõem diretivas e metadados para a modelagem de comportamento em um ambiente semântico, porém sem integração com os outros enfoques da aplicação, tais como os modelos de conteúdo, navegação e apresentação.

A principal motivação para o desenvolvimento deste trabalho parte do fato de não haver modelos na literatura que especifiquem operações semanticamente, de forma integrada com os outros modelos dos métodos de aplicações hipermídia. Portanto, nosso objetivo é desenvolver um modelo que descreva comportamento para aplicações na *web* semântica. O método SHDM será estendido de forma a incluir este novo modelo. Este método foi escolhido pela sua maturidade e por possui um ambiente de implementação estável, o HyperDE.