UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

DEPARTAMENTO DE SISTEMA E COMPUTAÇÃO - DSC

CURSO DE MESTRADO EM INFORMÁTICA - COPIN

DISSERTAÇÃO DE MESTRADO

SEI-TUR: UM SISTEMA DE CRIAÇÃO DE ROTEIROS TURÍSTICOS

ANA CAROLINA DE JESUS MAIA

(MESTRANDA)

DR. ULRICH SCHIEL

(ORIENTADOR)

CAMPINA GRANDE
ABRIL – 2004

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

DEPARTAMENTO DE SISTEMA E COMPUTAÇÃO - DSC

CURSO DE MESTRADO EM INFORMÁTICA - COPIN

SEI-TUR: UM SISTEMA DE CRIAÇÃO DE ROTEIROS TURÍSTICOS

ANA CAROLINA DE JESUS MAIA

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande, como requisito parcial para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Sistemas de Informações e Banco de Dados

DR. ULRICH SCHIEL

(ORIENTADOR)

CAMPINA GRANDE
ABRIL – 2004

Ficha Catalográfica

Maia, Ana Carolina de Jesus

M217S

SEI-Tur: Um Sistema de Criação de Roteiros Turísticos

Dissertação (Mestrado), Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande – PB, Abril de 2004.

118 p. Il.

Orientador: Ulrich Schiel

Palavras-Chaves:

1. Banco de Dados
2. Web Semântica
3. Web Services
4. Sistemas de Recomendação

CDU – ~~681.3.07B~~ 004.65(043)

"SEI-TUR: UM SISTEMA DE CRIAÇÃO DE ROTEIROS TURÍSTICOS"

ANA CAROLINA DE JESUS MAIA

DISSERTAÇÃO APROVADA EM 29.04.2004

PROF. ULRICH SCHIEL, Dr.
Orientador

PROF. CLÁUDIO DE SOUZA BAPTISTA , Ph.D
Examinador

PROF. CARLOS ALBERTO HEUSER, Dr.
Examinador

CAMPINA GRANDE – PB

“Não é mérito o fato de não termos caído e, sim, o de termos levantado todas as vezes que caímos”.

(Provérbio Árabe)

Dedico este trabalho aos meus pais, Antônio Maia e Aparecida, que tanto amo.

Agradecimentos

A Deus, que jamais me deixou desistir.

Ao meu pai, que nunca mediu esforços para que eu chegasse até aqui.

À minha mãe, por seu carinho e alegria.

Aos meus irmãos, pelo companheirismo em todas as horas.

Ao meu professor orientador Ulrich Schiel, por sua dedicação, incentivo e paciência.

Aos meus amigos Pasqueline, Wagner por todos os momentos que vivemos juntos durante estes dois anos, pelas brincadeiras, pelo companheirismo e paciência nos momentos difíceis.

Aos meus amigos Edemberg e Thiago pelo apoio, amizade e momentos de descontração.

À galera do Rubi: Luana, Lidiane (Pipokinha), Thiago (Repolhinho), Ricardo e Daniel.

A todos os amigos que conheci nesta fase de minha vida, em especial a César, Flávia, Roberta, Rohit e aos colegas de mestrado.

Aos meus velhos amigos que sempre torceram por mim: Iraí, Valéria, Liliane, Wendel, Jacqueline e João.

A Aninha, por sua simpatia e dedicação.

Enfim, a todos aqueles que contribuíram na realização deste trabalho, muito obrigada!

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O rápido crescimento e popularidade da World Wide Web tem largamente facilitado o acesso à Informação. Ao mesmo tempo, este crescimento tem dificultado às pessoas as tarefas de coletar, filtrar, avaliar e usar esta grande quantidade de informação.

A informação só passa a ter valor real quando filtrada de acordo com as necessidades do usuário. Sistemas de Recomendação são sistemas de informação que filtram conteúdos relevantes para um determinado usuário baseado em suas preferências.

Para automatizar a recuperação de informações na *Web*, a visão da Web Semântica está buscando definir uma infra-estrutura capaz de permitir que o significado das informações disponíveis na *Web* seja entendível por máquinas *(computer-readable)*. Temos nos deparado com o surgimento de várias tecnologias que possibilitam a comunicação entre computadores na Web, como por exemplo RDF e *Web services*.

Nesta dissertação apresentamos um sistema de recomendação para criação de roteiros turísticos e de negócios, chamado *SEI-Tur* (Serviço de Informações Turísticas). O *SEI-Tur* é um sistema de recomendação que utiliza dados na *Web* e tem a finalidade de sugerir ao usuário itens de roteiro, baseado em suas preferências, e compõe um roteiro de viagens completo, mantendo a consistência entre os elementos do roteiro.

Abstract

The fast growth and popularity of the World Wide Web have widely facilitated the access to information. At the same time, this growth has render more difficult the tasks to collect, to filter, to evaluate and to use this large amount of information.

The information only starts to have real value when filtered according to the necessities of the user. Recommender Systems are information systems that filter contents for a definite user based on its preferences.

To automatize the information retrieval in the Web, the vision of the Semantic Web is searching to define an infrastructure capable to allow that the meaning of the available information in the Web is understandable for machines (computer-readable). We have come across with the rise of some technologies that make possible the communication between computers in the Web, as for example RDF and Web services.

In this thesis we present a recommender system for creation of tourist and business trip plan, called *SEI-Tur* (Serviço de Informações Turísticas). *SEI-Tur* is a recommender system that uses data in the Web and has the purpose to suggest to the user itens of a trip, based on its preferences, and composes a complete script of the trip, keeping the consistency between the elements of the trip.

1. Introdução

Todos os dias, perdemos uma significante quantidade de tempo navegando na *Web*, procurando por informações do nosso interesse. Se por um lado a *Internet* oferece muitas alternativas para o consumidor, essa mesma vantagem, devido ao grande volume de informações, pode se transformar num empecilho. O grande valor da *Web* reside na sua capacidade de proporcionar acesso imediato à informação. Porém essa informação só passa a ter valor real quando filtrada de acordo com as necessidades do usuário.

A utilização de *máquinas de busca*[1] supre parte desta deficiência. Essas máquinas normalmente abrangem um domínio muito vasto de dados dificultando a recuperação de informações relevantes para o usuário.

A rede é usada hoje não apenas para procurar informação. Existe uma aceitação em comprar serviços pela *Internet*. Percebemos a necessidade de um ambiente que agregue informações de um dado domínio e que permita uma comunicação mais personalizada com o usuário para ajudá-lo a obter os resultados desejados. O que as pessoas desejam é que aplicações permitam acessar um conjunto de fontes relacionadas, extrair a informação que elas precisam, e agregar os dados com a finalidade de resolver seus problemas.

Devido ao grande volume de informação, a tarefa de acessar as fontes de dados para extrair informações é uma tarefa que precisa ser delegada às máquinas. Surge a necessidade de que o significado das informações disponíveis na *Web* seja entendível por máquinas *(computer-readable)*.

A visão da *Web* Semântica está buscando definir uma infra-estrutura capaz de possibilitar a comunicação entre computadores na *Web*. Temos nos deparado com o surgimento de várias tecnologias que permitem a recuperação automática de informações na *Web*.

Na área de turismo, os usuários da *Internet* parecem aceitar tornarem-se seus próprios agentes de viagens, organizando suas viagens sozinho. Na *Internet* podemos

[1] Máquinas de busca - são *Web sites* que possuem em seu banco de dados palavras-chave e os links onde estas foram encontradas, ajudando o usuário a encontrar alguma informação que ele esteja procurando.

encontrar um grande número de *sites* de turismo. Para planejar uma viagem, é necessário ir a vários *sites* para obter informações de vôos, hotéis, restaurantes, pontos turísticos, etc. Mesmo os *sites* que oferecem várias destas informações, as oferecem de forma desintegrada, deixando ao usuário a tarefa de organizar as informações recebidas. É um processo bastante tedioso ter que visitar cada uma destas fontes, entrando repetidamente com a mesma informação sobre datas, endereço, destino, etc. Ao invés disso necessitamos combinar os recursos em um simples ambiente integrado, que permita planejar uma viagem como um todo.

Para permitir uma comunicação mais personalizada com o usuário têm-se adotado os chamados Sistemas de Recomendação. Um Sistema de Recomendação irá estudar o perfil dos clientes para tornar o mais fácil possível o processo da compra. Sistemas de Recomendação são definidos como aplicações que *sites* de comércio eletrônico utilizam para sugerir produtos e fornecer aos consumidores informações para facilitar seu processo de tomada de decisão.

Muitos sistemas de turismo, como *Expedia*, *Travelocity* e *Tiscover* têm começado a lidar com planejamento de viagem incorporando sistemas de recomendação. A geração atual de sistemas de recomendação de turismo focaliza a seleção de destinos e não permite ao usuário interativamente compor uma viagem, com a acomodação e atrações adicionais (museu, teatro, etc.).

Esta dissertação tem como foco principal apresentar o desenvolvimento de um Serviço de Informações Turísticas (*SEI-Tur*) que possibilite um acesso unificado a várias fontes de informação ligadas ao turismo e que auxilie um usuário a criar um roteiro turístico de acordo com suas necessidades. O trabalho está inserido em um projeto maior denominado SEI – Serviços Especiais de Informação na Internet, que irá integrar diversos serviços especializados, como SEI-Bib (pesquisa bibliográfica), SEI-Gov (governo eletrônico), SEI-Classi (classificados), SEI-Com (comércio). Destes já está implementado o SEI-Bib.

1.1 Objetivos

O objetivo deste trabalho foi desenvolver um Serviço de Informações Turísticas que permite a um usuário criar um roteiro turístico composto de itens de viagem, hospedagem e programação no local, mantendo a consistência temporal entre os

elementos do roteiro. A programação no local incluí eventos de diversão (cinema, teatro, excursão, show de música e outros), eventos pré-programados (congresso) e opções de alimentação.

O sistema deve fazer sugestões ao usuário sobre os elementos do roteiro, ajudando-o a compor um roteiro turístico completo. As informações são filtradas e ordenadas com base nas preferências do usuário. As preferências do usuário são definidas através de padrões de roteiro, que são utilizados para fazer um casamento de padrões com os dados disponíveis no sistema. Além de sugerir itens separadamente, o sistema cria uma programação no local baseada nas preferências definidas e a sugere ao usuário, possibilitando que este faça os ajustes necessários.

Para compor o roteiro, o *SEI-Tur* combina dados disponíveis na Internet com dados cadastrais previamente armazenados em um banco de dados próprio. Para permitir a interoperabilidade entre sistemas, foram utilizados as tecnologias *Web services* e RDF.

Web Services são partes de software que permitem acessar dados remotos pelo envio de mensagens formatadas em XML sobre a rede. XML não é amarrada a uma particular linguagem de programação ou sistema operacional, desta forma esta pode ser usada como um protocolo para troca de informação entre sistemas heterogêneos.

Resource Description Framework (RDF) estabelece uma infra-estrutura que possibilita a codificação, o intercâmbio e a reutilização de estruturas de metadados[2]. O desenvolvedor de conteúdo e aplicações para a *Web* dispõe de um modelo de especificação de esquemas de metadados. Com isto, há a disponibilização de uma estrutura para a interoperabilidade entre sistemas de informação e seu processamento automatizado dentro do ambiente *Web*.

1.2 Contribuição

As principais contribuições desta dissertação são as seguintes:

- Implementamos um assistente de criação de roteiros turísticos que permite um usuário compor um roteiro formado de itens de viagem, hospedagem e

[2] Metadados – são dados sobre dados. Descrevem os dados, o ambiente onde são manipulados, como são manipulados e para onde são distribuídos.

programação mantendo a consistência temporal entre os elementos do roteiro.

- Criamos um esquema RDF para modelar os eventos utilizados pelo *SEI-Tur* para criar uma programação.
- Definimos padrões para todos componentes de um roteiro que são utilizados para fazer o casamento com os dados acessados pelo *SEI-Tur*.
- Criamos um algoritmo de escalonamento de eventos, utilizado para criar uma programação no local para o roteiro turístico.

1.3 Estrutura da Dissertação

Esta dissertação é constituída de seis capítulos, incluindo esta Introdução. Os capítulos restantes estão organizados da seguinte forma:

- O capítulo 2 apresenta uma visão geral das tecnologias relacionadas com esta pesquisa e cruciais para o entendimento da construção do *SEI-Tur*.
- O capítulo 3 descreve os aspectos relacionados ao funcionamento do *SEI-Tur*. Mostra as camadas que compõem a arquitetura do *SEI-Tur*, detalhando seus componentes; explica como o *SEI-Tur* interage com outros sistemas disponíveis na *Web* para coletar informações; detalha os padrões utilizados para definir as preferências do usuário; e por fim explica as atividades necessárias à tarefa de composição do roteiro turístico.
- No capítulo 4 discutimos aspectos relevantes da análise e do projeto do *SEI-Tur*.
- No capítulo 5 é exibido um estudo de caso.
- O capítulo 6 relata as conclusões obtidas, assim como propostas para trabalhos futuros.

2. Tecnologias e Trabalhos Relacionados

Neste capítulo abordamos as tecnologias utilizadas para o desenvolvimento do sistema proposto neste trabalho de pesquisa. O entendimento delas é essencial para uma melhor compreensão do nosso contexto. Também analisamos, no final do capítulo, os trabalhos relacionados à presente dissertação.

2.1 Sistemas de Recomendação

Sistemas de recomendação são aplicações usadas em *sites* de comércio eletrônico para sugerir produtos interessantes e úteis e fornecer aos consumidores informação para facilitar o processo de tomada de decisão [Schafer *et al*, 2001]. Possuem a função de automatizar o processo de sugestão de itens. São sistemas de informação que filtram conteúdos relevantes para um determinado usuário baseado em suas preferências. Dessa forma, provêem ao usuário uma visão personalizada da informação.

Se por um lado a Internet oferece muitas alternativas para o consumidor, essa mesma vantagem, devido ao grande volume de informações, pode se transformar num empecilho. Um dos grandes valores da Web reside na sua capacidade de proporcionar acesso imediato à informação. Porém essa informação só passa a ter valor real quando filtrada de acordo com as necessidades do usuário.

A solução proposta pelos Sistemas de Recomendação é estudar o perfil do usuário e tornar mais fácil o momento de aquisição de um produto. Através da análise de iterações anteriores e até mesmo de perfis, esses sistemas podem filtrar quais produtos são mais adequados ao usuário que está visitando uma página.

Os Sistemas de Recomendação baseiam-se principalmente em duas técnicas de filtragem de informação: Filtragem Baseada no Conteúdo e Filtragem Colaborativa.

2.1.1 Filtragem Baseada em Conteúdo

Em um sistema baseado no conteúdo, o usuário fornece, de forma implícita ou explícita, suas preferências e restrições e o sistema casa estas descrições com os itens contidos em um catálogo de produtos. Estes sistemas poderiam utilizar o histórico de consultas passadas para construir um perfil do usuário.

Os sistemas baseados no conteúdo utilizam apenas as preferências do usuário. Eles tentam recomendar itens que são similares aos que o usuário gostou no passado. O foco desses sistemas é aprender as preferências do usuário e filtrar dentre os novos itens aqueles que mais se adequarem a estas preferências.

Embora a filtragem baseada no conteúdo venha sendo usada com sucesso em vários domínios, essa técnica apresenta uma série de limitações:

- Os atributos geralmente precisam ser manualmente cadastrados para os itens. Com a tecnologia atual, mídias como som e vídeo apresentam grande dificuldade de serem analisados automaticamente para extração automática de atributos. Muitas vezes não é possível definir os atributos manualmente devido a limitações de recursos.
- As técnicas de filtragem baseadas no conteúdo não têm como encontrar itens que interessariam ao usuário, mas não são parecidos (no conteúdo) com outros itens que ele avaliou, isto é, apenas são encontrados os itens parecidos com os já conhecidos pelo usuário.
- Os métodos baseados no conteúdo não são capazes de avaliá-lo quanto a dimensões subjetivas, como qualidade. Por exemplo, em um sistema que faz recomendações de notícias de um jornal, apesar das pesquisas na área de inteligência artificial, ainda é dificil diferenciar um texto mal escrito de um bem escrito com conteúdos muito semelhantes.

2.1.2 Filtragem Colaborativa

Sistemas de Recomendação Colaborativa coletam avaliações de usuários em produtos oferecidos e informações sobre produtos anteriormente comprados, reconhecem similaridades entre usuários com base em suas avaliações, e geram novas

recomendações baseadas em produtos comprados ou altamente avaliadas por usuários similares.

A técnica de filtragem colaborativa baseia-se no fato de que as melhores recomendações para um indivíduo são aquelas fornecidas por pessoas que possuem gostos similares aos dele (vizinhos próximos).

O perfil de um usuário em um sistema colaborativo consiste tipicamente em um vetor de itens e suas avaliações, sendo constantemente incrementado durante as interações do usuário e o sistema no tempo [Burke, 2002].

Esta técnica possui algumas limitações, das quais destacam-se:

- Recomendação de itens novos: até que um item tenha sido avaliado por um número mínimo de usuários, não é possível recomendá-lo, pois o sistema não tem dados suficientes para prever uma nota para ele.
- Usuário "ovelha negra": caso o usuário em busca de recomendações não tenha vizinhos próximos o suficiente, o sistema apresentará um baixo desempenho, pois as recomendações serão baseadas em usuários que não se parecem muito com ele.
- Número de usuários insuficiente: para ter uma boa performance, um sistema de filtragem colaborativa necessita de uma grande comunidade de usuários, pois de outra forma não haverá vizinhos suficientemente próximos de cada usuário.

Estes sistemas são bons em sugerir novos produtos que o usuário poderia ainda não conhecer. Filtragem Colaborativa é aplicável apenas quando produtos a serem recomendados são padronizados, ou seja, vendidos da mesma forma para vários usuários, e um usuário está interessado em comprar muitos itens do mesmo tipo através do mesmo *site Web* [Ricci *et al*, 2003].

Para vencer estas limitações, vários sistemas têm adotado com sucesso uma estratégia híbrida, combinando a filtragem colaborativa com a baseada no conteúdo. Em [Ricci *et al*, 2003] é proposto uma metodologia combinando as duas abordagens.

2.2 Web Semântica

A maioria do conteúdo disponível hoje na Web está representada através de HTML (Hypertext Markup Language). HTML é uma linguagem básica de marcação utilizada nas páginas da Web [W3C-HTML], e é uma derivação da Standard Generalized Markup Language (SGML), que é um padrão internacional para definição de formatos de representação de texto em meio eletrônico.

HTML foi desenvolvida para a marcação de hipertextos, tanto para os autores dos documentos quanto para os desenvolvedores de *browsers*, os softwares interpretadores dos códigos HTML. A principal limitação apresentada por esta linguagem é a falta de estruturação dos dados de um documento HTML, ou seja, não há distinção entre a codificação de seus componentes básicos (conteúdo, estrutura e apresentação dos dados).

No entanto, a quantidade de informação disponível para os usuários é cada vez maior. Percebe-se a necessidade de delegar a manipulação desta grande quantidade de informação para as máquinas. Dentro deste contexto, surge a necessidade de que as informações disponíveis na Web sejam entendíveis por máquinas *(computer-readable).*

Atualmente, tornou-se uma necessidade, não só recuperar informação armazenada em depósitos de informação disponíveis na Internet, como também de responder com precisão as buscas feitas pelos usuários. Diante disto, tem-se feito um esforço no âmbito de adicionar informação semântica às páginas Web.

Três investigadores, Berners - Lee, James Hendler e Ora Lassila, num artigo publicado na revista Scientific American, com o título *The Semantic Web* [Berners-Lee e*t al*, 2001], falaram de uma Internet em que os computadores não só são capazes de apresentar a informação contida nas páginas Web, como também podem entender o significado dessa informação. Na prática isto significa que as máquinas (os computadores pessoais ou qualquer outro dispositivo conectado à Internet) poderão realizar, quase sem a necessidade de intervenção humana, uma infinidade de tarefas que simplificarão a nossa vida. A Web Semântica não é uma Web à parte, mas sim, uma extensão da atual em que a informação tem um significado bem definido, possibilitando aos computadores e às pessoas trabalharem em cooperação.

Berners - Lee, Hendler e Lassila mostram suas visões para o futuro da Web, através de uma situação cotidiana. Dois irmãos têm que ajustar o seu calendário para

acompanhar a mãe em umas sessões de reabilitação recomendadas pelos médicos. Para isso utilizam um agente de software que revisa as suas agendas para as próximas semanas e combina-as com as horas disponíveis dos centros de reabilitação mais perto e que tenham o seguro médico familiar. Em poucos minutos conseguem programar de novo as suas agendas, incluindo as visitas da mãe e resolvem o problema. Isto foi possível graças à Web Semântica. Como explicam estes três investigadores, "a maior parte do conteúdo que hoje aparece na Internet está projetado para ser lido pelas pessoas e não para serem manipulados por máquinas". Para que isto mude não será necessário que os computadores possuam uma inteligência artificial, será suficiente que os conteúdos semânticos sejam introduzidos nas páginas Web. Assim o que até agora havia sido basicamente um meio de documentos para as pessoas passará a ser um sistema de dados e informação que poderão ser processados automaticamente. Por exemplo, será possível que os agentes de software que perambulam de uma página a outra, possam realizar tarefas sofisticadas para os usuários.

2.2.1 XML

Com a finalidade de flexibilizar a descrição da informação na Web, surgiu a linguagem XML. XML (eXtensible Markup Language) é uma linguagem de marcação apropriada para representação de dados, documentos e demais entidades cuja essência fundamenta-se na capacidade de agregar informações. Foi desenvolvida pelo W3C (World Wide Web Consortium) [W3C], órgão independente que regulamenta padrões e métodos adotados pela Internet.

Comparado ao HTML, o XML oferece uma gama de recursos muito mais ampla e expressiva para descrever o formato das informações na Web. Com XML o usuário pode criar suas próprias *tags* e detalhes de apresentação do conteúdo não são expressos juntos com a representação deste conteúdo.

A seguir descrevemos e exemplificaremos os componentes básicos de um documento XML:

- Elemento

O elemento é descrito por uma marcação inicial (ex: <nome>) e uma marcação final (ex: </nome>). O conteúdo de um elemento é delimitado pela marcação. Cada elemento pode conter texto, conter outros elementos aninhados (subelementos), ter conteúdo misto ou ser vazio. Na FIGURA 2.1 as marcações <hotel> e </hotel> descrevem a estrutura do elemento hotel, o qual possui os subelementos *nome* e *endereço*. O elemento endereço, por sua vez, possui os subelementos *logradouro*, *bairro* e *cidade*.

```
<?xml version="1.0" encoding="UTF-8"?>
<hotel idHotel="H02">
    <nome>Hotel Serrano</nome>
    <endereço>
        <logradouro>rua Tavares Cavalcante, 27</logradouro>
        <bairro>Centro</bairro>
        <cidade>Campina Grande</cidade>
    </endereço>
</hotel>
```

FIGURA 2.1: Exemplo de documento XML.

- Atributo

XML permite associar atributos aos elementos. O valor de um atributo é sempre um texto e deve aparecer entre aspas. Atributos são geralmente usados para especificar propriedades dos elementos. No exemplo da FIGURA 2.1, o elemento hotel possui um atributo *idHotel*, que representa seu identificador.

Um documento XML pode ser associado a uma definição de esquema XML, a qual define a estrutura do documento. Assim, aplicações podem validar seus dados de acordo com um esquema.

A primeira linguagem proposta para definição de esquema XML foi a DTD (Document Type Definition) [Bray et al., 2000].

O principal objetivo de uma DTD é especificar quais elementos são permitidos e em que ordem estes devem aparecer no documento.

A FIGURA 2.2 mostra o exemplo de uma DTD. A declaração <DOCTYPE hotel [<!ELEMENT hotel...>...]> especifica o tipo do documento hotel.

```
<?xml version="1.0" ?>

<!DOCTYPE hotel [
    <!ELEMENT hotel (nome, endereço*)>
    <!ELEMENT nome (#PCDATA)>
    <!ELEMENT endereço (logradouro, bairro, cidade)>
    <!ELEMENT logradouro (#PCDATA)>
    <!ELEMENT bairro (#PCDATA)>
    <!ELEMENT cidade (#PCDATA)>
    <!ATTLIST hotel idHotel ID #REQUIRED >
]>
```

FIGURA 2.2: Exemplo de documento DTD.

Considerando o elemento hotel, a declaração <!ELEMENT hotel (nome, endereço*)> especifica que este elemento contém uma seqüência de um elemento nome e vários elementos endereços.

O símbolo * significa que o elemento hotel possui zero ou mais elementos do tipo endereço. Outros símbolos são permitidos: | (ou), ? (opcional), + (um ou mais). O tipo PCDATA indica que o valor do atributo é texto.

A declaração <!ATTLIST hotel idHotel ID #REQUIRED > especifica a declaração de um atributo denominado *IdHotel* para o elemento *hotel*. O tipo ID permite associar identificadores únicos aos elementos. REQUIRED especifica que o valor do atributo é obrigatório.

A DTD possui algumas limitações como tipos de dados, herança e integração com espaços de nomes XML. Para sobrepor as limitações da DTD, outras linguagens estão sendo propostas. XML Schema surge como o sucessor da DTD.

A linguagem XML Schema introduz novos construtores que fazem com que esta linguagem seja mais expressiva do que o DTD. Com isso, XML Schema pode ser utilizada em uma maior variedade de aplicações. E permite que o usuário defina novos tipos de dados e relacionamentos de tipo/subtipo entre tipos.

A FIGURA 2.3 mostra um exemplo de um documento XML Schema.

```
<?xml version="1.0" encoding="UTF-8">

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Declaração de Elementos -->
    <xsd:element name="listaDeHoteis">
        <xsd:complexType>
                <xsd:element name="hotel" type="Thotel"
                minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
```

```
    <xsd:complexType name="Thotel">
        <xsd:sequence>
                <xsd:element name="nome" type="xsd:string"/>
                <xsd:element name="endereço" type="TEndereço"
                minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="idHotel" type="xsd:string"/>
    </xsd:complexType>
    ...
</xsd:schema>
```

FIGURA 2.3: Exemplo de documento XML Schema.

Elementos são declarados utilizando o construtor *element*. Os principais atributos deste construtor são: *name*, que associa o elemento declarado a um nome; *type*, que atribui um tipo ao elemento; e *minOccurs* e *maxOccurs*, que especificam a restrição mínima e máxima de ocorrência do elemento, respectivamente.

Considerando a seguinte declaração <xsd:element name="endereço" type="TEndereço" minOccurs="1" maxOccurs="unbounded"/>, temos que o elemento denominado endereço é do tipo TEndereço e ocorre uma (minOccurs="1") ou várias vezes (maxOccurs="unbounded").

Tipos podem ser simples ou complexos. No exemplo acima, o elemento nome possui um tipo simples (string), o qual é definido pelo XML Schema. Tipos complexos podem ser definidos utilizando o construtor complexType. Tais definições geralmente contêm um nome, declarações de elementos e declarações de atributos. O tipo complexo Thotel é definido como uma seqüência dos elementos *nome* e *endereço*, e do atributo *idHotel*.

2.2.2 RDF

O RDF (Resorce Description Framework) [W3C-RDF] é a recomendação W3C para a definição e uso de metadados. Metadados são dados sobre dados. Descrevem os dados, o ambiente onde são manipulados, como são manipulados e para onde são distribuídos.

Diferentemente de XML Schema ou DTD, voltados para tipagem e definição estrutural, a proposta de RDF envolve metadados conceituais, que têm por objetivo tornar informações disponíveis para aplicações diferentes daquelas para as quais estas informações foram originalmente criadas.

O RDF é uma plataforma para descrição de metadados. Expressa o significado dos dados permitindo interoperabilidade na Web e servindo como um meio de integração entre diferentes domínios descritos por RDF [Lassila & Swich,1999].

São três os componentes de RDF: o modelo de dados, a sintaxe para intercâmbio de metadados e o esquema.

2.2.2.1 Modelo de Dados RDF

O modelo de dados do RDF fornece uma estrutura abstrata e conceitual para descrição de recursos, definindo uma sintaxe neutra como forma de representação. O modelo de dados do RDF permite especificar o relacionamento entre entidades e prover interoperabilidade estrutural porém, não fornece mecanismos para a declaração de propriedades e nem para a definição de relacionamentos entre tais propriedades e outros recursos. Para que estas habilidades sejam realizadas, é necessário aplicar o RDF Schema, descrito mais adiante.

Uma expressão RDF consiste de três partes, que descrevem relacionamentos entre recursos em termos de propriedades e valores nomeados:

- Recurso: tudo que é descrito através de expressões RDF é chamado de recurso. Um recurso pode ser tanto um documento eletrônico, uma coleção, uma página Web, como também um objeto não acessível diretamente na Web (ex.: livro impresso). Recursos são sempre nomeados através de URIs (Uniform Resource Identifier) que permitem a introdução de identificadores para qualquer entidade.
- Propriedade: é um atributo ou característica que descreve o recurso. Uma propriedade representa também o relacionamento entre recursos.
- Sentença: corresponde à associação de um recurso específico, uma propriedade e o valor dessa propriedade para esse recurso. Uma declaração é dividida em: sujeito, predicado e objeto, onde o objeto dessa declaração pode ser um outro recurso ou um literal. Desta forma, um objeto pode ser especificado por uma URI, uma cadeia de caracteres ou outro tipo de dado primitivo definido pela XML.

O modelo RDF pode ser representado através de uma tripla consistindo de um recurso, uma propriedade para esse recurso, e o valor desta propriedade.

Recurso	Propriedade	Valor
http://www.dsc.ufcg.edu.br/seitur/Eventos#Ev_0010	Data	"28/03/2004"

TABELA 2.1: As partes de uma sentença RDF.

Outra forma de representar uma sentença é através de um grafo de arestas rotuladas, conforme ilustrado na FIGURA 2.4.

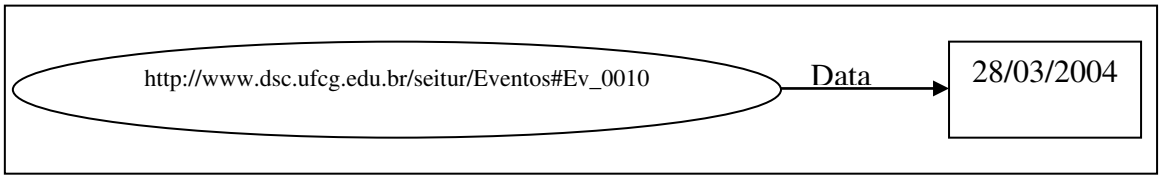

FIGURA 2.4: Representação do modelo RDF através de um grafo.

2.2.2.2 Sintaxe RDF

Para que as sentenças RDF possam ser compreendidas pelas máquinas, se faz necessário o uso de uma linguagem que consiga expressar o modelo de dados RDF. Como a linguagem XML tem se tornado um padrão de interoperabilidade sintática, a especificação do modelo de dados inclui uma codificação XML para o RDF.

Duas sintaxes em XML são propostas para expressar os modelos RDF [Marino, 2001]: serializada, que expressa toda a potencialidade do modelo RDF; e abreviada, que inclui construtores adicionais para expressar de forma mais compacta o modelo RDF. A FIGURA 2.5 ilustra como a sentença da TABELA 2.1 pode ser expressa de forma serializada em XML.

```
<?xml version="1.0"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#">

    <rdf:Descritpion about="http://www.dsc.ufcg.edu.br/seitur/Eventos#Ev_0010">
        <s:Data>28/03/2004</s:Data>
    </rdf:Descritpion>

</rdf:RDF>
```

FIGURA 2.5: Serialização em XML de descrições RDF.

O conteúdo entre os elementos <rdf:RDF> e </rdf:RDF> em um documento XML representa uma instância do modelo de dados RDF. O elemento *Description* é utilizado para agrupar múltiplas sentenças em um mesmo recurso, ou seja, fornece uma maneira de atribuir um nome a um recurso apenas uma vez para várias sentenças. A identificação do recurso a ser descrito pode ser feita através do atributo *about* ou através do atributo *ID*. Estes atributos são mutuamente exclusivos. A diferença entre eles é que o atributo *ID* sinaliza a criação de um novo recurso e o atributo *about* refere-se a um recurso já existente [Braganholo, 2001].

O prefixo *s* refere-se a um *namespace* específico definido através da propriedade xmlns:s="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#". Isto significa que o elemento *Data* é definido no *namespace s*, identificado pela URI http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#".

Um *namespace* é uma coleção de nomes, identificados por uma referência URI, que são usadas em documentos XML como tipos de elementos e nomes de atributos. Segundo [Bray, 1999] *namespaces* são formas de distinguir nomes usados em documentos XML, independente de onde eles vêm.

Sentenças RDF também podem ser expressas na forma abreviada (FIGURA 2.6).

```
<?xml version="1.0"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#">

    <rdf:Description about="http://www.dsc.ufcg.edu.br/seitur/Eventos#Ev_0010"
    s:Data="28/03/2004"/>

</rdf:RDF>
```

FIGURA 2.6: Serialização em XML abreviada de descrições RDF.

.

A forma abreviada tem a vantagem de ser mais próxima dos formatos XML mais comuns para representação de dados, o que facilita a migração de aplicações existentes que utilizam XML.

2.2.2.3 RDF Schema

O modelo de dados RDF define um modelo simples para descrever inter-relacionamentos entre recursos em termos de propriedades e valores. Entretanto, não

fornece mecanismos para definir as propriedades (por exemplo, título, autor, tamanho, cor, etc.), para definir os relacionamentos entre as propriedades e recursos, e nem para definir os tipos de recursos sendo descritos (por exemplo, livros, páginas Web, pessoas, etc.) [Izeki, 2001]. O RDF Schema [W3C-RDFSchema] veio para suprir esta necessidade, sendo utilizado para escrever vocabulários a serem usados em instâncias RDF.

A FIGURA 2.7 mostra um exemplo de RDF Schema para eventos turísticos.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF xmlns:Evento="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="
    http://www.w3.org/TR/2004/REC-rdf-schema-20040210#">
  <rdfs:Class
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento"
      rdfs:label="Evento">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
        19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema"
      rdfs:label="Cinema">
    <rdfs:subClassOf
        rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#E
        vento" />
  </rdfs:Class>
  <rdf:Property
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#nomeEve
      nto" rdfs:label="nomeEvento">
    <rdfs:domain
        rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#E
        vento" />
    <rdfs:range rdf:resource="http://www.w3.org/TR/2004/REC-rdf-schema-
        20040210#Literal" />
  </rdf:Property>
  ...
</rdf:RDF>
```

FIGURA 2.7: Um exemplo de RDF Schema para eventos turísticos.

O vocabulário do RDF Schema é definido em um *namespace* denominado *rdfs* e identificado pela URI http://www.w3.org/TR/2004/REC-rdf-schema-20040210#. A especificação também utiliza o prefixo *rdf* para referir-se ao espaço de nomes do modelo e sintaxe RDF, indicado pela URI http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Evento é uma classe, denotada através de *rdfs:Class*. *Cinema* é uma subclasse de *Evento*. Essa relação é denotada por *rdfs:subClassOf*. A propriedade *rdfs:domain* é utilizada para indicar as classes que as propriedades podem ser usadas. Por exemplo, a propriedade *nomeEvento* só pode ser usada na classe *Evento*. No entanto, como a classe

Cinema é subclasse de *Evento*, esta herda todas as características inerentes à classe mãe. Sendo assim, *Cinema* possui a propriedade *nomeEvento*.

O valor da propriedade *nomeEvento* é uma literal. Essa relação é denotada pela restrição *rdfs:range*. O rdfs:Literal é uma classe definida no RDF Schema, que denota um conjunto de literais, declarado como uma classe.

A seguir resumimos alguns dos principais elementos do RDF Schema:

- *rdfs:subPropertyOf*: A propriedade rdfs:subPropertyOf é utilizada para especificar que uma propriedade é especialização de outra. Se uma propriedade P2 é subPropertyOf de outra propriedade mais geral P1, e se um recurso A tem uma propriedade P2 com um valor B, isto implica que o recurso A também tem uma propriedade P1 com valor B.
- *rdfs:Class, rdf:type e rdfs:subClassOf:* Classes são recursos que definem um conjunto de recursos através da propriedade rdf:type (nas instâncias o valor da propriedade rdf:type é a classe). Quando um recurso é uma classe, o valor da propriedade rdf:type é rdfs:Class. Por outro lado, o valor da propriedade rdf:type é rdfs:Property para todos os recursos que são propriedades. Classes são estruturadas em uma hierarquia do mesmo modo que as propriedades. Do mesmo modo que rdfs:subPropertyOf indica que uma propriedade é subpropriedade de outra, rdfs:subClassOf indica que uma classe é subclasse de outra. Toda classe é subclasse de rdf:Resource.
- *rdfs:domain e rdfs:range:* São usados para restringir o conjunto de recursos que podem possuir uma determinada propriedade (domain) e o conjunto de valores válidos para uma propriedade (range).
- *rdfs:Literal:* RDF Schema define um recurso rdfs:Literal, que denota um conjunto de literais, declarado como uma classe.
- *rdfs:seeAlso:* utilizada para fornecer informações adicionais sobre um determinado recurso.

Segundo [Braganholo, 2001], RDF Schema é um sistema de classes extensível e genérico que pode ser utilizado como base para esquemas de um domínio específico. Esses esquemas podem ser compartilhados e estendidos através de refinamento de subclasses.

2.3 Web Services

Um *Web service* é uma aplicação que pode ser publicada, localizada e chamada através da Internet. É identificado por um URI (Universal Resource Identifier), cujas interfaces e implementações são capazes de serem definidas, descritas e localizadas utilizando-se a linguagem XML e suas ferramentas. Um *Web service* deve ser capaz de interagir com outras aplicações através da troca de mensagens baseadas em XML utilizando os protocolos de comunicação padrão existentes na Internet [Austin *et al*, 2002].

Para explicar o que é um *Web service*, vamos analisar um cenário típico que pode ocorrer na *Web*.

Tomemos o exemplo de um sistema de uma agência de turismo virtual, que disponibiliza a criação automática de um roteiro turístico de acordo com as preferências do cliente. A criação do roteiro turístico necessita utilizar outros sistemas disponíveis na Internet. Como exemplo, um dos passos para criação do roteiro é a alocação de um hotel em um determinado período de tempo. O sistema deve consultar vários hotéis obtendo informações sobre quartos, seus recursos e preços, verificar a disponibilidade de quartos para um determinado período e fazer reservas. Uma solução seria o sistema de criação de roteiros turísticos copiar em sua base de dados todas as informações referentes aos hotéis e quartos disponíveis. É evidente que isto acarretaria em muitos problemas de atualização, já que a base de dados do sistema de hotel é freqüentemente modificada.

Diante disto, é desejável que uma aplicação (por exemplo, um hotel), que gera dados de interesse para outra aplicação (por exemplo, uma agência de turismo), pudesse ser utilizada automaticamente por esta outra aplicação. A forma tradicional de comunicação via Internet, utilizando navegadores que na maioria dos casos apenas interpretam documentos HTML é um fator limitante a essa comunicação entre aplicações. Sendo assim, passaram a surgir novas tecnologias, numa tentativa de melhor identificar e resolver esse problema.

O uso de *Web services* permite a interoperabilidade entre aplicações. A consulta a informações na Web está largamente difundida e é realizada facilmente por pessoas. É preciso então, permitir que programas possam realizar essas mesmas consultas sem o auxílio de seres humanos.

Um *Web service* é uma aplicação que aceita solicitações de outros sistemas através da Internet [Kao, 2001], baseado em padrões para facilitar a comunicação entre as máquinas.

Uma melhor solução para o nosso exemplo seria criar um *Web service* disponível no hotel, que recebesse uma mensagem solicitando informações de disponibilidade de quartos e retornasse uma outra mensagem com os dados solicitados.

2.3.1 Visão Geral

Web services são serviços oferecidos por uma aplicação para outras aplicações via World Wide Web [Sun-WS, 2003]. É um middleware[3] emergente distribuído que utiliza um protocolo simples baseado em XML que permite às aplicações trocarem dados ao longo da Web. Os usuários de serviços não precisam saber como tal serviço foi implementado, ou em que linguagem. Devem apenas saber como enviar e receber mensagens.

A arquitetura do Web services possui três componentes: o provedor de serviços, o cliente e o registro de serviços. Estes componentes interagem entre si através de três operações fundamentais: publicar, procurar e *bind* (mapear a interface e utilizar). A interação entre eles está ilustrada na FIGURA 2.8.

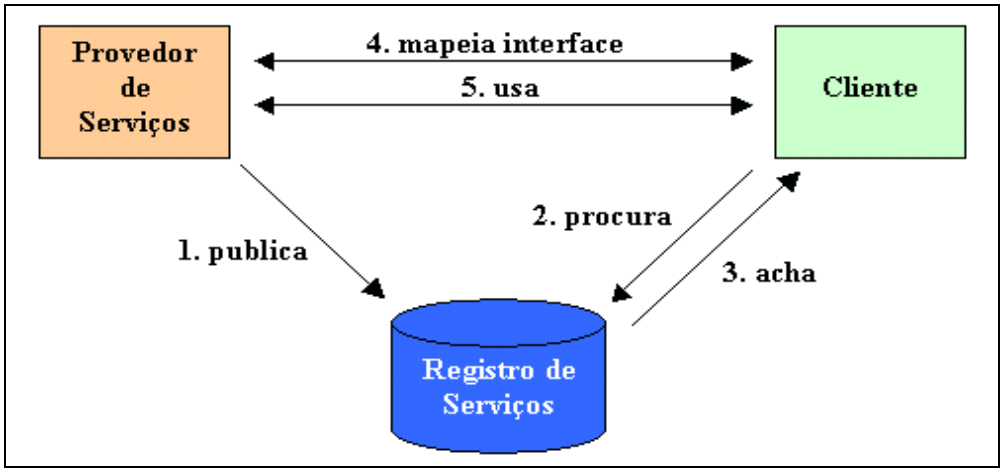

FIGURA 2.8: Arquitetura de Web services.

[3] Middleware - Software que conecta duas aplicações separadas e transfere dados entre elas.

1. O Provedor de Serviços publica seus serviços para os Clientes em um Registro de Serviços.
2. Um Cliente acessa o Registro de Serviços a procura de um serviço. Os Registros de Serviços podem ser considerados como as páginas amarelas dos Web services.
3. O Cliente acha a descrição do serviço, informações de onde encontrá-lo e como chamá-lo.
4. Amarrações são feitas para que o Cliente e o Provedor de Serviços possam se comunicar.
5. O Cliente e o Provedor de Serviços trocam mensagens.

Web service é ideal para integração de sistemas internos ou para fazer um link com as aplicações existentes na rede. É uma tecnologia baseada em padrões abertos, recomendados pela W3C.

Os pontos chaves da padronização de Web services são: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) e Universal Description, Discovery, and Integration (UDDI). Todos eles são baseados em XML.

2.3.2 SOAP

SOAP (*Simple Object Access Protocol*) [W3C-SOAP] é um protocolo baseado em XML para troca de informações em um ambiente distribuído. SOAP descreve um formato de troca de mensagens para comunicação entre aplicações, com duas características principais: simplicidade ao estabelecer formato XML para as mensagens e extensibilidade ao não colocar restrições às implementações dos serviços.

Ao invés de definir um novo protocolo, SOAP trabalha em cima de um protocolo de transporte já existente, tal como HTTP [Curbera *et al.*, 2002], para fazer chamada a procedimentos remotos, sem impor restrições de implementação para os pontos de acesso. O uso de HTTP para transporte ocorre porque estes protocolos podem

atravessar *firewalls*[4] e podem trabalhar em ambientes heterogêneos [Fremantle *et al.*, 2002].

Utilizando padrões como XML em suas especificações e o protocolo HTTP na comunicação de dados, o SOAP vem sendo utilizado como o protocolo para a troca de informações em ambientes descentralizados e distribuídos. Uma especificação SOAP contém três partes principais: *envelope*, *header* e *body*. A FIGURA 2.9 mostra a estrutura de uma mensagem SOAP.

```
<SOAP:Envelope xmlns:SOAP=
        "http://schemas.xmlsoap.org/soap/envelope/"
        SOAP:encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP:Header>
        <!— content of header goes here —>
    </SOAP:Header>

    <SOAP:Body>
        <!— content of body goes here —>
    </SOAP:Body>

</SOAP:Envelope>
```

FIGURA 2.9: Estrutura de uma mensagem SOAP.

O elemento *envelope* é composto de dois elementos: *XMLnamespace*, que define um conjunto de nomes para tipos de elementos XML e nomes de atributos, e *encodingStyle*, identifica o tipo de dados reconhecido pelas mensagens SOAP.

O atributo *header* pode ser usado para que, ao passar por diversos nós quando viaja pela Web, sejam feitos processamentos com a mensagem, como autorização e gerenciamento.

O elemento *body* é onde estão armazenadas as informações necessárias para que o destinatário possa processar o pedido e retornar uma resposta.

2.3.3 WSDL

SOAP oferece comunicação básica, mas não diz quais mensagens devem ser trocadas para se interagir com um serviço. Desenvolvedores precisam informar onde um

[4] Firewalls - dispositivos utilizados na proteção de redes de computadores contra ataques externos, dificultando o trânsito de invasores entre as redes.

serviço está localizado e como ele pode ser acessado. O Web Services Description Language (WSDL) provê esta informação.

WSDL é usada para responder três perguntas sobre um web service: o que é o serviço, onde encontrá-lo e como chamá-lo. WSDL define os serviços como um conjunto de *endpoints*, isto é, pontos de acesso na rede [W3C-WSDL].

Um documento WSDL é dividido em dois grupos: uma definição abstrata e uma descrição concreta. As seções abstratas definem mensagens SOAP de maneira independente de linguagem e de plataforma. Seus elementos são: *types*, *message* e *portType*. As seções concretas são específicas de localização, máquina ou linguagem. Os elementos são *binding* e *service*.

A FIGURA 2.10 mostra um exemplo de um documento WSDL, descrevendo os serviços disponibilizados pelo *Web service* de um hotel. Neste caso, o *Web service* oferece a operação de reservar um quarto.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="hotelSerrano" targetNamespace="urn:Foo"
     xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
   <types/>

   <message name="reservaQuartoInput">
      <part name="cliente" type="xsd:string" />
      <part name="quarto" type="xsd:string" />
      <part name="entrada" type="xsd:dateTime" />
      <part name="saida" type="xsd:dateTime" />
    </message>
   <message name="reservaQuartoOutput">
      <part name="result" type="xsd:boolean" />
    </message>

   <portType name="HotelIF">
      <operation name="reservaQuarto">
         <input message="tns:reservaQuartoInput" />
         <output message="tns:reservaQuartoOutput " />
      </operation>
    </portType>

   <binding name="HotelIFBinding" type="tns:HotelIF">
      <operation name="reservaQuarto">
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               use="encoded" namespace="urn:Foo" />
         </input>
         <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               use="encoded" namespace="urn:Foo" />
         </output>
         <soap:operation soapAction="" />
       </operation>
```

```
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    </binding>

    <service name="HotelSerrano">
        <port name="HotelIFPort" binding="tns:HotelIFBinding">
            <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                location="http://localhost:8080/hotelSerrano/hotelSerrano" />
        </port>
    </service>
</definitions>
```

FIGURA 2.10: Exemplo de um documento WSDL.

O elemento *types* contém definições dos tipos de dados. No exemplo da FIGURA 2.10, a marcação <types/> é vazia, pois utilizamos apenas tipos simples definidos no espaço de nomes xmlns:xsd="http://www.w3.org/2001/XMLSchema".

O elemento *message* descreve os dados trocados entre os provedores de Web services e os clientes. Cada método Web tem duas mensagens: *input* e *output*. O input descreve os parâmetros do método Web; o output descreve o retorno do método Web. Cada mensagem contém zero ou mais parâmetros <*part*>, uma para cada parâmetro do método Web. Cada parâmetro está associado com um tipo definido em um elemento <types> ou em um esquema externo. No exemplo acima o serviço possui as mensagens *reservaQuartoInput* e *reservaQuartoOutput*.

O elemento *portType* indica quais são as operações disponibilizadas pelo serviço, enquanto que o elemento *operation* especifica se as mensagens são de entrada ou de saída. Cada *operation* representa um padrão de troca de mensagens que o Web service suporta.

Um *binding* é uma implementação de uma interface definida pelo elemento *portType*, utilizando algum protocolo de comunicação. Nesta seção, são especificados os detalhes da transmissão de dados, como o formato das mensagens e o protocolo utilizado na comunicação.

Um elemento *service* possui vários subelementos *port*. Cada elemento *port* está relacionado a um elemento *binding* particular, através do atributo *binding*, especificando qual interface e qual protocolo de comunicação estão sendo utilizados nessa implementação. Dependendo do protocolo de comunicação utilizado, um elemento *port* pode possuir sublementos específicos, indicando qual URL deve ser utilizada para se acessar a implementação do serviço. Por exemplo, na FIGURA 2.10 estamos utilizando o protocolo SOAP, definimos o elemento *soap:address*, que irá conter informações

sobre a URL pela qual um determinado elemento *port* de um serviço poderá ser acessado.

2.3.4 UDDI

UDDI oferece aos usuários um unificado e sistemático caminho para encontrar provedores de serviços através de um registro de serviços centralizado.

Um catálogo de serviços deve permitir a publicação e recuperação de informações sobre serviços conhecidos, bem como a descoberta de novos serviços através de descrições informadas por usuários.

UDDI possui três tipos de informação sobre Web services:

- Páginas brancas: incluem nome e detalhes de contato das entidades de negócios;
- Páginas amarelas: provêem a categorização baseado no negócio e nos tipos de serviços;
- Páginas verdes: incluem dados técnicos sobre os serviços oferecidos pelas entidades de negócios. Estas incluem detalhes técnicos suficientes para se contratar um dado serviço;

Existem quatro grandes estruturas que armazenam os dados no UDDI: businessEntity, businessService, bindingTemplate e tModel.

- businessEntity: descreve a empresa que publica o Web service. Contém informações sobre a organização como nome, descrição e contatos.
- businessService: descreve os Web services oferecidos pela empresa.
- bindingTemplate: contém detalhes técnicos sobre onde está localizado o Web service.
- tModel: tem como objetivo geral o armazenamento de metadados sobre o serviço. Geralmente, esta estrutura é utilizada para armazenar também, a URL do documento WSDL que descreve o serviço, através de um subelemento de nome overviewURL.

2.4 Trabalhos relacionados

O projeto *Harmonise* [Dell'Erba *et al*, 2002] propõe uma solução para o problema da interoperabilidade no domínio de turismo, criando o chamado "*harmonisation space*", o qual permite que participantes troquem informações sem alterar seus formatos proprietários de dados. A interoperabilidade é resolvida pelo fornecimento de reconciliação semântica entre diferentes sistemas de informação turísticas considerando uma ontologia do domínio. Esta solução permite aos participantes visualizarem informações de outros participantes como extensões de seus próprios sistemas de informação, sem se preocuparem com as diferenças de nomes e representação de dados.

O *Harmonise* possibilita que fornecedores sejam capazes de integrar-se com outros fornecedores para fornecer novas combinações de serviços. No entanto, o foco deste projeto está em solucionar o problema da interoperabilidade entre sistemas.

Na área de sistemas de recomendação para viagem e turismo, as duas tecnologias mais bem sucedidas são *TripHop* [Delgado *et al*, 2002] (usado pelo www.ski-europe.com, entre outros) e o *VacationCoach* [VacationCoach, 2001] (usado por travelocity.com).

Ambos tentam imitar a interatividade observada em sessões com agentes de viagens tradicionais, quando um usuário procura por sugestões de destinos para suas férias. *VacationCoach* explora o perfil do usuário pedindo explicitamente que este classifique-se em um perfil, o qual deduz preferências implícitas que o usuário não fornece. O usuário pode ainda fornecer informações precisas sobre seu perfil preenchendo um formulário apropriado.

O *TripHop* usa uma técnica mais sofisticada para reduzir as entradas do usuário. Ele deduz a importância dos atributos que o usuário não menciona explicitamente, verificando a importância atribuída por usuários similares em consultas anteriores.

Nenhum destes sistemas permite a criação de um roteiro definido pelo usuário, combinando viagens, localizações pra visitar, hospedagem e eventos adicionais (cinema, teatro, etc.).

Outro sistema de recomendação é o *ITR (Intelligent Travel Recommender)* [Ricci *et at*, 2002], desenvolvido por uma equipe de pesquisadores do Laboratório de

Pesquisa em Turismo e Comercio Eletrônico (*eCTRL*) do *ITC-irst*[5] na Itália. O sistema permite ao usuário filtrar informação e selecionar itens para planejar uma viagem. Este permite a seleção de destinos de viagem, atividades e atrações, e dá suporte à construção de um plano de viagens personalizado. Os planos são armazenados em uma memória de casos, e são explorados posteriormente para ordenar os itens de viagens extraídos de um catálogo. Comportamentos de usuários anteriores, com comportamento similar, são trocados com sessões similares. No entanto, o *ITR* não trata do planejamento temporal ou escalonamento de eventos no roteiro.

Um planejador de entretenimentos noturnos é apresentado no projeto *GraniteNights* [Grimnes *et al*, 2003] do departamento de *Computing Science*, da Universidade de Aberdeen, na Escócia. *GraniteNights* é uma aplicação multi-agentes que permite um usuário programar sua noite na cidade de Aberdeen. O sistema possui uma lista de eventos, incluindo tempo inicial e sua duração. Um evento é uma visita a um barzinho, um restaurante ou um cinema, com restrições opcionais em cada evento. O sistema recupera uma lista de possíveis eventos que casam com as restrições e gera um conjunto de possíveis planos para a noite, escolhendo tempos e eventos de acordo com os critérios especificados pelo usuário. Uma programação é então apresentada ao usuário, que tem a opção de aceitar ou perguntar pela próxima solução. O *GraniteNights* usa três agentes de informação: *PubAgent* e *RestaurantAgent* recuperam os dados de arquivos RDF estáticos, e *CinemaAgent* recupera os dados de arquivos RDF gerados dinamicamente de páginas *Web* convencionais. A extração é feita via simples casamento de padrões de expressões regulares, e precisa ser reescrita se a estrutura da fonte HTML for alterada.

Para criar um conjunto de programações válidas, o *GraniteNights* implementa um algoritmo usando o *SICStus Prolog*. No entanto, o *GraniteNights* apenas mostra ao usuário todas as combinações possíveis de horários para os eventos solicitados por ele.

Nenhum dos sistemas encontrados na literatura busca manter a consistência temporal entre os elementos do roteiro. O *SEI-Tur*, por sua vez, além de integrar viagem, hospedagem e programação no local, procura organizar os elementos do roteiro de maneira que estes obedeçam regras temporais bem definidas (seção 3.4). O *SEI-Tur* implementa um algoritmo de escalonamento de eventos, descrito na seção 3.5.2.1, com a finalidade de construir uma programação, tentando buscar a melhor combinação

[5] Istituto Trentino di Cultura - Istituto per la Ricerca Scientifica e Tecnologica

possível de eventos, a fim de maximizar o número de eventos escalonados e levando em conta todos os dias do roteiro.

2.5 Considerações finais

Neste capítulo, apresentamos uma visão geral dos conceitos que envolvem a nossa pesquisa. Inicialmente falamos dos aspectos relacionados aos sistemas de recomendação. Em seguida, descrevemos os conceitos relacionados à Web Semântica, que proporcionam subsídios para a recuperação de informações. E por fim, descrevemos os trabalhos relacionados encontrados na literatura.

O embasamento teórico, descrito de forma bem sucinta nesse capítulo, foi indispensável para que pudéssemos traçar os caminhos que deveríamos seguir nas etapas posteriores (projeto e implementação) até a conclusão do nosso objetivo com êxito.

3. SEI-Tur: Serviço de Informações Turísticas

O sistema descrito neste capítulo é um Sistema de Recomendação, que tem a finalidade de auxiliar um usuário na tarefa de criar um roteiro de viagens. Sistemas de Recomendação são aplicações que fornecem conselhos aos usuários sobre produtos que poderiam interessá-los. Diferentemente de outros Sistemas de Recomendação ligados ao turismo, o SEI-Tur além de fornecer conselhos sobre os itens do roteiro separadamente, busca oferecer ao usuário uma programação completa, que é composta de itens que poderiam interessá-lo. O usuário interage com o sistema e especifica suas preferências através da criação de padrões. Estes padrões serão utilizados para fazer um casamento com os dados disponíveis no sistema, é o que chamamos casamento de padrões.

O *SEI-Tur* auxilia o usuário a criar um roteiro de viagens que é composto de vários itens de roteiro. São considerados quatro tipos de itens: viagem, hospedagem, alimentação e eventos. Os eventos e os itens de alimentação formam uma programação. A programação considera tanto eventos de diversão (cinema, teatro, etc.), quanto eventos como uma programação pré-definida para a viagem (congresso, simpósio, festival, etc.). O sistema busca manter a consistência temporal dos itens incluídos no roteiro, não permitindo situações como, uma hospedagem ocorrer após a estadia do turista na cidade. As condições temporais devem ser mantidas pelo módulo responsável pela composição do roteiro.

Neste capítulo, apresentamos detalhadamente as funcionalidades do SEI-Tur. Descrevemos a arquitetura do sistema e seus componentes, mostramos como ele recupera as informações disponíveis na *Web*, especificamos os padrões utilizados para criar o roteiro e, por fim, detalhamos como é feita a criação do roteiro.

3.1 Arquitetura do Sistema

A arquitetura do *SEI-Tur* viabiliza a composição de seus serviços com a adoção de múltiplas camadas formadas por componentes especializados. A FIGURA 3.1 ilustra os componentes da arquitetura.

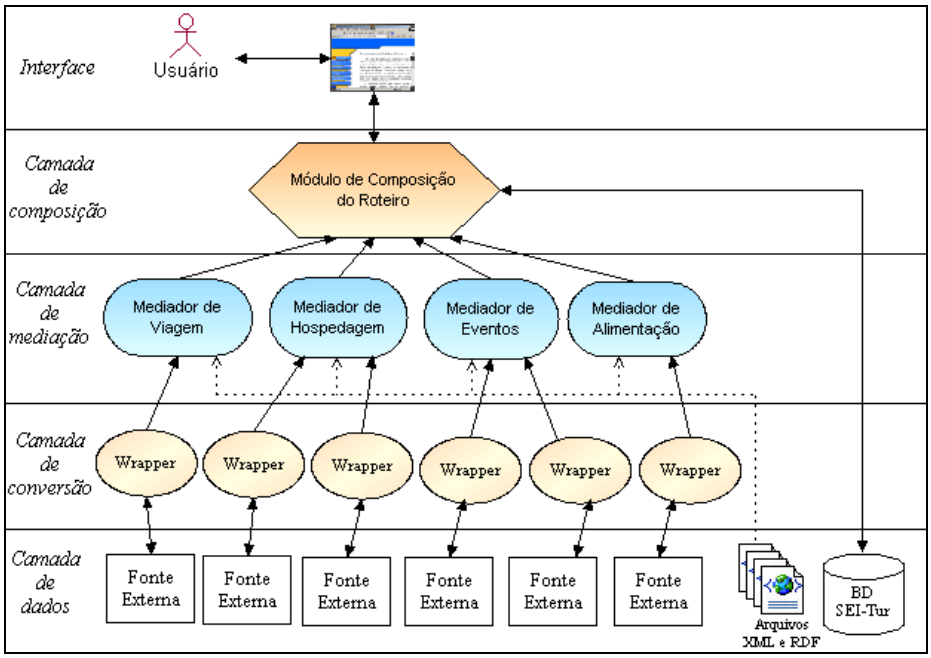

FIGURA 3.1: Arquitetura do SEI-Tur.

A composição do roteiro é executada combinando quatro tipos de serviços: viagem, hospedagem, eventos e alimentação. Um usuário interage através de uma interface *Web*, fornecendo dados e preferências para a composição do roteiro. Trata-se de instruções e restrições de viagem, hospedagem, eventos e alimentação. Tais informações formam um padrão de roteiro que será entregue ao módulo de composição do roteiro para que este possa recuperar dados que casem com o padrão especificado. A recuperação dos dados é feita acionando os respectivos módulos mediadores responsáveis por cada elemento do roteiro. Estes possuem uma interface única para acessar serviços semanticamente equivalentes. O padrão de roteiro é então decomposto em padrões específicos para cada elemento do roteiro, que são entregues aos mediadores responsáveis. Os mediadores acessam os *wrappers* a fim de recuperar os dados necessários. Os *wrappers* fazem o mapeamento do formato específico de cada aplicação para o formato do mediador. Após a recuperação dos dados, o mediador é responsável por filtrar os dados, eliminando aqueles que não atendem ao padrão. Os

dados resultantes são ordenados de acordo com o padrão e sugeridos ao usuário. Os selecionados pelo usuário são então entregues ao módulo de composição do roteiro para que este possa organizar os dados no roteiro completo. O roteiro completo é então criado, permitindo que o usuário aceite o roteiro ou faça alterações.

Interface: quando um usuário fornece suas preferências (intervalo de tempo disponível para a viagem, custo máximo, recursos desejados para hospedagem, estilos musicais, tipos de comida, etc.), a interface cria um padrão de viagem, um padrão de hospedagem e um padrão de programação. O padrão de programação é composto de padrões de eventos e um padrão de alimentação. Padrões são detalhados na seção 3.4. Os padrões criados são enviados para o módulo de composição do roteiro.

Módulo de Composição do Roteiro: responsável por compor um roteiro de viagens de acordo com os padrões fornecidos pelo usuário através da interface. O módulo de composição do roteiro encaminha cada padrão ao seu respectivo mediador. O primeiro item a ser inserido no roteiro é a viagem. O módulo de composição do roteiro aciona o mediador de viagem, que então retorna uma lista ordenada de itens que casam com o padrão de viagem solicitado. O módulo de composição do roteiro, através da Interface, sugere uma lista ordenada de viagens de ida e uma lista de viagens de volta ao usuário. As viagens de ida e volta selecionadas são então inseridas no roteiro. O módulo de composição do roteiro calcula o tempo de estadia na cidade e aciona o mediador de hospedagens para que este retorne uma lista de hospedagens a ser sugerida ao usuário. Da mesma forma, ocorre para eventos e programação. Neste caso, ao invés de sugerir uma lista de itens, o módulo de composição do roteiro cria uma programação composta de eventos e opções de alimentação e a sugere ao usuário. Para compor a programação, um algoritmo de escalonamento de eventos é utilizado e será detalhado na seção 3.5.2.

Mediadores: um mediador é responsável por interagir com as fontes de informação. O mediador recebe um padrão do Módulo de Composição do Roteiro, acessa as fontes, recupera os dados necessários e filtra os dados com base no padrão. Os dados que casam com o padrão são colocados em uma lista ordenada e enviados ao módulo de composição do roteiro.

Wrappers: são componentes responsáveis pela conversão do formato de dados das fontes externas para o formato do mediador.

Camada de dados: para criar roteiros turísticos, o *SEI-Tur* utiliza dados de outros sistemas disponíveis na Web. Na arquitetura da FIGURA 3.1, denominamos estes sistemas de Fontes Externas. As Fontes Externas podem ser páginas HTML ou Web services. Além disso, dados sobre o usuário, suas preferências e as informações do roteiro criado são armazenados em um banco de dados próprio. Os catálogos contendo informações de serviços de hotéis, viagens, eventos e alimentações são descritos em arquivos XML. Já os dados recuperados na *Web* são armazenados em arquivos RDF. O armazenamento em arquivos, como forma de persistência, foi escolhido pelo fato destes dados serem utilizados apenas como leitura, sendo atualizados com pouca freqüência.

3.2 Recuperando dados em páginas HTML

3.2.1 Justificativa

Algumas informações bastante úteis para compor um roteiro de viagens podem ser encontradas em páginas *HTML* disponíveis na *Web*. Estas páginas contêm informações estáticas, ou seja, não são geradas dinamicamente a cada solicitação de um usuário, podendo ser atualizadas apenas em intervalos de tempo maiores. Informações sobre cinemas (horários de filmes, gêneros, locais onde estão sendo exibidos, etc.), shows (horários de shows, gêneros musicais, etc.), teatro (horários de peças em cartaz, etc.) e excursões (passeios, tipo de atividades, preços, etc.) são alguns exemplos que podem ser encontrados na *Web*.

No entanto, estas informações estão disponíveis de forma não estruturadas. Muitas pesquisas têm sido feitas na tentativa de converter dados de páginas HTML em dados estruturados.

Diversas ferramentas, denominadas extratores, têm sido construídas. Estas permitem um usuário especificar o que deve ser extraído das fontes Web.

A extração e conversão geralmente são feitas através de casamento de padrões de expressões regulares e precisam ser reescritas se a estrutura das fontes HTML for alterada.

Em nosso, protótipo decidimos utilizar uma estratégia mais simples e fácil de ser implementada, já que o foco do nosso trabalho está na composição de roteiros utilizando casamento de padrões.

A estratégia utilizada consiste em definir um esquema que será utilizado pelos fornecedores interessados em disponibilizar seus dados ao *SEI-Tur*.

Esta abordagem, no entanto, acarreta no problema das fontes de dados terem que se adequar ao esquema definido pelo *SEI-Tur*, porém a arquitetura em camadas proposta permite que novas abordagens de recuperação da informação sejam incorporadas, adaptando-se apenas alguns módulos do sistema.

Como foi mostrado no capítulo anterior, RDF tem demonstrado ser a linguagem mais promissora para possibilitar a interoperabilidade de sistemas no ambiente *Web*.

Para alguns domínios já existem esquemas *RDF* visando atender determinadas necessidades, como o padrão de metadados *Dublin Core*[6], que é utilizado para a catalogação de conteúdo.

No protótipo implementado, utilizamos esquemas para modelar dados de eventos e restaurantes. Para restaurantes utilizamos um esquema (http://sf.us.agentcities.net/ontologies/restaurant.daml) desenvolvido por membros do projeto Agentcities[7].

Em relação a eventos turísticos que ocorrem em uma cidade, encontramos um padrão também desenvolvido no projeto Agentcities. O esquema desenvolvido (http://sf.us.agentcities.net/ontologies/shows.daml) modela shows que são classificados em cinema, teatro e musical. Porém, ele não modela eventos compostos e não permite definir o intervalo em um evento.

Devido à ausência destes requisitos, decidimos criar um esquema *RDF*, fornecendo um conjunto básico de atributos para descrever eventos que ocorrem em uma cidade e podem interessar ao turista na hora de criar seu roteiro de viagens.

Empresas que desejam disponibilizar seus dados devem descrever seus dados em RDF obedecendo ao esquema definido pelo *SEI-Tur*.

---

[6] Dublin Core – é uma lista de quinze elementos de metadados principais criados pelos participantes do OCLC/NCSA *Metadata Workshop* (março de 1995) com o objetivo de definir um conjunto básico de atributos que sirvam para identificar todos os recursos disponíveis na *Web*. O OCLC é a sigla do *Online Computer Library Centre*, e NCSA é a sigla do *National Center for Supercomputing Applications*.

[7] http://www.agentcities.org

Optamos por utilizar RDF porque esta linguagem satisfaz o grau de interoperabilidade desejado para o sistema. Está fora do escopo deste trabalho resolver problemas de interoperabilidade semântica entre fontes de dados.

RDF e RDF schema aplicam-se à interoperabilidade estrutural e casos não muito complexos de interoperabilidade semântica [Santos, 2002]. A interoperabilidade estrutural refere-se ao modelo de dados adotado para definir a estrutura dos metadados.

3.2.2 Descrição

Antes de explicarmos como os dados são recuperados na Web pelo *SEI-Tur*, faremos algumas considerações de como criar estas fontes de dados, contendo descrições RDF, a serem acessadas pelo *SEI-Tur*.

O esquema de classes mostrado na FIGURA 3.2 mostra a estrutura desenvolvida para modelar eventos. Os elementos da UML foram mapeados para RDF com a ferramenta *Protégé* [Protégé, 2000], que foi concebida para desenvolver ontologias e aquisição de conhecimento, podendo ser adaptada para editar modelos em diferentes linguagens da *Web* Semântica.

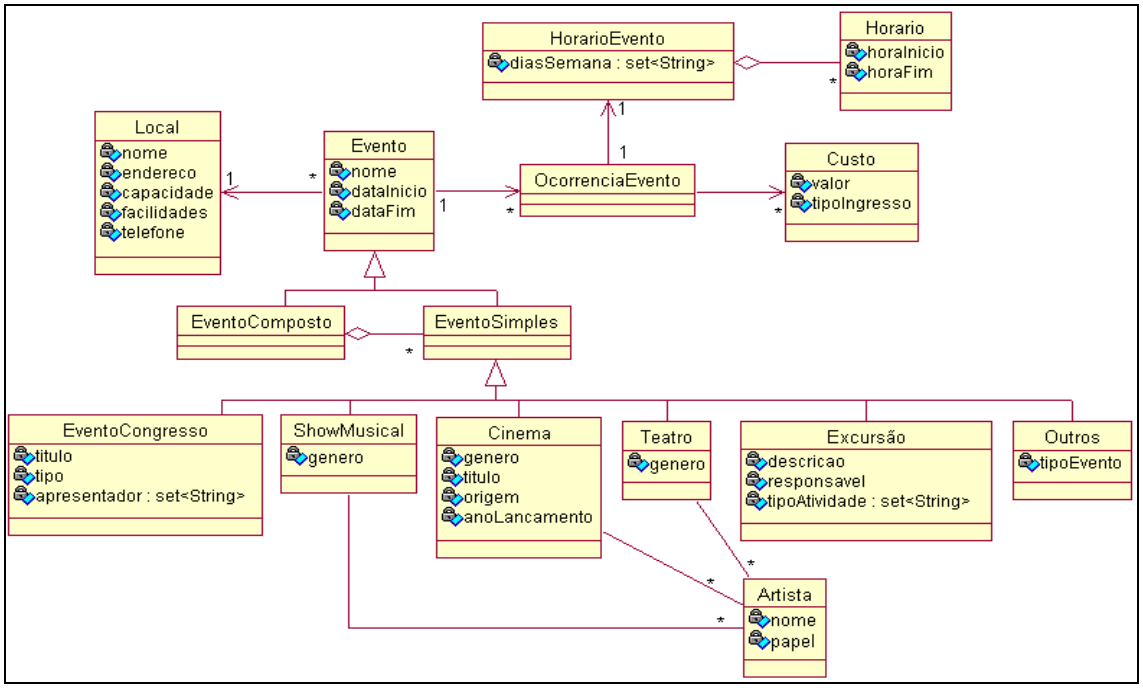

FIGURA 3.2: Diagrama de classes de Eventos

A ferramenta possui uma interface gráfica, que permite a criação de conceitos do domínio sob uma estrutura de árvore, organizados numa hierarquia de subclasses. Como o diagrama UML acima não é uma estrutura em árvore, foi necessário criar uma hierarquia para representar as instâncias das classes do diagrama. Em [Braganholo, 2001] são feitas algumas considerações que facilitaram o mapeamento de UML para RDF Schema. Depois de criados os conceitos, o esquema *RDF* é gerado automaticamente pela ferramenta. Um trecho do esquema é mostrado na FIGURA 3.3. O esquema completo pode ser encontrado no APÊNDICE A ou em http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF xmlns:Evento="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:a="http://protege.stanford.edu/system#"
     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
   <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#Evento"
        rdfs:label="Evento">
       <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
           19990303#Resource" />
    </rdfs:Class>
   <rdfs:Class
        rdf:about="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#EventoSimples"
        rdfs:label="EventoSimples">
       <rdfs:subClassOf
           rdf:resource="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#Evento"
           />
    </rdfs:Class>
    <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#Cinema"
        rdfs:label="Cinema">
       <rdfs:subClassOf
           rdf:resource="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#EventoSi
           mples" />
    </rdfs:Class>
   <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#nomeEvento"
        a:maxCardinality="1" rdfs:label="nomeEvento">
       <rdfs:domain
           rdf:resource="http://www.dsc.ufcg.edu.br/~carol/schemas/Evento#Evento"
           />
       <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
           19990303#Literal" />
    </rdf:Property>
  ...
</rdf:RDF>
```

FIGURA 3.3: Trecho do esquema RDF do domínio Eventos.

As aplicações que desejam integrar seus dados sobre eventos ao *SEI-Tur* devem utilizar o esquema na representação de suas instâncias. As instâncias devem ser embutidas na *tag head* do código *HTML*. A forma como as aplicações fazem isto é dependente de cada aplicação. A FIGURA 3.4, mostra um exemplo de uma página HTML contendo sentenças RDF.

```
<html>
<head>
<title>Filmes em Cartaz - Campina Grande</title>
<rdf:RDF xmlns:Evento="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
  ...
  <Evento:Cinema rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento_00074"
Evento:anoLancamento="2003" Evento:generoFilme="Comedia" Evento:nomeEvento="Os Normais"
Evento:origem="Brasil" Evento:titulo="Os Normais" rdfs:label="Os Normais">
    ...
  </Evento:Cinema>
...
</rdf:RDF>
</head>
<body>
<body text="#000099" bgcolor="#FFFFCC" link="#000099" vlink="#000099" alink="#000099">
<table BORDER=0 CELLSPACING=0 CELLPADDING=0 COLS=1 WIDTH="600" >
...
</body>
</html>
```

FIGURA 3.4: Exemplo de uma página HTML contendo instâncias RDF.

Para que estas aplicações sejam acessadas pelo *SEI-Tur*, elas precisam ser previamente cadastradas no sistema. O *SEI-Tur* possui então um catálogo contendo a URL de todas as aplicações que devem ser acessadas a fim de recuperar informações sobre eventos. O módulo do *SEI-Tur* responsável pela leitura das páginas HTML é o Mediador de Eventos. A FIGURA 3.5 ilustra o funcionamento deste módulo.

```
public void extraiDadosRDF(String pesq_url){
...
URL myurl = new URL(pesq_url);
BufferedReader in = new BufferedReader(new
InputStreamReader(myurl.openStream()));
...
model = new ModelMem();
model.read(in, RDFS.getURI());
iter = model.listStatements();
while (iter.hasNext()) {...}
}
```

FIGURA 3.5: Trecho de código utilizado para extrair RDF de uma URL.

A seguir (FIGURA 3.6), vemos como o *SEI-Tur* interage com aplicações *Web* para extrair os dados RDF.

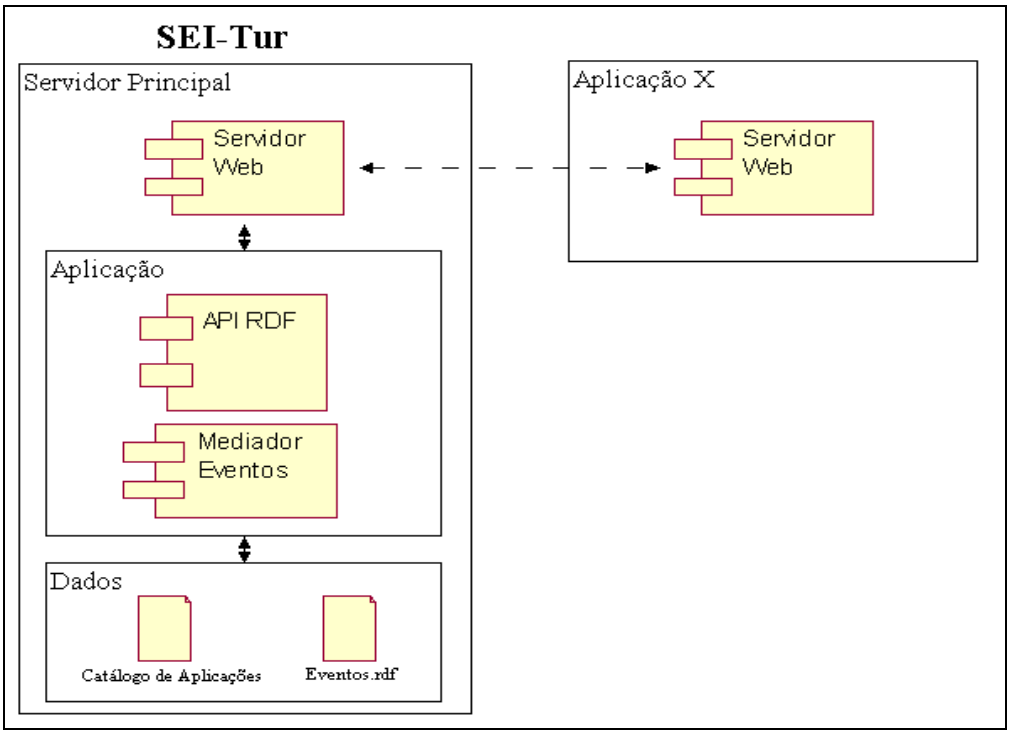

FIGURA 3.6: Interação entre o SEI-Tur e Aplicações Web.

Periodicamente, o *SEI-Tur* deve atualizar os eventos disponíveis para a composição do roteiro. O módulo responsável pela recuperação de informações sobre eventos é o Mediador de Eventos. Este consulta o catálogo de Aplicações, que possui uma lista de URLs, previamente cadastradas no sistema, onde podem ser encontradas informações sobre eventos.

A manipulação dos dados RDF é tratada por uma API RDF. Utilizamos a *API RDF JENA*[8], uma API RDF para a linguagem Java, desenvolvida pela *Hewlett-Packard Company*.

O Mediador de Eventos acessa cada URL, captura o modelo RDF embutido no *tag head* e submete-o à API RDF. A API RDF fornece triplas (sujeito, predicado e objeto) do modelo RDF.

As triplas RDF capturadas pelo Mediador de Eventos são incluídas em um arquivo denominado *Eventos.rdf*, o qual reúne todos os eventos descritos pelas aplicações contidas no catálogo.

[8] http://www.hpl.hp.com/semweb/jena-top.html

Se a estrutura da fonte HTML for alterada, o arquivo *Eventos.rdf* precisa ser atualizado. Por questões de performance, o RDF deve ser extraído uma vez por dia e armazenado localmente.

Quando um usuário acessa o sistema, solicitando a criação de um roteiro turístico, o Mediador de Eventos é ativado para que os eventos contidos no arquivo *Eventos.rdf* sejam recuperados. O mediador utiliza o *framework Castor*[9] para criar os objetos contendo os eventos do arquivo *Eventos.rdf*. O *Castor* permite transformar os dados contidos em um modelo de objetos Java em um documento *XML* e vice-versa.

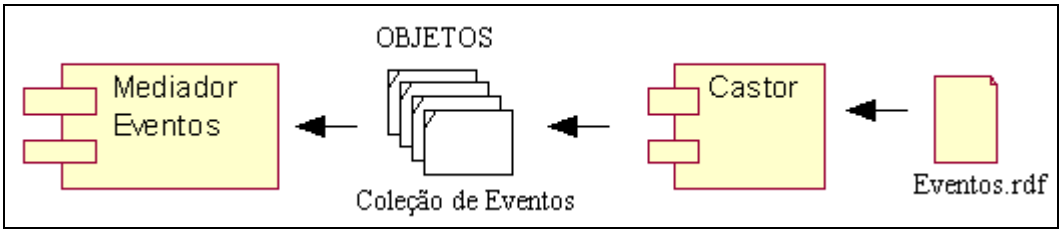

FIGURA 3.7: Instanciação dos Eventos.

O Mediador de Eventos, então, possuiu uma coleção de eventos que serão manipulados na criação do roteiro.

3.3 Interagindo com Web Services

3.3.1 Justificativa

Muitas vezes não é possível apenas definir um esquema e esperar que todas as fontes de dados o utilizem. Na grande maioria dos casos, cada aplicação possui seu próprio esquema.

No entanto, uma forma destas aplicações poderem ser integradas ao SEI-Tur é utilizando a tecnologia de *Web services*. Como foi dito anteriormente (capítulo 2), um *Web service* é uma aplicação que aceita solicitações de outros sistemas através da Internet, sempre baseado em padrões para facilitar a comunicação entre as máquinas. Esses padrões são baseados em *XML*, isto é o que vai permitir uma grande portabilidade de dados.

[9] http://castor.exolab.org

No *SEI-Tur*, utilizamos a tecnologia de *Web services* para acessar aplicações de hotéis e de reservas de passagens aéreas.

Algumas considerações na implementação do protótipo são levantadas a seguir:

- Viagem: para recuperar dados de viagem utilizamos um *Web Service* (netViagens.com[10]) que fornece as informações requeridas em XML. Ele fornece informações sobre vôos de várias companhias aéreas, de acordo com origem, destinos, data de partida e data de retorno especificados como entrada.
- Hospedagem: como não encontramos Web Services de hospedagem com as características desejadas, resolvemos simular o funcionamento destes. Criamos então três sistemas Web de hotéis e disponibilizamos suas funcionalidades através de arquivos WSDL.

3.3.2 Funcionamento

Todas as hospedagens utilizadas pelo *SEI-Tur* devem ser previamente cadastradas. Um arquivo contendo informações como nome, localização, categoria, tipo de hospedagem, ponto de acesso, etc, é mantido pelo sistema. Um trecho deste arquivo é mostrado na FIGURA 3.8. O *CatalogoDeHospedagens.xml* define apenas as características da hospedagem. Os dados são lidos utilizando-se o *framework Castor* e são criados objetos de hospedagem.

```
...
<Hospedagem:Hospedagem
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Hospedagem#Hospedagem_00013"
      Hospedagem:categoria="CincoEstrelas"
      Hospedagem:endPointAddress="http://localhost:8080/hotel/hotelSerrano"
      Hospedagem:nomeHospedagem="Hotel Serrano" Hospedagem:tipo="Hotel"
      rdfs:label="Hotel Serrano">

      <Hospedagem:telefone>3413131</Hospedagem:telefone>
            <Hospedagem:endereco rdf:resource=
            "http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Hospedagem#Hospedagem_00
            014"/>
      <Hospedagem:facilidades>ArCondicionado</Hospedagem:facilidades>
      <Hospedagem:facilidades>Estacionamento</Hospedagem:facilidades>
      <Hospedagem:facilidades>Piscina</Hospedagem:facilidades>
      <Hospedagem:facilidades>Restaurante</Hospedagem:facilidades>

</Hospedagem:Hospedagem>
...
```

FIGURA 3.8: O arquivo CatalogoDeHospedagens.xml

[10] http://ws.netviagens.com/webservices/AirFares.asmx

No entanto, quando se deseja saber se existem quartos disponíveis para uma determinada hospedagem, é necessário utilizar as operações disponibilizadas pelo *Web service*. Desta forma, o *SEI-Tur* irá interagir com a aplicação da hospedagem, acessar os dados internos da aplicação, verificando se existem quartos disponíveis.

Para criar um roteiro turístico, o *SEI-Tur* necessita recuperar os quartos disponíveis de todas as hospedagens cadastradas. Então precisa interagir com cada *Web service* disponível no sistema. Esta interação é mostrada na FIGURA 3.9.

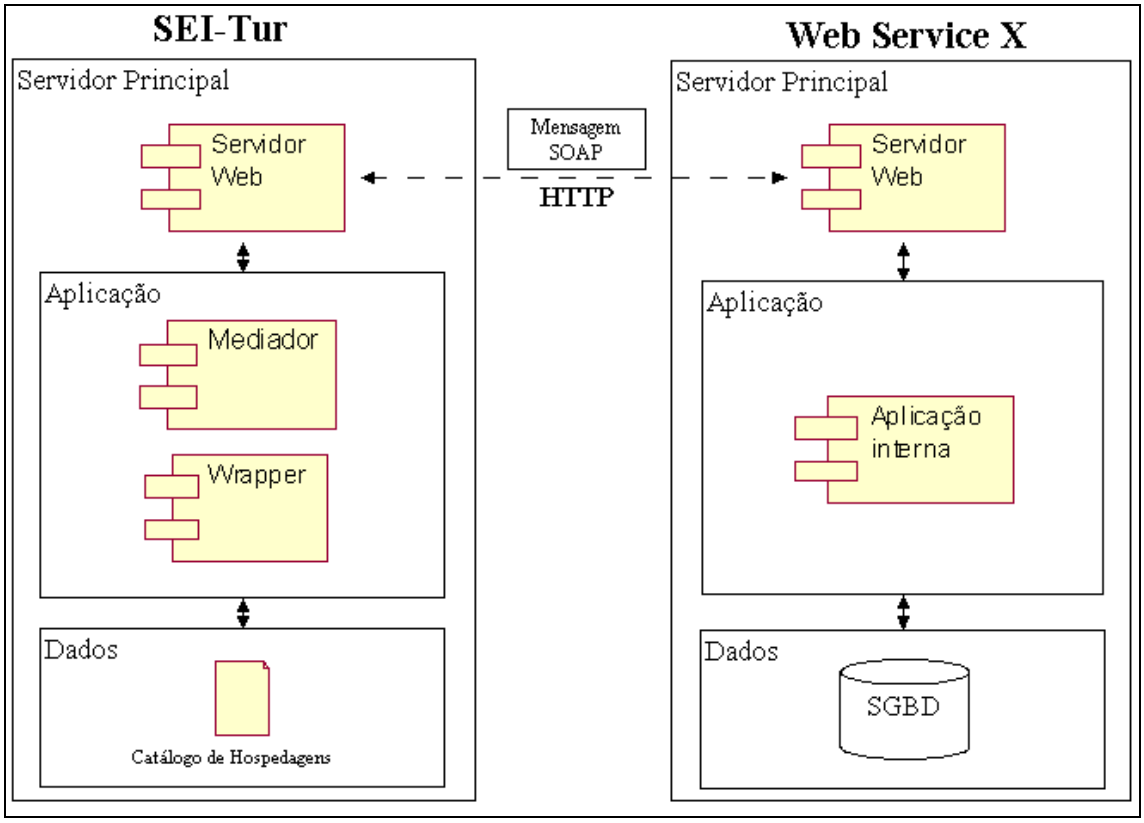

FIGURA 3.9: Interação entre o SEI-Tur e um Web Service.

Em nosso protótipo, utilizamos o pacote *Java Web Service Developer Pack* (JWSDP) desenvolvido pela *Sun Microsystems*. Este permite a geração de documentos WSDL, chamadas de métodos em serviços disponíveis usando RPC (Chamadas de Procedimentos Remotos), envio e recebimento de mensagens com o uso de SOAP e facilidade de publicação do serviço nos *service providers* existentes.

A interação do *SEI-Tur* com os *Web services* é feita através do mecanismo RPC, que possibilita um cliente executar chamadas a procedimentos localizados em

servidores remotos. O pacote JWSDP utilizado pelo *SEI-Tur* possui a API JAX-RPC para esta finalidade. No RPC baseado em XML, uma chamada a um procedimento remoto é usada tendo como base o protocolo SOAP. As chamadas e o seu retorno são transmitidos como mensagens SOAP sobre o HTTP.

Mostraremos agora como o *SEI-Tur* interage com um *Web service* cadastrado em seu catálogo de hospedagens. Entre os *Web services* disponíveis no sistema, utilizaremos o *Web service* hotelSerrano para exemplificar:

O *Web service* hotelSerrano dispõe das seguintes funções básicas:

- Consultar a disponibilidade de quartos no hotel;
- Reservar um quarto;

Um *Web service* define suas *interfaces* usando *WSDL*. Um documento *WSDL* indica quais são as operações de um serviço e onde elas podem ser localizadas.

A seguir, mostramos o arquivo *WSDL*, publicado pelo *Web service* hotelSerrano, a fim de disponibilizar suas interfaces.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="hotelSerrano"
    targetNamespace="urn:Foo">
  <types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-
        enc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace="urn:Foo">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"
          />
      <complexType name="ArrayOfstring">
        <complexContent>
          <restriction base="soap11-enc:Array">
            <attribute ref="soap11-enc:arrayType"
                wsdl:arrayType="string[]" />
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </types>
  <message name="HotelIF_disponibilidade">
    <part name="entrada" type="xsd:dateTime" />
    <part name="saida" type="xsd:dateTime" />
  </message>
  <message name="HotelIF_disponibilidadeResponse">
    <part name="quartos" type="tns:ArrayOfstring" />
  </message>
  <message name="HotelIF_reservaQuarto">
    <part name="cpfCliente" type="xsd:string" />
    <part name="codTipoQuarto" type="xsd:string" />
```

```
        <part name="entrada" type="xsd:dateTime" />
        <part name="saida" type="xsd:dateTime" />
        <part name="numCartao" type="xsd:string" />
        <part name="titularCartao" type="xsd:string" />
        <part name="dataExpiracaoCartao" type="xsd:dateTime" />
    </message>
    <message name="HotelIF_reservaQuartoResponse">
        <part name="reservaOK" type="xsd:boolean" />
    </message>
    <portType name="HotelIF">
        <operation name="disponibilidade" parameterOrder="entrada saida">
            <input message="tns:HotelIF_disponibilidade" />
            <output message="tns:HotelIF_disponibilidadeResponse" />
        </operation>
        <operation name="reservaQuarto" parameterOrder="cpf codTipoQuarto
             entrada saida numCartao titularCartao dataExpiracaoCartao">
            <input message="tns:HotelIF_reservaQuarto" />
            <output message="tns:HotelIF_reservaQuartoResponse" />
        </operation>
    </portType>
    <binding name="HotelIFBinding" type="tns:HotelIF">
        <operation name="disponibilidade">
            <input>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding
                    /" use="encoded" namespace="urn:Foo" />
            </input>
            <output>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding
                    /" use="encoded" namespace="urn:Foo" />
            </output>
            <soap:operation soapAction="" />
        </operation>
        <operation name="reservaQuarto">
            <input>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding
                    /" use="encoded" namespace="urn:Foo" />
            </input>
            <output>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding
                    /" use="encoded" namespace="urn:Foo" />
            </output>
            <soap:operation soapAction="" />
        </operation>
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
            style="rpc" />
    </binding>
    <service name="HotelSerrano">
        <port name="HotelIFPort" binding="tns:HotelIFBinding">
            <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                location="http://localhost:8080/hotelSerrano/hotelSerrano" />
        </port>
    </service>
</definitions>
```

FIGURA 3.10: Arquivo WSDL de um Web service de hotel.

O documento mostra que o serviço possui duas mensagens: disponibilidade e reservaQuarto. O elemento *portType* indica quais são as operações disponibilizadas pelo serviço, enquanto que o elemento *operation* especifica se as mensagens são de entrada

ou saída. No elemento *binding* são indicados os esquemas que devem ser usados para as operações que foram especificadas no *portType*, além do tipo e da maneira como devem ser transportados os dados. O elemento *port* indica onde o serviço pode ser localizado na rede.

Para permitir que o *SEI-Tur* acesse os procedimentos remotos é necessário gerar classes *stubs* com base no documento WSDL do *Web service*. Os *stubs* são objetos locais que representam o serviço remoto.

Depois da geração das classes *stubs* é que podemos construir programas Java que permitem a chamada a procedimentos remotos.

As operações disponibilizadas pelos diferentes *Web services* de hospedagem possuem sintaxe diferente. Os parâmetros exigidos, assim como os tipos de retorno, são alguns exemplos de problemas encontrados. Por isso verificamos a necessidade de construir uma camada de software responsável pela conversão destes dados para o formato do *SEI-Tur*. Cada *Web service* utilizado pelo *SEI-Tur* deve possuir um *wrapper* responsável por acessar o *Web service* e interagir com ele. O *wrapper* conhece as operações disponíveis e como acessá-las.

Na FIGURA 3.11, mostramos um trecho de como é feita a conversão dos dados do formato específico do *Web service* para o formato do *SEI-Tur*.

```
Package hospedagem;

import javax.xml.rpc.Stub;
import java.util.*;

public class WrapperHotelSerrano {

	public HotelIF hotel;

	public Collection obtemQuartosDisponíveis(DataHora entrada, DataHora saida){
	   String[] retorno = null;
	   Collection resultado = new Vector();

	   Calendar parametro1 = entrada.toCalendar();
	   Calendar parametro2 = saida.toCalendar();

	   if(hotel==null){
	            obtemInterface();
	   }

	   try{
	            retorno = hotel.disponibilidade(parametro1, parametro2);
	   }
	   catch (Exception ex) {
	   ex.printStackTrace();
	}

	   for(int i = 0; i<retorno.length; i++){
	            Quarto quarto = new Quarto(retorno[i]);
	            resultado.add(quarto);
```

```
      }
         return resultado;
       }

       public void obtemInterface(){
       try {
          Stub stub = createProxy();
          stub._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
             "http://localhost:8080/hotelSerrano/hotelSerrano");
          hotel = (HotelIF)stub;
       } catch (Exception ex) {
          ex.printStackTrace();
       }
       }

   private static Stub createProxy() {
      return (Stub)(new HotelSerrano_Impl().getHotelIFPort());
   }
}
```

FIGURA 3.11: Exemplo de um Wrapper.

O *Web service* hotelSerrano possui uma operação denominada *disponibilidade*, que recebe como parâmetro dois objetos da classe *Calendar*. No entanto, o SEI-Tur manipula datas através da classe *DataHora*. Então o método *obtemQuartosDisponíveis* recebe dois parâmetros do tipo *DataHora* e converte-os para objetos da classe *Calendar*. Da mesma forma o retorno da operação disponibilidade deve ser convertida para uma coleção (*Collection*) de quartos.

Os *Mediadores* são responsáveis por acessar todos os *wrappers* e prover um único ponto de acesso aos *Web services*.

3.4 Definição de Padrões de Roteiro

Quando um usuário entra no sistema e deseja criar um roteiro de viagens, antes ele precisa informar suas preferências em relação à hospedagem, viagem, eventos e alimentação.

Para isto, definimos um conjunto de atributos que auxiliarão o sistema a manipular estas preferências definidas pelo usuário. Este conjunto de preferências denominamos padrões.

O usuário define padrões para os diversos componentes do roteiro, a serem utilizados pelo *SEI-Tur* para selecionar os dados que melhor casam com suas preferências.

Veremos inicialmente algumas notações utilizadas para definir os padrões.

3.4.1 Definição do tempo

Primeiramente, precisamos definir como será tratado o tempo em nosso sistema. O tempo pode ser representado através de duas formas: por um instante, que determina um único momento; e por um período, que pode ser representado por um intervalo de tempo entre duas datas diferentes [Allen, 1983].

Para nossos propósitos, definimos os seguintes elementos temporais:

T = um conjunto ordenado de instantes t.
Todo subconjunto conexo de T é denominado um **intervalo** I, ou seja
$I \subseteq T$ é um intervalo se, somente se, $\forall\ t_1, t_2$ (se $t_1, t_2 \in I$ então $\forall\ t\ (t_1 \leq t \leq t_2 \Rightarrow t \in I)$
Seja INT = {Intervalos de tempo} = $\{I_1, I_2,..., I_n\}$, definimos duas funções i,f: INT $\rightarrow$ T que determinam o início e o fim do intervalo, ou seja
$\forall\ I \in$ INT temos, $\forall\ t \in I\ (i(I) \leq t \leq f(I))$
Denotamos um intervalo de tempo I = [i(I), f(I)]

Allen em sua Teoria Geral de Ações e Tempo [Schiel, 96] define os seguintes predicados temporais:

3.4.1.1 Eventos

Um evento p tem um tempo determinado I, com início i(I) e fim f(I). Portanto, em um subintervalo não pode ocorrer o evento todo.

ocorre(p, I) $\Leftrightarrow \exists$ I' $\subset$ I ~ocorre(p, I')

Eventos (E) aqui são mais genéricos que os eventos (*ev*) de uma programação (cinema, teatro, congresso, etc.). Eventos são fenômenos que devem ocorrer em um certo tempo, conhecido antecipadamente. Portanto, classificamos como eventos (E) a Viagem de Ida (V_{ida}), a Viagem de Volta (V_{volta}) e os eventos da programação (*ev*).

Definimos os eventos do roteiro E_R como uma coleção de eventos tal que:

$E_R = \{E_k\}_{k=1,2,...n}$ tal que $\forall\ i, j \Rightarrow t(E_i) \cap t(E_j) = \varnothing$

Um evento pode ser contínuo (EC) ou descontínuo (ED). Sendo, $E = EC \cup ED$.

- EC = {eventos contínuos}

$\forall$ ec $\in$ EC $\Rightarrow$ t(ec) = I, onde I $\in$ {intervalo de tempo}.

- ED = eventos descontínuos

$\forall$ ed $\in$ ED $\Rightarrow$ t(ed) = $\{I_1, I_2, ... I_n\}$, onde $I_i \in$ {intervalo de tempo}.
$\forall$ i, j $\Rightarrow I_i \cap I_j = \varnothing$

Com as definições de eventos contínuos e descontínuos acima, analisamos que:
$V_{ida} \in EC$, $V_{volta} \in EC$
$P_g = \{ev_i\}_{i=1,2,...n} \Rightarrow P_g \in ED$, onde P_g é a Programação.
$E = EC \cup ED = \{ V_{ida}, V_{volta}, P_g \}$

3.4.1.2 Processos

Um processo p é dito estar *ocorrendo* em um intervalo I apenas se existe um subintervalo I' de I tal que p está ocorrendo em I'.

ocorrendo(p, I) $\Leftrightarrow \exists$ I'$\subseteq$ I $\Rightarrow$ ocorrendo(p, I')

No roteiro, caracterizamos hospedagem (H) e alimentação (A) como processos (P), pois devem ocorrer durante um certo tempo, mas sua hora exata não é conhecida antecipadamente.
P = {H, A}

3.4.2 Definição do Roteiro

Um roteiro é composto por quatro elementos: Viagem de Ida (V_{ida}), Hospedagem (H), Programação (P_g) e Viagem de Volta (V_{volta}).

Roteiro = <V_{ida}, $P_g \parallel H$, V_{volta}>

O primeiro elemento do roteiro é V_{ida}. A Hospedagem H deve ocorrer após V_{ida} e antes de V_{volta}. $P_g \parallel H$ significa que a programação P_g ocorre em paralelo ($\parallel$) com a hospedagem H.

A cada elemento X do roteiro é associado um tempo, denotado por t(X) = [i(X), f(X)], onde i(X) é o instante inicial e f(X) é o instante final.

Para os elementos que compõem o roteiro as seguintes regras devem ser satisfeitas:

- $f(V_{ida}) = i(H) \wedge f(H) + 1 = i(V_{volta})$
- $\forall\ e \in P_g, f(V_{ida}) \leq i(e) \wedge f(e) \leq i(V_{volta})$

3.4.2.1 Padrões

Denotamos P_x como o padrão de um elemento x. Assim temos:

$P_R = < P_{Vida} , P_H , P_{Pg} , P_{Vvolta} >$

Um Padrão de Roteiro P_R é composto de um Padrão de Viagem de Ida (P_{Vida}), um Padrão de Hospedagem (P_H), um Padrão de Programação (P_{Pg}) e um Padrão de Viagem de Volta (P_{Volta}).

Um padrão de Programação, por sua vez, é composto de Padrões de Eventos e um Padrão de Alimentação.

$P_{Pg} = < P_{ev}, P_A >$

3.4.2.2 Dados

Os dados disponíveis no sistema são denotados por D_x, e são usados no casamento de padrões.

3.4.3 Viagem

Um roteiro é formado por dois elementos Viagem: Viagem de Ida (V_{ida}) e Viagem de Volta (V_{volta}). V_{ida} e V_{volta} são elementos cujo $t(V_{ida})$ e $t(V_{volta})$ são intervalos de tempo I_i e I_v respectivamente. A Viagem de Ida deve preceder a Viagem de Volta, ou seja $f(V_{ida}) < i(V_{volta})$.

A seguir, definiremos as regras de filtragem, que são regras que um dado (D_x) deve satisfazer para ser sugerido a um usuário, e os critérios de ordenação, que são as características que serão consideradas na ordenação da lista de sugestões.

- Regras de Filtragem

Para uma Viagem de Ida, as seguintes regras devem ser satisfeitas:

- $i(t(P_{Vida})) \leq i(t(D_{Vida})) \wedge f(t(P_{Vida})) \geq f(t(D_{Vida}))$
- $origem(P_{Vida}) = origem(D_{Vida})$
- $destino(P_{Vida}) = destino(D_{Vida})$
- $meio(P_{Vida}) \in meio(D_{Vida})$
- $custo(D_{Vida}) \leq C_{max}(P_{Vida}) + k$

De forma análoga, para uma Viagem de Volta, temos:

- $i(t(P_{Vvolta})) \leq i(t(D_{Vvolta})) \wedge f(t(P_{Vvolta})) \geq f(t(D_{Vvolta}))$
- $origem(P_{Vvolta}) = origem(D_{Vvolta})$
- $destino(P_{Vvolta}) = destino(D_{Vvolta})$
- $meio(P_{Vvolta}) \in meio(D_{Vvolta})$
- $custo(D_{Vvolta}) \leq C_{max}(P_{Vvolta}) + k$

A função origem(X) corresponde à cidade de origem; destino(X) corresponde à cidade de destino; meio(X) corresponde ao meio de transporte (Avião, Trem, Ônibus, Carro Próprio). A função custo(X) determina o custo de um elemento X e deve ser limitada por uma constante C_{max} definida externamente. A constante k possibilita definir

um valor aceitável para que um dado que não satisfaça a um critério de filtragem, mas que se aproxime deste, possa ser sugerido ao usuário.

- Critérios de Ordenação

  Definimos os critérios utilizados para ordenação da Viagem de Ida:

  - $\min(d_c = |\text{classe}(P_{Vida}) - \text{classe}(D_{Vida})|)$
  - $\min(\text{custo}(D_{Vida}))$
  - $\min(d_i = |f(t(P_{Vida})) - i(t(D_{Vida}))|)$

  Para a Viagem de Volta, temos:

  - $\min(d_c = |\text{classe}(P_{Vvolta}) - \text{classe}(D_{Vvolta})|)$
  - $\min(\text{custo}(D_{Vvolta}))$
  - $\min(d_v = |i(t(P_{Vvolta})) - f(t(D_{Vvolta}))|)$

A função classe(X) corresponde à classe da viagem (Primeira classe, Executiva, Econômica). A variável d_c indica a distância entre a classe do padrão e a classe do dado.

Uma Viagem de Ida deve ocorrer dentro de um intervalo especificado em P_{Vida}. Para a Ida, quanto mais próxima a viagem for do fim deste, melhor será a solução. A variável d_i indica a distância entre o final do intervalo de P_{Vida} e o inicio do intervalo de D_{Vida}. Isto significa que quanto menor d_i mais próximo D_{Vida} estará do padrão requerido. De forma análoga, quanto mais próxima a viagem de volta for do inicio do intervalo de P_{Vvolta}, melhor será a solução. E d_v indica a distância entre o inicio do intervalo de P_{Vvolta} e o fim do intervalo de D_{Vvolta}.

3.4.4 Programação

A programação de um roteiro é formada por um conjunto de eventos e um conjunto de alimentações. Os eventos são escalonados de forma que um evento não pode sobrepor o outro.

$P_{Pg} = \{E_k\} \cup \{A_k\}_{k=1,2,...n}$ tal que $\forall$ i, j $\Rightarrow t(E_i) \cap t(E_j) = \varnothing$

3.4.4.1 Eventos

Os eventos da programação podem ser obrigatórios (E_{OB}) ou opcionais (E_{OP}). Eventos obrigatórios são aqueles fixos no roteiro, não podendo ser rearranjados de forma a incluir outro evento. Eventos opcionais podem ser alterados de lugar e até mesmo excluídos. Temos que:

$Pg = E_{OB} \cup E_{OP}$

- Regras de Filtragem

  - $i(t(Pg)) > f(t(D_{Vida})) \wedge f(t(Pg)) < i(t(D_{Vvolta}))$
    $\forall e_i, e_j \in Pg \Rightarrow t(e_i) \cap t(e_j) = \varnothing$

Seja $e \in D_{ev}$, e $P_{ev} \in P_{Pg}$, temos:

- $tipo(e) = tipo(P_{ev})$
- $custo(e) \leq C_{max}(P_{ev}) + k$
- $dist(loc\_ref(P_{ev}), pos(e)) \leq D_{limite}(P_{ev}) + k$

- Critérios de Ordenação

Seja $e \in D_{ev}$, temos:

- $min(custo(e))$
- $min(dist(loc\_ref(P_{ev}), pos(e)))$

A função tipo(X) indica o tipo de um evento que pode ser cinema, teatro, excursão, show de música, etc.

Outra propriedade referente a um evento diz respeito a sua localização. Para cada evento pode-se definir a distância máxima D_{limite} em relação a um local de referência dado pela função $loc\_ref(P_{ev})$. $D_{limite}(P_{ev})$ e $loc\_ref(P_{ev})$ são constantes

definidas em um Padrão de Evento. A função pos(e) define a posição de um evento *e* no espaço. E dist(x, y) retorna a distância entre duas localizações x e y.

3.4.4.2 Alimentação

A Alimentação pode ser caracterizada como um processo por possuir um intervalo de tempo *i*, mas não estar definido antecipadamente por quanto tempo dentro do intervalo *i* irá ocorrer a refeição.

Eventos podem interceptar o intervalo associado à alimentação. No entanto, existe uma duração mínima durMin(a) necessária para se realizar uma alimentação. Se, dado um intervalo I, dur(I) for o tempo de duração de I, temos:
$\forall a \in A \wedge \forall e \in (E_{OB} \cup E_{OP})$ deve valer $dur(t(a)-t(e)) > durMin(a)$

E ocorrer durante a estadia:
$\forall a \in A\ (f(V_{ida}) \leq i(a) \wedge f(a) < i(V_{volta})$.

- Regras de Filtragem

  - $tipoCozinha(P_A) = tipoCozinha(D_A)$
  - $tipoServiço(P_A) \subseteq tipoServiço(D_A)$
  - $dist(loc\_ref(P_A), pos(D_A)) \leq dist\_max(P_A) + k$
  - $custo(D_A) \leq C_{max}(P_A) + k$

A função tipoCozinha(X) indica o tipo de cozinha (Tradicional, Italiana, Chinesa, Japonesa, etc.). A função tipoServiço(X) corresponde aos tipos de serviços disponíveis (A La Carte, Self-Service, Rodízio, etc.).

- Critérios de Ordenação

  - $max(qtd(tipoCozinha(D_A) \cap tipoCozinha(P_A)))$
  - $max(qtd(tipoServiço(D_A) \cap tipoServiço(P_A)))$
  - $max(qtd(facilidades(D_A) \cap facilidades(P_A)))$

- dist(loc_ref(P_A), pos(D_A))
- min(custo(D_A))

A função facilidades(X) indica as facilidades (Aceita Cartão, Estacionamento, Ar Condicionado, etc.).

3.4.5 Hospedagem

O roteiro possui um elemento Hospedagem. A Hospedagem é caracterizada como um processo, pois possui um intervalo de tempo definido, porém este intervalo não precisa estar completamente alocado, podendo os eventos do roteiro ocorrer em paralelo com a Hospedagem.

$(E_{OB} \cup E_{OP}) \parallel H$

Uma Hospedagem deve ocorrer depois da Viagem de Ida e antes da Viagem de Volta:

- $i(t(D_H)) = f(t(D_{Vida}))$, $f(t(D_H)) = i(t(D_{Vvolta})) - 1$

- Regras de Filtragem

  - $i(t(P_H)) = i(t(D_H))$, $f(t(P_H)) = f(t(D_H))$,
  - $tipo(P_H) = tipo(D_H)$,
  - $cidade(P_H) = cidade(D_H)$
  - $dist(loc\_ref(P_H), pos(D_H)) + k$
  - $custo(D_H) \leq C_{max}(P_H) + k$

A função tipo(X) corresponde ao tipo de Hospedagem (Hotel, Pousada, Flat, etc.).

- Critérios de Ordenação

  - $min(d = |categoria(P_H) - categoria(D_H)|)$,

- $\min(dist(loc\_ref(P_H), pos(D_H)) \leq dist\_max(P_H))$
- $\max(qtd(facilidades(D_H) \cap facilidades(D_H)))$
- $\min(custo(D_H))$

A função categoria(X) corresponde a categoria da hospedagem (Cinco estrelas, Quatro estrelas, Três estrelas, Duas estrelas, Uma estrela).

A função facilidades(X) as facilidades de uma hospedagem (Estacionamento, piscina, sauna, quadra de tênis, restaurante, etc.).

Na TABELA 3.1, apresentamos um resumo dos padrões utilizados para compor um roteiro.

Elemento	*Condição básica*	*Melhor solução*
Viagem ida Padrão: P_{Vida} Dado: D_{Vida}	$i(t(P_{Vida})) \leq i(t(D_{Vida})) \wedge$ $f(t(P_{Vida})) \geq f(t(D_{Vida}))$, $origem(P_{Vida}) = origem(D_{Vida})$, $destino(P_{Vida}) = destino(D_{Vida})$, $meio(P_{Vida}) \in meio(D_{Vida})$, $custo(D_{Vida}) \leq C_{max}(P_{Vida}) + k$	$\min(\lvert f(t(P_{Vida})) - i(t(D_{Vida}))\rvert)$, $classe(P_{Vida}) = classe(D_{Vida})$, $\min(custo(D_{Vida}))$
Viagem volta Padrão: P_{Vvolta} Dado: D_{Vvolta}	$i(t(P_{Vvolta})) \leq i(t(D_{Vvolta})) \wedge$ $f(t(P_{Vvolta})) \geq f(t(D_{Vvolta}))$, $origem(P_{Vvolta}) = origem(D_{Vvolta})$, $destino(P_{Vvolta}) = destino(D_{Vvolta})$, $meio(P_{Vvolta}) \in meio(D_{Vvolta})$, $custo(D_{Vvolta}) \leq C_{max}(P_{Vvolta}) + k$	$\min(\lvert i(t(P_{Vvolta})) - f(t(D_{Vvolta}))\rvert)$, $classe(P_{Vvolta}) = classe(D_{Vvolta})$, $\min(custo(D_{Vvolta}))$
Hospedagem Padrão: PH Dado: DH	$i(t(P_H)) = i(t(D_H))$, $f(t(P_H)) = f(t(D_H))$, $i(t(D_H)) = f(t(D_{Vida}))$, $f(t(D_H)) = i(t(D_{Vvolta})) - 1$, $tipo(P_H) = tipo(D_H)$, $cidade(P_H) = cidade(D_H)$, $custo(D_H) \leq C_{max}(P_H) + k$, $dist(loc\_ref(P_H), pos(D_H)) \leq$ $dist\_max(P_H) + k$	$categoria(P_H) = categoria(D_H)$, $\min(custo(D_H))$, $\min(dist(loc\_ref(P_H), pos(D_H)))$
Facilidades da Hospedagem Padrão: P_H Dado: D_H	$facilidades(P_H) \subseteq facilidades(D_H)$	$\max(qtd(facilidades(D_H)))$
Programação	$tipo(P_{Pg}) = tipo(D_{Pg})$, $i(t(P_g)) > f(t(D_{Vida})) \wedge$ $f(t(P_g)) < i(t(D_{Vvolta}))$,	$\min(dist(loc\_ref(P_{ev}), pos(e)))$, $\min(custo(e))$

Padrão: P_{Pg} Dado: D_{Pg}	$\forall e_i, e_j \in P_g \Rightarrow t(e_i) \cap t(e_j) = \varnothing$ Seja $e \in D_{ev}$, temos: $tipo(e) = tipo(P_{ev})$ $custo(e) \leq C_{max}(P_{ev}) + k$ $dist(loc\_ref(P_{ev}), pos(e)) \leq D_{limite}(P_{ev}) + k$	
<u>Alimentação</u> Padrão: P_A Dado: D_A	$tipoCozinha(P_A) = tipoCozinha(D_A)$, $tipoServiço(P_A) \subseteq tipoServiço(D_A)$, $facilidades(P_A) \subseteq facilidades(D_A)$ $custo(D_A) \leq C_{max}(P_A) + k$, $dist(loc\_ref(P_A), pos(D_A)) \leq dist\_max(P_A) + k$	$min(dist(loc\_ref(P_A), pos(D_A)))$, $max(qtd(facilidades(D_H)))$, $min(custo(D_A))$

TABELA 3.1: Definição dos Padrões

3.5 Compilação do Roteiro

A compilação do roteiro é executada confrontando-se os padrões com os dados recuperados pelos mediadores. Um roteiro é composto de viagem de ida, viagem de volta, hospedagem e programação no local.

A viagem é incluída no roteiro selecionando-a de uma lista ordenada de viagens fornecida pelo mediador de Viagem. Da mesma forma, uma hospedagem é incluída selecionando-a de uma lista de hospedagens fornecida pelo Mediador de Hospedagens. No entanto, a programação exige um esforço maior para sua criação. O Compilador de roteiro recebe os eventos do Mediador de Eventos e cria uma programação que será incluída no roteiro. A programação é criada utilizando-se um algoritmo de escalonamento de eventos, descrito na seção 3.5.2.

Os Mediadores utilizam os padrões para ordenar suas coleções de itens do roteiro. A ordenação é feita de acordo com os itens que mais se assemelham ao padrão criado.

3.5.1 Ordenando os itens

Um mediador ordena seus elementos de acordo com a semelhança de cada elemento com o padrão solicitado.

A seguir mostraremos a função que mede a similaridade entre um padrão e um elemento, utilizada pelo SEI-Tur.

3.1.1.1. Função de distância

A distância euclidiana é amplamente utilizada e bastante apropriada para atributos quantitativos, mas, freqüentemente, essa função de distância não manipula atributos qualitativos, também chamados de atributos simbólicos.

Uma forma de lidar com conjuntos de dados com atributos qualitativos e quantitativos é usar uma função de distância heterogênea que utiliza funções de distância diferentes para diferentes tipos de atributos. Uma abordagem bastante difundida é usar a métrica *overlap* para atributos qualitativos e a distância euclidiana para atributos quantitativos. Essa abordagem é conhecida como *Heterogeneous Euclidean Overlap Metric (HEOM)* [Wilson *et al*, 1997], e define a distância entre dois exemplos x e y conforme a equação 3.1.

$$heom(x,y) = \sqrt{\sum_{i=1}^{n} d_i(x_i, y_i)^2} \qquad \text{(Equação 3.1)}$$

Onde:

$$d_i(x_i,y_i) = \begin{cases} 1 & \text{se } x_i \text{ ou } y_i \text{ forem desconhecidos} \\ overlap(x_i,y_i) & \text{se o atributo } A_i \text{ for qualitativo} \\ \dfrac{|x_i - y_i|}{max(A_i) - min(A_i)} & \text{se o atributo } A_i \text{ for quantitativo} \end{cases}$$

onde $overlap(x_i,y_i) = 1$ se $x_i \neq y_i$ e igual a 0, caso contrário.

Em [Ricci *et at*, 2003], é utilizada uma versão modificada da equação 3.1. A equação 3.2 atribui pesos $0 \leq w_i \leq 1$, i=1,...,n para cada atributo na consulta.

$$heom(x,y) = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i}} \sqrt{\sum_{i=1}^{n} w_i d_i(x_i, y_i)^2} \qquad \text{(Equação 3.2)}$$

Além das condições mostradas na equação 3.1, inclui a seguinte condição, para abranger aqueles atributos que possuem um conjunto de símbolos em seu valor:

$d_i(x_i,y_i) = 1- \mid x_i \cap y_i \mid / \mid x_i \cup y_i \mid$ se o atributo for um conjunto de símbolos

No *SEI-Tur*, utilizamos uma adaptação da equação 3.2. A equação original não atende completamente às necessidades. Por exemplo, se um usuário procura um hotel com sauna e piscina e temos cadastrados dois hotéis: hotel A, possuindo piscina, sauna, restaurante; e hotel B, com: Piscina, sauna, restaurante, quadra de tênis, ar condicionado. Utilizando a equação acima, o hotel B ficaria em desvantagem em relação ao hotel A, pois o mesmo possui um maior número de facilidades.

Então utilizamos a seguinte adaptação:

$d_i(x_i, P_i) = 1- \mid x_i \cap P_i \mid / \mid P_i \mid$ se o atributo for um conjunto de símbolos

Onde P é o padrão.

3.1.1.2. Um exemplo

Mostraremos agora um exemplo de como é calculado o fator que indica a semelhança entre um item de hospedagem e um padrão de hospedagem.

Imagine que um usuário deseja hospedar-se em um hotel cinco estrelas, com piscina e sauna, e que seja próximo ao shopping, no máximo 2 Km.

O intervalo refere-se ao tempo de utilização da hospedagem e é um atributo fornecido indiretamente pelo usuário. O usuário fornece apenas o intervalo completo do roteiro e o intervalo de hospedagem depende dos intervalos das viagens de ida e volta.

O seguinte padrão de hospedagem é criado:

Tipo de Hospedagem	Hotel
Categoria	Cinco estrelas
Facilidades	Piscina, sauna
Custo Máximo	Não especificado
Distância Máxima	2 km

Local de Referência	Shopping Iguatemi (CEP: 58104-901)
Intervalo	[20/03/2004 12:00, 25/03/2004 12:00]

TABELA 3.2: Padrão de Hospedagem

Quando o usuário solicita a criação de um roteiro, uma das tarefas do *SEI-Tur* é a seleção de uma hospedagem que esteja próxima às necessidades do usuário. O *SEI-Tur* então deve acionar o *MediadorDeHospedagem*, solicitando a recuperação das hospedagens que possuem quartos disponíveis no intervalo definido no Padrão de Hospedagem. Considerando que as seguintes hospedagens foram recuperadas:

	Nome	**Tipo**	**Categoria**	**Facilidades**	**Endereço**	**Preço**
		A_1	a_2	a_3	a_4	a_5
H_1	Hotel Serrano	Hotel	Cinco estrelas	Piscina, restaurante, estacionamento, ar condicionado.	CEP: 58100-160 Distância=1,5 Km (*)	R$ 50,00
H_2	Hotel Village	Hotel	Cinco estrelas	Piscina, restaurante, sauna, quadra de tênis, ar condicionado.	CEP: 58104-575 Distância=0,8 Km (*)	R$ 80,00
H_3	Pousada Aconchego	Pousada	Duas estrelas	Ar condicionado	CEP: 58101-030 Distância=1,2 Km (*)	R$ 30,00

TABELA 3.3: Hospedagens

(*) A distância é calculada com base no CEP. Os valores atribuídos na TABELA 3.3 são fictícios.

No protótipo implementado, estamos supondo que todos os atributos possuem o mesmo peso.

De acordo com a EQUAÇÃO 3.2, temos:

$$HEOM(H1, P)^2 = (0^2 + 0^2 + 0{,}5^2 + 0{,}75^2 + 1^2)/5 \Rightarrow HEOM(H1, P) = 0{,}60$$

$$HEOM(H2, P)^2 = (0^2 + 0^2 + 0^2 + 0{,}4^2 + 1^2)/5 \Rightarrow HEOM(H2, P) = 0{,}48$$

$$HEOM(H3, P)^2 = (1^2 + 1^2 + 1^2 + 0{,}6^2 + 1^2)/5 \Rightarrow HEOM(H3, P) = 0{,}93$$

Os resultados obtidos mostram que H2 possui a menor distância do padrão P, em relação às outras hospedagens.

3.5.2 Criando uma Programação

Uma programação é uma coleção de eventos escalonados dentro de um intervalo de tempo de forma que um evento não sobreponha o outro. Apresentaremos a seguir a lógica utilizada pelo *SEI-Tur* para criar uma programação composta de eventos que casam com as preferências do usuário.

3.1.1.3. Algoritmo de escalonamento de eventos

O algoritmo escalonador de eventos procura maximizar o número de eventos inseridos no roteiro. Quando um evento não pode ser inserido, devido a um choque de horário com outros eventos já existentes no roteiro, o algoritmo tenta fazer um deslocamento de eventos. Ele verifica se o mesmo evento do roteiro pode ocorrer em outro intervalo. Em caso positivo, verifica se este intervalo pode ser alocado. Caso o novo intervalo também esteja ocupado por outro evento, o algoritmo vai agora tentar realocar este próximo evento. Este processo é feito de forma recursiva e será detalhado a seguir.

```
1  eventos ← lista ordenada de eventos que casam com os padrões definidos pelo usuário.

2  criaProgramação(eventos)
3  {  Para cada evento ev ∈ eventos
4          {  Se ev ⊄ eventosDoRoteiro
5              {  intervalos ← obtém todos os intervalos que o evento ev pode ocorrer, dentro do
                  intervalo do roteiro.
6                 Para cada i ∈ intervalos
7                 {  Se i não dá choque com eventosDoRoteiro
8                     {  fixa i para ev;
9                        adiciona ev → eventosDoRoteiro
10                       retorne "Evento inserido com sucesso".
11                       FIM.
                      }
                  }

                  /* Neste ponto, todos os intervalos que o evento pode ocorrer já estão ocupados por
                     outros eventos dentro do roteiro. Tentaremos então escalonar os eventos
                     anteriormente inseridos.
                  */
12                Para cada i ∈ intervalos
13                {  Se desloca(i) = verdadeiro;
14                    {  fixa i para ev;
15                       adiciona ev → eventosDoRoteiro
16                       retorne "Evento inserido com sucesso".
17                       FIM
                      }
                  }
              }
18            senão
19                retorne "O evento já está contido no roteiro."
```

```
            }
    }

20  desloca(i)
21  { Para cada evRoteiro ∈ eventosDoRoteiro
22              {  /* Tenta mover todos evRoteiro que intercepta i. */
23                  Se evRoteiro intercepta i
24                  {  Se move(evRoteiro) = falso;
25                          retorne falso;
                    }
                }
26              retorne verdadeiro;
    }

27  move(evento)
28  { intervalos ← obtém outros intervalos possíveis para evento.
29          Para cada i ∈ intervalos
30                  Se i não dá choque com algum ev ∈ eventosDoRoteiro
31                          fixa i para evento;
32                          retorne verdadeiro;
33                  senão
34                          ev ← evento que dá choque com i.
35                          retorne move(ev);
36          retorne falso;
    }
```

FIGURA 3.12: Algoritmo de escalonamento de eventos.

Na linha 1, é criada uma lista contendo os eventos que serão inseridos no roteiro. Da linha 2 à linha 19, é mostrada a função CriaProgramação(*eventos*), passando como parâmetro a lista de eventos. A linha 3 percorre todos os eventos contidos na lista, e verifica (linha 4) se cada um destes eventos já estão contidos no roteiro. Da linha 5 à linha 11 o algoritmo analisa todos os intervalos em que um evento pode ocorrer e para cada intervalo destes tenta inseri-lo no roteiro. Caso nenhum dos intervalos esteja disponível no roteiro, o algoritmo tenta deslocar os eventos anteriormente inseridos (linhas 12-17). Na linha 13 é chamada a função desloca(*i*). Esta função recebe um intervalo como parâmetro, e para cada evento no roteiro que intercepta o intervalo, ela tenta mover o evento para outro intervalo possível dentro do roteiro (linhas 20-25). Na linha 23, para cada evento que intercepta o intervalo é chamada a função move(*evento*).

Em move(*evento*), (linhas 26-35) o algoritmo primeiramente obtém uma lista de todos os intervalos possíveis para o evento que se deseja mover, exceto o intervalo em que este já esta fixado no roteiro. O algoritmo, então, percorre todos estes intervalos e para cada intervalo verifica se este dá choque com outros eventos existentes no roteiro. Em caso negativo, o evento é então movido para este intervalo e a função retorna

verdadeiro. No entanto, se o intervalo der choque com um evento, este evento é recuperado e a função move(*evento*) é chamada recursivamente (linha 34).

3.1.1.4. Exemplo de funcionamento do algoritmo

Seja I = [a, h] o intervalo de um roteiro. E dados os intervalos possíveis para os eventos:

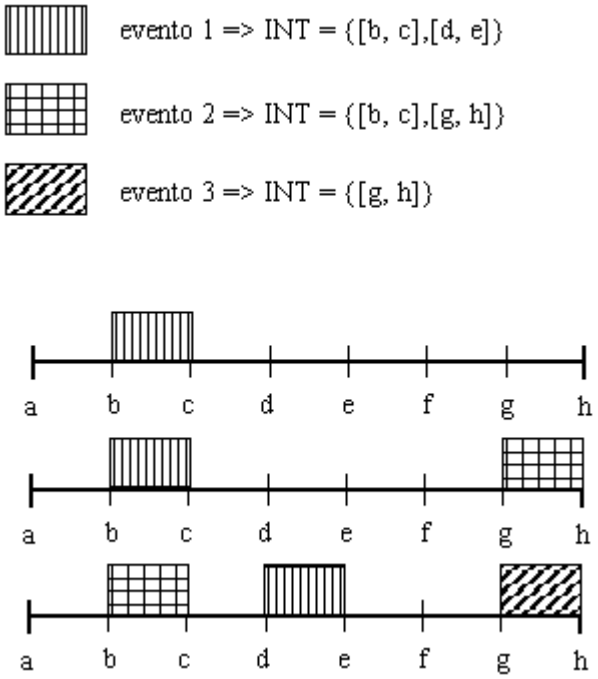

FIGURA 3.13: Exemplo de funcionamento do algoritmo.

O evento 1 é inserido no roteiro sem choques com outros eventos. No entanto, o evento 2 será inserido no segundo intervalo, pois no primeiro, [b, c], ocorre um choque com o evento 1. Já o evento 3, apenas pode ser inserido no intervalo [g, h], que está sendo ocupado pelo evento 2. O escalonador tenta então movê-lo para outro intervalo, [b, c]. O intervalo [b, c], por sua vez, está sendo ocupado pelo evento 1. O evento 1 é então movido para o intervalo [d, e].

3.6 Considerações finais

Neste capítulo descrevemos os aspectos relacionados ao funcionamento do *SEI-Tur*. Mostramos as camadas que compõem a arquitetura do *SEI-Tur*, detalhando seus componentes; explicamos como o *SEI-Tur* interage com outros sistemas disponíveis na *Web* para coletar informações; detalhamos os padrões utilizados para definir as preferências do usuário; e por fim explicamos as atividades necessárias à tarefa de composição do roteiro turístico.

Como será exemplificado no capítulo 5, a programação de eventos não se restringe a eventos turísticos, como cinemas, teatros, shows e outras atrações oferecidas pela cidade, mas também, à participação em congressos, festivais ou simpósios, que oferecem uma programação de palestras e cursos, dos quais o participante escolhe o que lhe convier.

4. Desenvolvimento do SEI-Tur

Neste capítulo, apresentamos a modelagem para a construção do *SEI-Tur* (Serviço de Informações Turísticas). O objetivo do projeto é criar um sistema *Web* que monte um roteiro turístico, incluindo itens de viagem, hospedagem e programação no local, de acordo com as preferências do usuário.

Utilizamos o processo de [Larman, 1998], que propõe uma abordagem orientada a objetos.

4.1 Modelo Conceitual

Nesta seção apresentamos o modelo conceitual do *SEI-Tur*. Neste modelo (FIGURA 4.1) realizamos uma visão de alto nível do sistema.

No *SEI-Tur*, temos um processo de criação de padrões que é feito por um usuário do sistema. Um Padrão de Roteiro é composto por dois Padrões de Viagem, um Padrão de Hospedagem e um Padrão de Programação. O Padrão de Programação, por sua vez, é composto por um Padrão de Alimentação e vários Padrões de Evento. Um Padrão de Viagem pode ser um Padrão de Viagem de Ida ou um Padrão de Viagem de Volta.

Após o usuário criar o Padrão de Roteiro, ele pode ativar o Compilador de Roteiro para que este crie o Roteiro. O Compilador de Roteiro usa o Padrão criado pelo usuário e ativa os mediadores para que estes acessem as fontes de dados disponíveis no *SEI-Tur*.

O Roteiro a ser criado é composto de duas Viagens (Ida e Volta), uma Hospedagem e uma Programação. A Programação é composta de vários eventos e várias opções de alimentação.

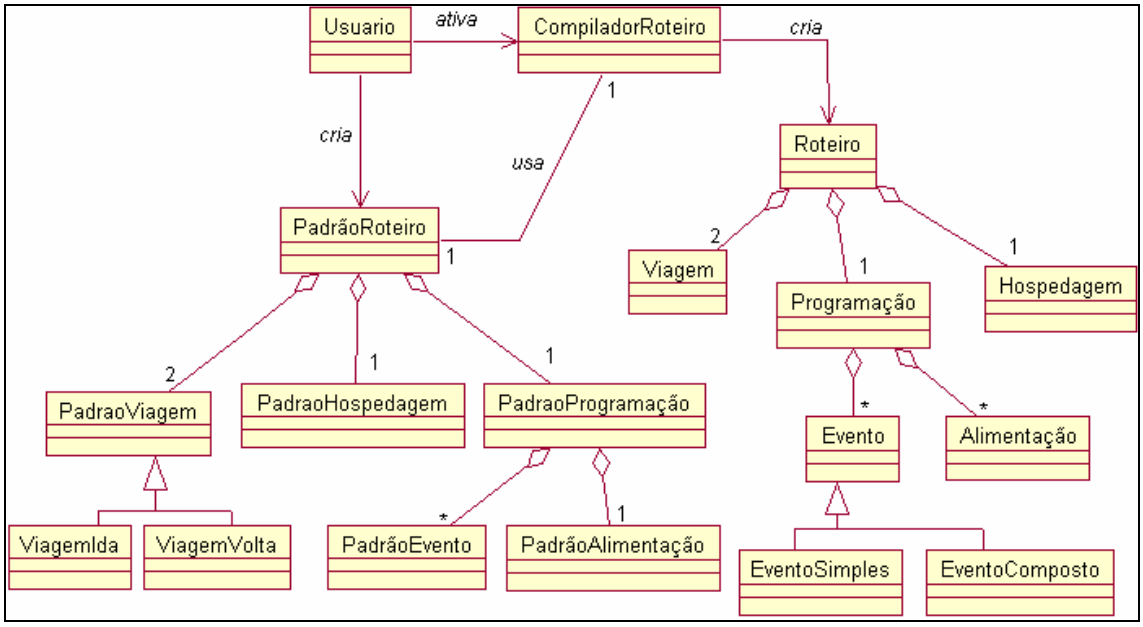

FIGURA 4.1: Modelo Conceitual.

4.2 Levantamento de Requisitos

Nesta seção, detalhamos os requisitos funcionais e não-funcionais do *SEI-Tur*.

4.2.1 Requisitos Funcionais

Estes requisitos dizem respeito ao funcionamento do sistema, determinando "o quê" o sistema faz.

RF1- Auxiliar o usuário a compor um roteiro turístico de acordo com suas preferências

O sistema deve auxiliar um usuário a criar um roteiro turístico de acordo com as suas preferências. Para cada elemento do roteiro, o sistema deve sugerir uma lista de opções ordenada de acordo com as preferências do usuário. Este deve incluir itens no roteiro a fim de compor o roteiro completo.

RF2- Propor uma programação completa ao usuário

O sistema, além de sugerir itens isolados, deve propor uma programação formada de eventos que casam com as preferências do usuário. Os eventos disponíveis no sistema devem ser escalonados dentro do intervalo de estadia do turista na cidade de destino, de forma a maximizar o número de eventos no roteiro.

RF3- Disponibilizar o sistema na *Web*

O *SEI-Tur* deve estar disponível na *Web* permitindo que os usuários, através de um simples *browser*, possam acessar o sistema e realizar operações a qualquer momento.

RF4- Garantir a consistência temporal dos elementos do roteiro

Um dos requisitos mais importantes do sistema é garantir que os itens que compõem o roteiro sejam cronologicamente consistentes entre si. Ou seja, o sistema deve garantir que eventos ocorram dentro de um intervalo de tempo previamente definido e que um evento não sobreponha outro.

RF5- Utilizar serviços de outros sistemas

Uma das características do *SEI-Tur* é utilizar dados disponíveis na *Web* para compor roteiros turísticos. Ao invés de manter um cópia local dos dados, o sistema irá interagir com outros sistemas para obter dados atualizados a todo momento.

RF6- Ordenar os elementos do roteiro.

O sistema deve manter uma lista ordenada dos elementos de acordo com as preferências do usuário definidas pelo padrão de roteiro. A lista será utilizada para compor o roteiro e para possíveis alterações solicitadas pelo usuário.

RF7- Permitir alteração dos elementos do roteiro

Caso o usuário deseje alterar as opções selecionadas, ele poderá consultar outras opções disponíveis que também casem com suas preferências. O sistema deve então disponibilizar outras opções para o usuário, exibindo uma lista ordenada de elementos que podem ser incluídos no roteiro sem perda de consistência.

RF8- Calcular o custo total do roteiro

Um fator bastante relevante na hora de selecionar um roteiro é verificar se o custo total está dentro das possibilidades do usuário. Então, para facilitar a visibilidade do usuário, o sistema deverá apresentar a soma dos custos de todos os itens que compõem o roteiro.

RF9- Considerar eventos programados

O *SEI-Tur* deve aceitar a programação de um evento programado (Congresso, Simpósio, etc.). Ou seja, o sistema deve permitir que um usuário responsável pela manutenção do sistema insira eventos que possuam uma programação pré-definida (eventos programados). Uma programação pré-definida é composta de eventos com intervalos de tempo fixos, em que o usuário poderá selecionar os eventos dos quais irá participar. Na hora de criar uma programação completa para o roteiro, o sistema deverá considerar tanto os eventos programados, como os eventos livres (cinema, teatro, excursões, etc).

4.2.2 Requisitos Não-Funcionais

Através do auxilio dos requisitos não funcionais os projetistas procuram determinar "como" o sistema executa determinada ação. Para isso, esses requisitos caracterizam o sistema dentro de categorias, tais como: facilidade de uso, tempo de resposta, perfil dos usuários, etc. Alguns requisitos não funcionais do *SEI-Tur* são:

RNF1- Interface adequada aos tipos de usuários

A interface deverá ser de fácil utilização. O *SEI-Tur* abrange vários tipos de usuários, incluindo desde usuários mais experientes a usuários leigos. Deste modo, um tipo de interface amigável deve ser definido.

RNF2- Manter os cadastros das fontes de dados atualizados

O sistema deverá manter um cadastro das fontes de informações a serem utilizadas na construção de roteiros. Estas fontes podem ser incluídas, alteradas ou excluídas do sistema.

RNF3- Independência de plataforma

Com a difusão de diferentes plataformas e o crescimento do número de usuários, se faz necessário o desenvolvimento de aplicações que sejam portáveis. As aplicações que são executadas via *browser* favorecem os usuários que utilizam diferentes plataformas.

4.3 Modelo de Casos de Uso

Um caso de uso é um documento narrativo que descreve a seqüência de eventos de um ator (um agente externo) que usa um sistema para completar um processo [Jacobson, 1992].

A técnica de casos de uso ajuda no levantamento dos requisitos funcionais, e pode ser usada para identificar os conceitos que fazem parte do sistema. Para cada suposta funcionalidade do sistema, criamos um caso de uso que irá explorar como esta funcionalidade será executada pelo sistema. Um caso de uso pode também apenas apresentar uma funcionalidade de forma genérica. Neste caso o caso de uso é denominado de "alto nível". Para detalhar mais cada funcionalidade e obter uma compreensão mais profunda, utilizaremos outro tipo de caso de uso, o caso de uso "expandido".

Na análise e no projeto do *SEI-Tur* avaliamos dois tipos de usuários para o sistema:

- usuários que desejam criar um roteiro turístico completo, de acordo com as suas preferências;
- usuários administradores que desejam fazer manutenção nas fontes de dados existentes no sistema.

A FIGURA 4.2 apresenta o diagrama de casos de uso do nosso sistema.

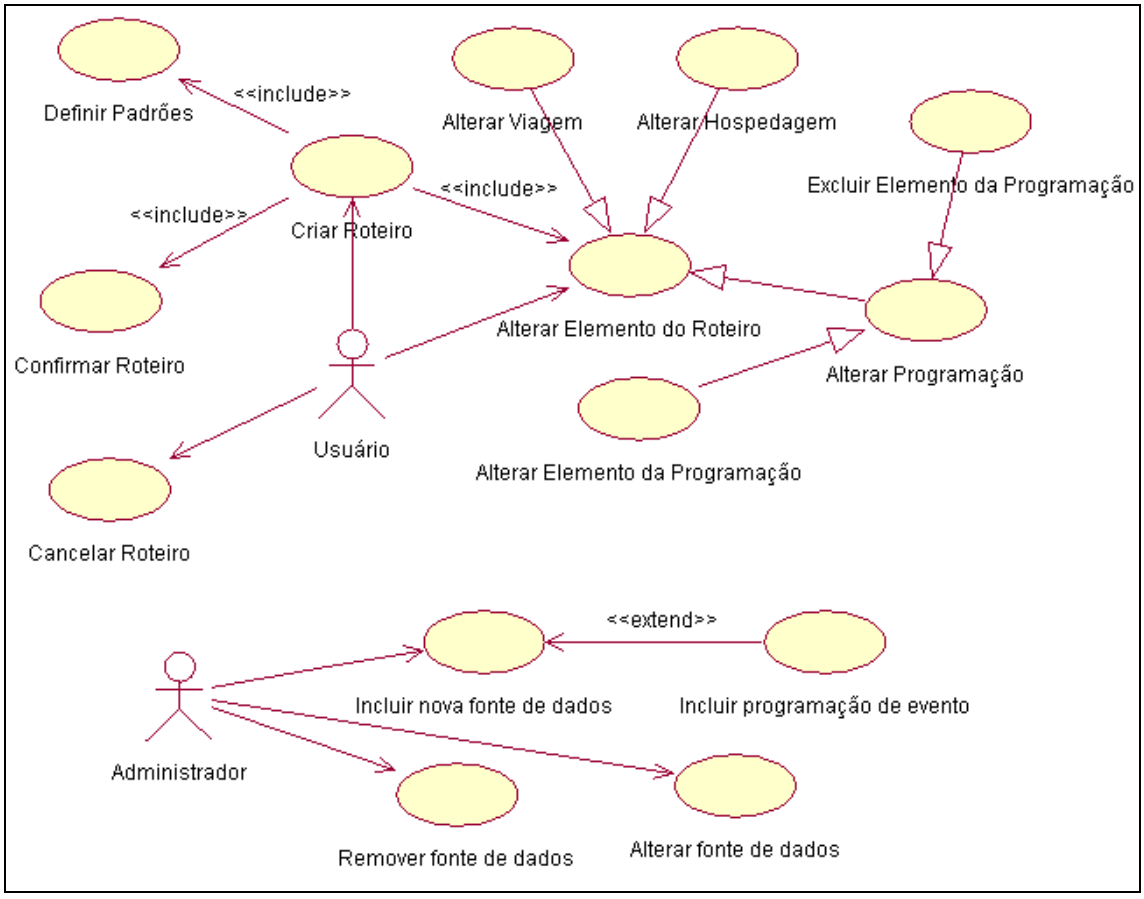

FIGURA 4.2: Diagrama de Casos de Uso

A seguir estão as descrições dos principais casos de uso *Definir Padrões*, *Criar Roteiro*, *Confirmar Roteiro* e *Cancelar Roteiro*.

Caso de Uso: Definir Padrões **Atores:** Um usuário do sistema **Descrição:** Um usuário aciona o sistema informando preferências para a definição de padrões.

Seqüência típica de eventos:

Ação do ator	Resposta do sistema
1. O usuário aciona o sistema para definir seus de padrões de roteiro.	2. O sistema apresenta um formulário solicitando as preferências do usuário em relação à viagem.
3. O usuário preenche o formulário e submete os dados ao sistema.	4. O sistema cria um padrão de viagem de ida e um padrão de viagem de volta com os dados submetidos.
	5. O sistema apresenta um formulário solicitando as preferências do usuário em relação à hospedagem.
6. O usuário preenche o formulário e submete os dados ao sistema.	7. O sistema cria um padrão de hospedagem com os dados submetidos.
	8. O sistema apresenta um formulário solicitando as preferências do usuário em relação a eventos.
9. O usuário preenche o formulário e submete os dados ao sistema.	10. O sistema cria padrões de eventos com os dados submetidos.
	11. O sistema apresenta um formulário solicitando as preferências do usuário em relação à alimentação.
12. O usuário preenche o formulário e submete os dados ao sistema.	13. O sistema cria um padrão de alimentação com os dados submetidos.
	14. O sistema cria um padrão de programação, que será composto dos padrões de eventos e do padrão de alimentação criados nos passos 10 e 13 respectivamente.
	15. O sistema cria o padrão de roteiro, que será composto dos padrões de viagem de ida e volta, do padrão de hospedagem e do padrão de programação criados nos passos 4, 7 e 14 respectivamente.

TABELA 4.1: Caso de Uso Definir Padrões.

Caso de Uso: Criar Roteiro
Atores: Um usuário do sistema
Descrição: Um usuário aciona o sistema para a criação de um roteiro de viagens.

Seqüência típica de eventos:

Ação do ator	Resposta do sistema
1. O usuário aciona o sistema para criar um roteiro para sua viagem.	2. O sistema solicita, através de um formulário, os dados da viagem (origem,

	destino, data de partida, data de chegada, motivo da viagem).
3. O usuário preenche os dados e submete ao sistema.	
	4. O sistema solicita que o usuário defina seus padrões.
5. **Iniciar** *Definir Padrões*.	
	6. O sistema consulta as opções de viagem disponíveis no sistema e casa com os padrões de viagem definidos pelo usuário.
	7. O sistema mostra as opções de viagem ao usuário.
8. O usuário seleciona uma viagem de ida e uma viagem de volta no roteiro.	9. O sistema consulta as opções de hospedagens disponíveis no sistema
10. O usuário seleciona uma hospedagem no roteiro.	11. O sistema verifica o motivo da viagem: a. Se motivo da viagem for participação em evento, ver seção *Selecionar Eventos Programados*.
	12a. O sistema consulta os eventos disponíveis no sistema e casa com os padrões de evento.
	12b. O sistema consulta as opções de alimentação disponíveis no sistema e casa com o padrão de alimentação.
	13. O sistema cria uma programação no local com os eventos e opções de alimentação disponíveis, e apresenta programação ao usuário.
14. O usuário pode alterar ou excluir os itens da programação.	
	15. O sistema cria o roteiro incluindo a programação, a viagem de ida, a viagem de volta, e a hospedagem selecionadas pelo usuário, e apresenta o roteiro completo ao usuário.
16. Se usuário deseja alterar elemento do roteiro, **iniciar** *Alterar Elemento do Roteiro*.	
17. Se usuário deseja confirmar roteiro, **iniciar** *Confirmar Roteiro*.	

TABELA 4.2: Caso de Uso Criar Roteiro.

Seção: Selecionar Eventos Programados

Ação do ator	Resposta do sistema
	1. O sistema solicita que o usuário selecione os eventos que irá participar

	dentro da programação de um evento programado.
2. O usuário informa os eventos que irá participar.	3. O sistema guarda os eventos selecionados em uma lista de eventos obrigatórios que serão utilizados na composição da programação.

TABELA 4.3: Seção Selecionar Eventos Programados.

Caso de Uso: Confirmar Roteiro
Atores: Um usuário do sistema
Descrição: Um usuário aciona o sistema informando que aceita o roteiro criado.

Seqüência típica de eventos:

Ação do ator	Resposta do sistema
1. O usuário aciona o sistema informando que aceita o roteiro criado.	2. O solicita a identificação do usuário.
3. Usuário informa se já é cadastrado no sistema. a. Se usuário possui cadastro, ver seção *Confirmar Cadastro*. b. Se usuário não possui cadastro, ver seção *Cadastrar Usuário*.	4. O sistema efetua as reservas necessárias.

TABELA 4.4: Caso de Uso Confirmar Roteiro.

Seção: Confirmar Cadastro

Ação do ator	Resposta do sistema
1. O usuário informa que já possui cadastro no sistema e fornece identificador.	2. O sistema recupera os dados do usuário e solicita confirmação.
3. O usuário verifica se seus dados estão atualizados.	4. O sistema atualiza os dados do usuário.

TABELA 4.5: Seção Confirmar Cadastro.

Seção: Cadastrar Usuário

Ação do ator	Resposta do sistema
1. O usuário informa que ainda não possui cadastro no sistema.	2. O sistema solicita, através de um formulário, os dados do usuário.
3. O usuário preenche os dados e submete ao sistema.	4. O sistema cadastra o usuário com os dados enviados.

TABELA 4.6: Seção Cadastra Usuário.

Caso de Uso: Cancelar Roteiro **Atores:** Um usuário do sistema **Descrição:** Um usuário aciona o sistema informando que deseja cancelar o roteiro criado.

Seqüência típica de eventos:

Ação do ator	Resposta do sistema
1. O usuário aciona o sistema informando que deseja cancelar o roteiro criado.	2. O sistema pede para o usuário identificar-se.
3. O usuário identifica-se.	4. O sistema cancela as reservas efetuadas.

TABELA 4.7: Caso de Uso Cancelar Roteiro.

4.4 Diagrama de Classes

Um diagrama de classes ilustra as especificações para as classes desenvolvidas para o *SEI-Tur*. Mostramos, nas FIGURAS 4.3 e 4.4, o diagrama desenvolvido.

Para simplificar o entendimento, dividimos o diagrama em duas partes.

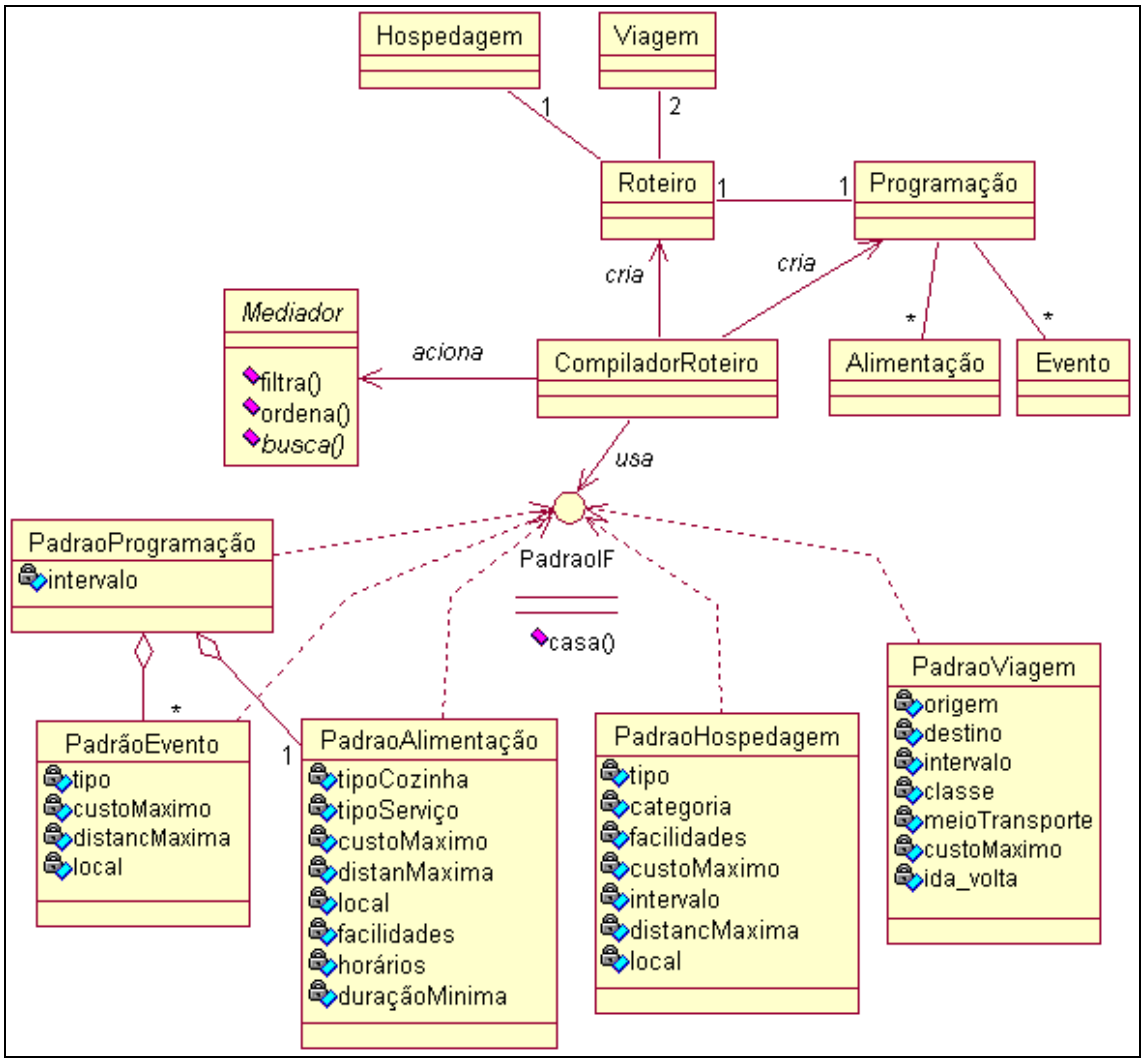

FIGURA 4.3: Diagrama de Classes

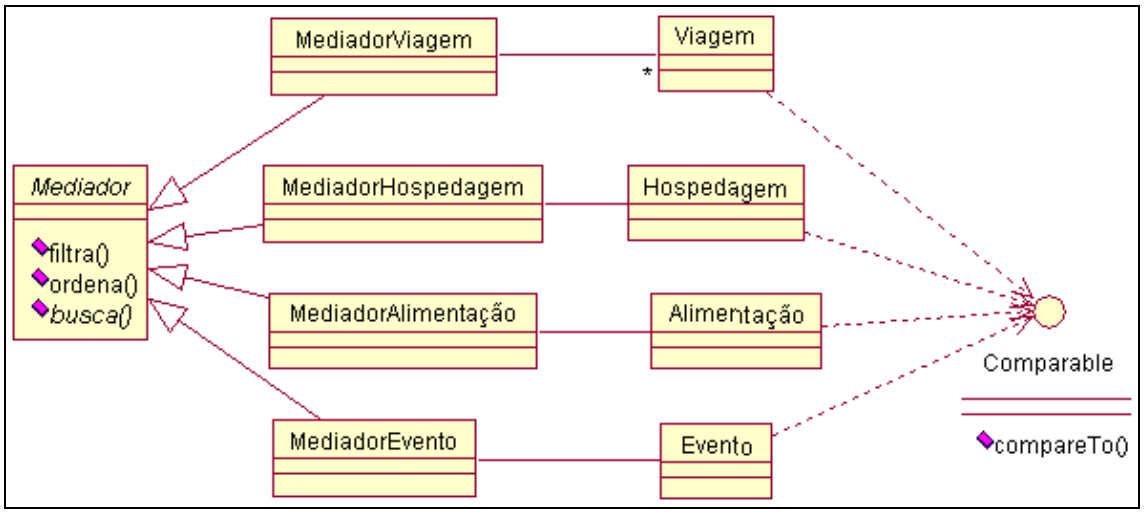

FIGURA 4.4: Diagrama de Classes (continuação)

A classe CompiladorRoteiro é responsável por criar o objeto Roteiro. Para isto ela utiliza os padrões que foram criados pelo usuário. O CompiladorRoteiro aciona o Mediador passando um PadrãoIF. O Mediador acessa as fontes de informação, recupera os dados, confronta-os com o padrão e retorna uma lista ordenada de itens ao CompiladorRoteiro.

As classes que irão compor um roteiro (viagem, hospedagem, alimentação e evento) devem implementar a interface *Comparable* do pacote *Java.lang*, que possui o método *compareTo()*. Implementando esta interface a classe indica que possui uma ordem natural, sendo possível ordenar e comparar objetos.

Todos os padrões devem implementar o método *casa()* que recebe um *Comparable* como parâmetro e retorna um *float* que corresponde ao grau de similaridade entre um padrão e um objeto *Comparable*.

O Compilador Roteiro cria um objeto Programação com os itens retornados pelos Mediadores de Evento e Alimentação e inclui a Programação no Roteiro, junto com os outros itens (Hospedagem e Viagens).

4.5 Diagrama de Seqüência

O diagrama de seqüência enfatiza o comportamento dos objetos em um sistema, incluindo suas operações, interações, colaborações e histórias de estado em seqüência temporal de mensagem e representação explícita de ativação de operações [Furlan, 1998].

Na FIGURA 4.5, mostramos o diagrama de seqüência para o processo de criação de um roteiro.

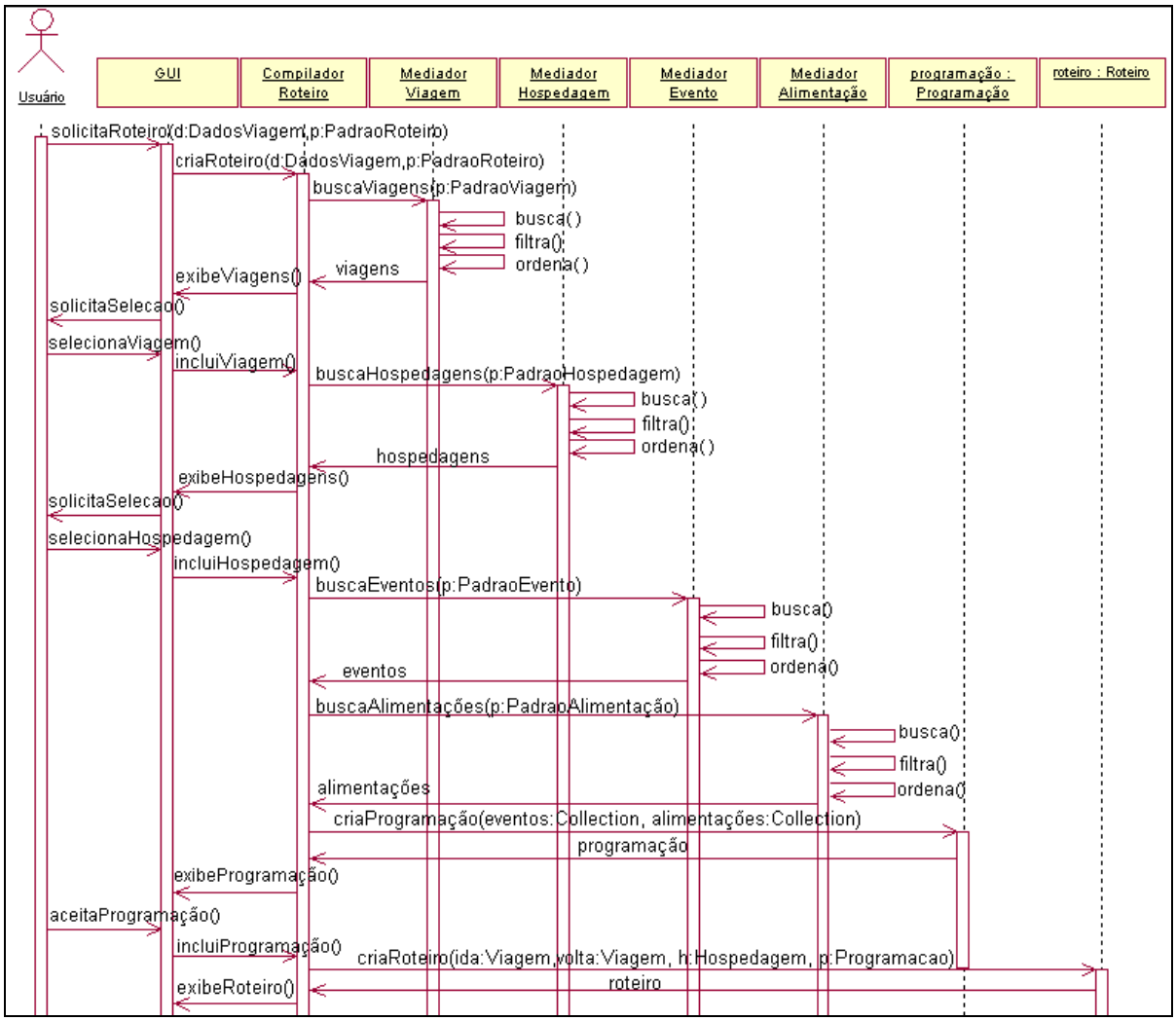

FIGURA 4.5: Diagrama de Seqüência do processo de criação de um roteiro.

4.6 Diagramas de Atividade e Estado

Diagramas de atividades são usados para modelagem do fluxo de controle entre atividades de um sistema. Através deste diagrama torna-se possível modelar o fluxo normal e os desvios causados pela manifestação de situações excepcionais em casos de uso ou funções do sistema.

No diagrama de atividades da FIGURA 4.5, podemos as atividades executadas no processo de criação de um roteiro. Alguma atividades são independentes e podem ocorrer em paralelo.

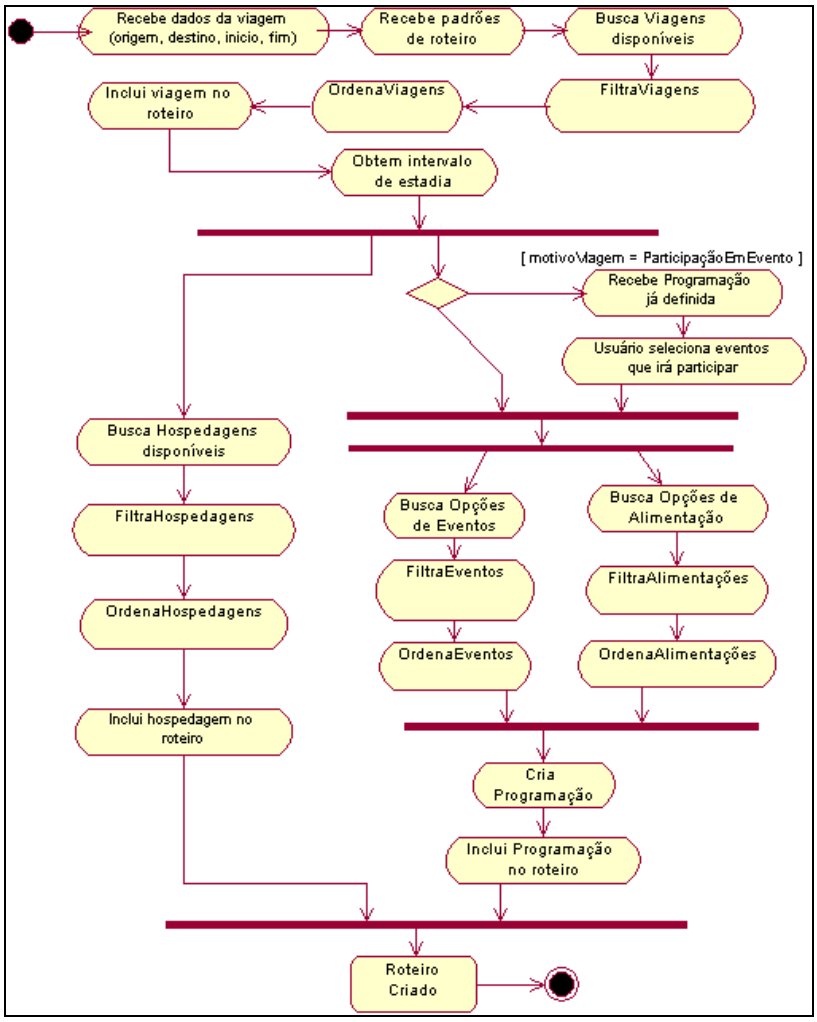

FIGURA 4.6: Diagrama de Atividades do processo de criação do roteiro.

Um diagrama de estado ilustra os eventos e os estados interessantes de um objeto e o comportamento de um objeto em resposta a um evento [Larman, 1998].

Para entender um caso de uso, ele pode ser visto como um gráfico de transição de estados. A cada estimulo enviado pelos atores, alguma atividade é realizada e dependendo do estado em que o sistema se encontra, ele é levado a executar uma transição de estado.

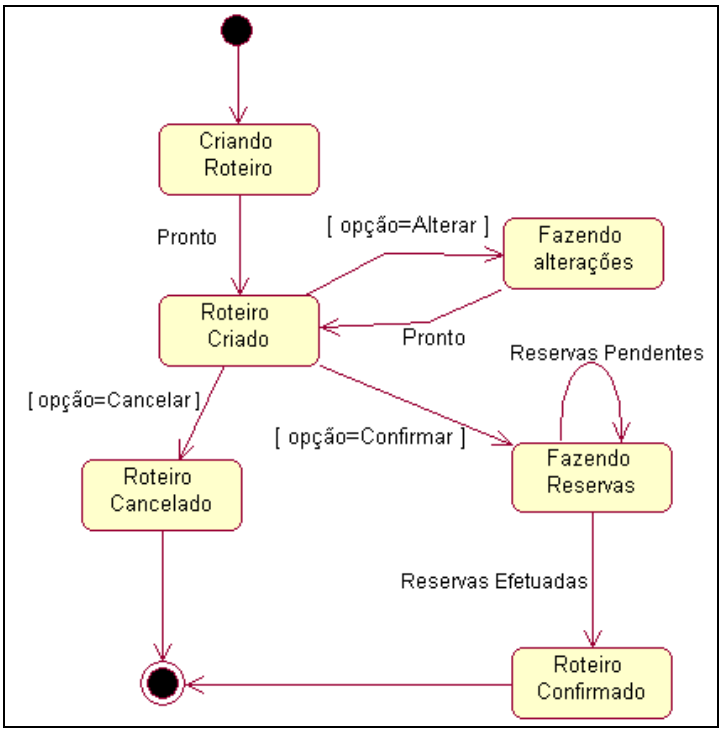

FIGURA 4.7: Diagrama de Estados do processo de criação do roteiro.

Como mostra a FIGURA 4.7, o objeto Roteiro pode passar por seis estados. O primeiro estado, Criando Roteiro, é ativado por um usuário do sistema. Quando o sistema finaliza a atividade de criação (detalhada nos diagramas de seqüência e de atividades anteriores), o Roteiro vai para o estado Criado e o usuário pode então escolher a opção de alterar, cancelar ou confirmar o roteiro. Se a opção for confirmar, o Roteiro entra em um estado de Fazendo Reservas. Assim que todas as reservas forem efetuadas o Roteiro vai para o estado Confirmado. É importante destacar que a escolha da opção alterar pode significar, em alguns casos, uma recriação do resto do roteiro.

Este é um diagrama de estados simplificado, pois o estado Criando Roteiro é bastante complexo e possui uma série de sub-estados.

4.7 Projeto Arquitetural

O projeto arquitetural ilustra a arquitetura do sistema, ou seja, os níveis de comunicação que o sistema terá (camadas) e os diversos módulos existentes nesses níveis.

Na FIGURA 4.8, mostramos a arquitetura do *SEI-Tur*, a qual possui três camadas: Camada de Apresentação, Camada da Lógica da Aplicação e Camada de Dados.

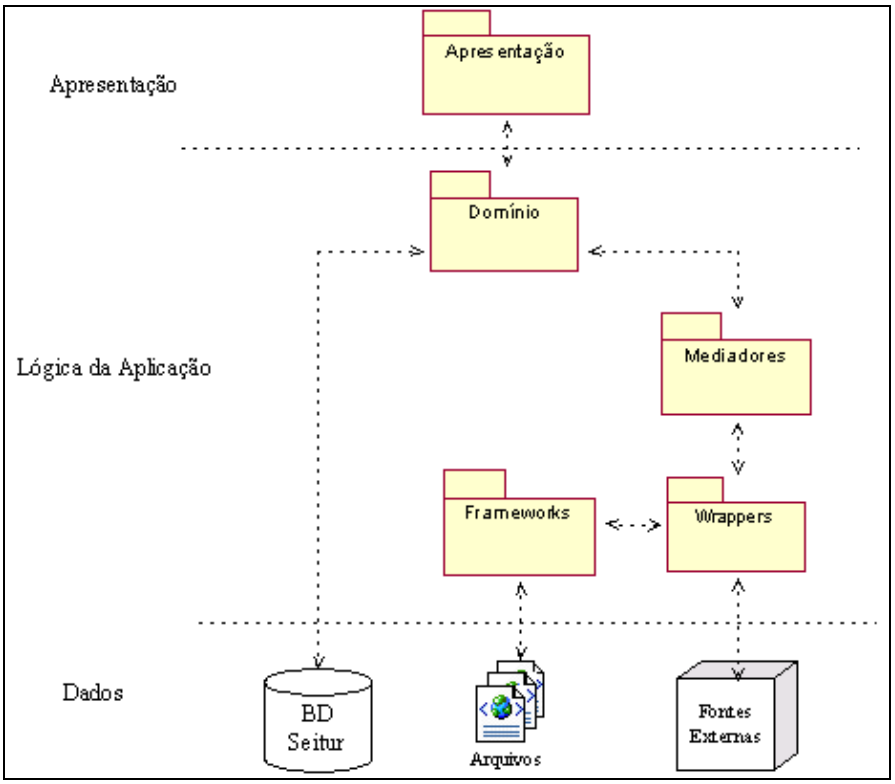

FIGURA 4.8: Diagrama Arquitetural

Através da Camada de Apresentação, os clientes (via browser) interagirão com o *SEI-Tur* por meio de páginas JSP/HTML.

As solicitações são enviadas para a camada da Lógica da Aplicação, a qual engloba a maior parte da aplicação. A Lógica da aplicação interage com a Camada de Dados e com as fontes de Informação Externas, que representam as páginas HTML e os *Web services*.

A terceira camada engloba a base de dados do *SEI-Tur*, a qual possui as informações do usuário, suas preferências e o roteiro de viagens criado. Outras informações, como catálogo de eventos, hospedagens, alimentação, informações de eventos, etc, são armazenadas em arquivos. Estes arquivos incluem os arquivos XML e RDF gerados pelo administrador do sistema ou extraídos de páginas HTML, assim como foi explicado no Capítulo 3. Os arquivos são lidos por meio do *framework Castor*.

4.8 Considerações Finais

Descrevemos neste capítulo, as principais características, da análise até a implementação, envolvidas para construir o *SEI-Tur*.

Descrevemos, segundo a metodologia adotada para o desenvolvimento do nosso sistema, os requisitos funcionais e não funcionais, os casos de uso expandidos. Mostramos os diagramas de classe representando as classes projetadas para o *SEI-Tur*, o diagrama de seqüência, atividades e estado para o processo de criação do roteiro. E por fim, fizemos algumas considerações sobre a arquitetura do sistema.

5. *Estudo de Caso*

Este capítulo apresenta um estudo de caso ilustrando o funcionamento do *SEI-Tur*. Além das viagens de lazer e negócios, o sistema foi preparado para ser utilizado pelos participantes de um congresso fictício[11], que ocorreria na cidade de Campina Grande. Mostraremos os dados disponíveis no sistema que serão utilizados para fazer o casamento de padrões. Na seqüência, apresentamos as interações de um usuário com o sistema.

5.1 **Dados Disponíveis**

Nesta seção, mostramos os dados que foram alimentados no *SEI-Tur* para a execução do estudo de caso.

Um usuário, ao entrar no sistema, deve informar o motivo de sua viagem. Além das opções convencionais de lazer e negócios, o sistema permite inserir participações em eventos programados, como será mostrado na seção seguinte.

5.1.1 **Evento Programado**

Neste estudo de caso, o *SEI-Tur* será utilizado principalmente pelos participantes de um congresso (VII Workshop de Métodos Formais) fictício que será realizado nos dias 26 a 28 de março, na cidade de Campina Grande.

A programação do congresso é mostrada na TABELA 5.1.

[11] Este evento foi inspirado no VI Workshop de Métodos Formais (WMF) que ocorreu na cidade de Campina Grande, em 2003.

Data	Horário	Evento
26/03/2004	08:45 - 10:15	Mini-Curso 1/ Mini-Curso 2
26/03/2004	10:30 - 12:00	Mini-Curso 1/ Mini-Curso 2
26/03/2004	14:00 - 15:30	Mini-Curso 3/ Mini-Curso 4/ Mini-Curso 5
26/03/2004	15:45 - 17:15	Mini-Curso 3/ Mini-Curso 4/ Mini-Curso 5
27/03/2004	09:00 - 10:00	Palestra Convidada 1
27/03/2004	10:30 - 12:00	Sessão Técnica 1
27/03/2004	14:00 - 15:30	Sessão Técnica 2
27/03/2004	16:00 - 18:00	Tutorial 1
28/03/2004	09:00 - 10:00	Palestra Convidada 2
28/03/2004	10:30 - 12:00	Sessão Técnica 3
28/03/2004	14:00 - 15:30	Sessão Técnica 4
28/03/2004	16:00 - 18:00	Tutorial 2

TABELA 5.1: Programação do Congresso.

A programação do congresso é alimentada no sistema através da criação de instâncias RDF do esquema definido na seção 3.2.2. O congresso é modelado como uma instância da classe eventoComposto e os eventos do congresso são modelados como instâncias da classe eventoCongresso.

5.1.2 Viagens

No protótipo implementado, dispomos inicialmente, apenas de viagens aéreas. No entanto, é nosso objetivo disponibilizar outras opções ao usuário.

No caso de viagens aéreas, o SEI-Tur utiliza o sistema netViagens.com disponível na *Internet*. O sistema fornece dados sobre horário de vôos, conexões, preços, etc. O sistema possui um *Web service* que disponibiliza suas operações através de um arquivo WSDL, assim como foi explicado nos capítulos anteriores. Este sistema permite consultar os vôos de várias companhias aéreas disponíveis, de acordo com as informações de origem, destino, data de partida e data de retorno.

5.1.3 Hospedagens

Possuímos três opções de hospedagem, que são mostradas na TABELA 5.2.

Nome	Tipo	Categoria	Facilidades	Endereço	Preços
Hotel Village	Hotel	Cinco estrelas	Piscina, restaurante, sauna, quadra de tênis, ar condicionado.	CEP: 58104-575	R$ 100,00 R$ 150,00
Hotel Serrano	Hotel	Cinco estrelas	Piscina, restaurante, estacionamento, ar condicionado.	CEP: 58100-160	R$ 50,00 R$ 80,00
Pousada Aconchego	Pousada	Duas estrelas	Ar condicionado	CEP: 58101-030	R$ 25,00 R$ 30,00

TABELA 5.2: Opções de Hospedagem.

No momento de sugerir as opções de hospedagem ao usuário, o sistema levará em conta não apenas as características mostradas na tabela, mas também a disponibilidade de quartos. O sistema irá então interagir com o Web service de cada hospedagem e verificar se existem quartos disponíveis. Caso uma hospedagem não possua quartos disponíveis para as datas desejadas, esta será excluída da lista de opções apresentadas ao usuário.

5.1.4 Eventos

As tabelas abaixo exibem os eventos disponibilizados no SEI-Tur. Mostramos apenas os principais atributos, utilizados no casamento de padrões.

Filme	Gênero	Data Inicio	Data Fim	Endereço	Preço	Horários
Alguém tem que ceder	Romance	26/03/2004	02/04/2004	CEP: 58104-901	R$ 8,00	15:00 - 17:00 17:00 - 19:00 19:00 - 21:00 21:00 - 23:00
Fúria em duas Rodas	Ação	26/03/2004	02/04/2004	CEP: 58104-901	R$ 8,00	14:30 – 17:30 17:30 – 20:30 20:30 – 23:30
Pânico na Floresta	Terror	19/04/2004	25/04/2004	CEP: 58104-901	R$ 8,00	15:00 - 17:00 17:00 - 19:00 19:00 - 21:00 21:00 - 23:00
Quero Ficar com Poly	Comédia	26/03/2004	02/04/2004	CEP: 58104-901	R$ 8,00	14:30 – 17:30 17:30 – 20:30 20:30 – 23:30
Revelações	Drama	19/03/2004	02/04/2004	CEP: 58104-901	R$ 8,00	15:00 - 17:00 17:00 - 19:00 19:00 - 21:00 21:00 - 23:00

TABELA 5.3: Opções de Cinema.

Show	Gênero	Data	Endereço	Preço	Horário
Limão com Mel	Forró	26/03/2004	CEP: 58104-590	R$ 10,00	22:00 - 03:00
Babado Novo	Axé	27/03/2004	CEP: 58104-590	R$ 12,00	22:00 - 02:00
Jota Quest	MPB Rock	28/03/2004	CEP: 58104-590	R$ 12,00	19:00 - 22:00

TABELA 5.4: Opções de Show Musical.

Peça	Gênero	Data Inicio	Data Fim	Endereço	Preço	Horários
As Malditas	Comédia	25/03/2004	28/03/2004	CEP: 58103-345	R$ 5,00	19:00 - 21:00

TABELA 5.5: Opções de Teatro.

Passeio	Descrição	Tipo Atividade	Data	Preço	Horários
City Tour	Passeio pelos principais pontos turísticos da cidade.	Paisagem, Arte, Cultura	27/03/2004	R$ 20,00	08:00 - 11:00 13:00 - 15:00
Boqueirão	Passeio no açude de Boqueirão (50 km sudoeste). Atrações: açude, pescaria, passeios de barco.	Esporte, Aventura, Paisagem	28/03/2004	R$ 20,00	08:00 - 12:00 12:00 - 15:00
Museu do Índio	Visita a Lagoa Seca (10 km norte). Atrações: Museu do Índio, artesanato (estopas).	Arte, Ecologia	29/03/2004	R$ 15,00	08:00 - 12:00 12:00 - 15:00

TABELA 5.6: Opções de Excursão.

5.2 Interação com o sistema

Esta seção descreve uma interação típica do usuário com o sistema e mostra algumas funções do sistema. Vamos assumir que o usuário deseja criar um roteiro turístico, dado que o motivo de sua viagem é a participação no Workshop que irá acontecer em Campina Grande.

Inicialmente, este informa os dados iniciais da viagem: motivo e a cidade da qual está partindo. A FIGURA 5.1 mostra esta interação.

FIGURA 5.1: O usuário informa os dados inicias da viagem.

Nós alimentamos o sistema previamente apenas com dados da cidade de Campina Grande. Por isso, o sistema permite que sejam criados apenas roteiros turísticos referentes a esta cidade.

Se o a motivo da viagem selecionado foi a participação no Congresso, o usuário deve informar os eventos do congresso em que ele irá participar (FIGURA 5.2). Neste exemplo, o usuário selecionou os seguintes eventos: Mini-Curso 3, Mini-Curso 4, Palestra Convidada 1, Sessão Técnica 1, Sessão Técnica 4 e Tutorial 2.

Estes eventos são classificados como eventos obrigatórios. Ou seja, durante a criação da programação, estes eventos são fixos, não podendo ser rearranjados no momento da inserção dos eventos opcionais.

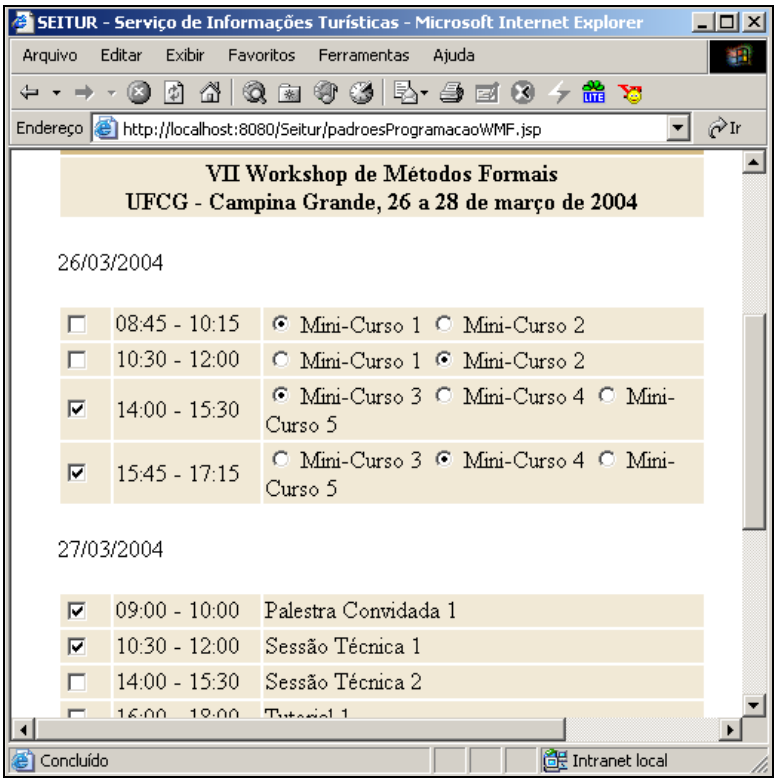

FIGURA 5.2: O usuário seleciona os eventos que irá participar do Congresso.

Em seguida, o usuário informa suas preferências em relação à viagem, criando um padrão de Viagem, como é mostrado na FIGURA 5.3.

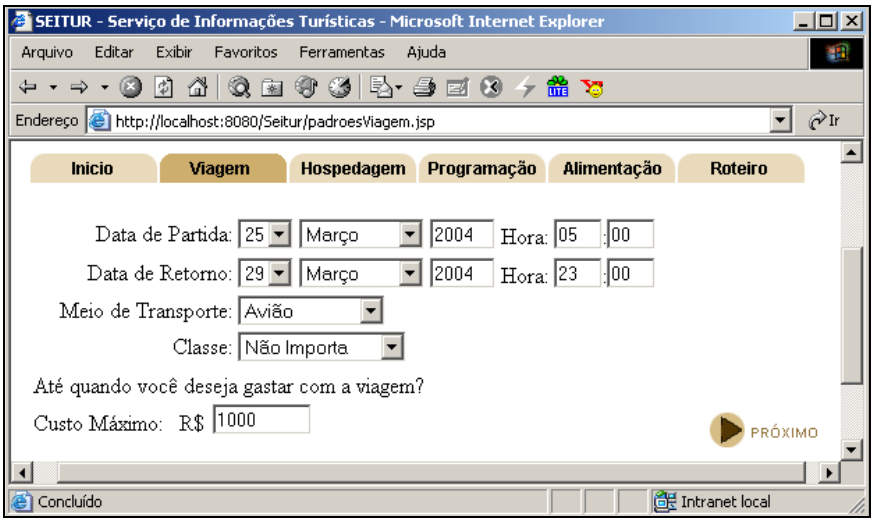

FIGURA 5.3: O usuário define Padrão de Viagem.

Definidos o Padrão de Viagem de Ida e o Padrão de Viagem de Volta, o sistema utiliza estes padrões para fazer sugestões de viagem ao usuário. Como já foi dito anteriormente, o *SEI-Tur* acessa um *Web service* que fornece informações sobre viagens aéreas.

A FIGURA 5.4 mostra as opções fornecidas pelo *SEI-Tur* para a viagem de ida. As opções são ordenadas de acordo com as preferências do usuário. Neste caso, como as duas opções disponíveis possuem o mesmo intervalo, o critério de desempate foi o custo.

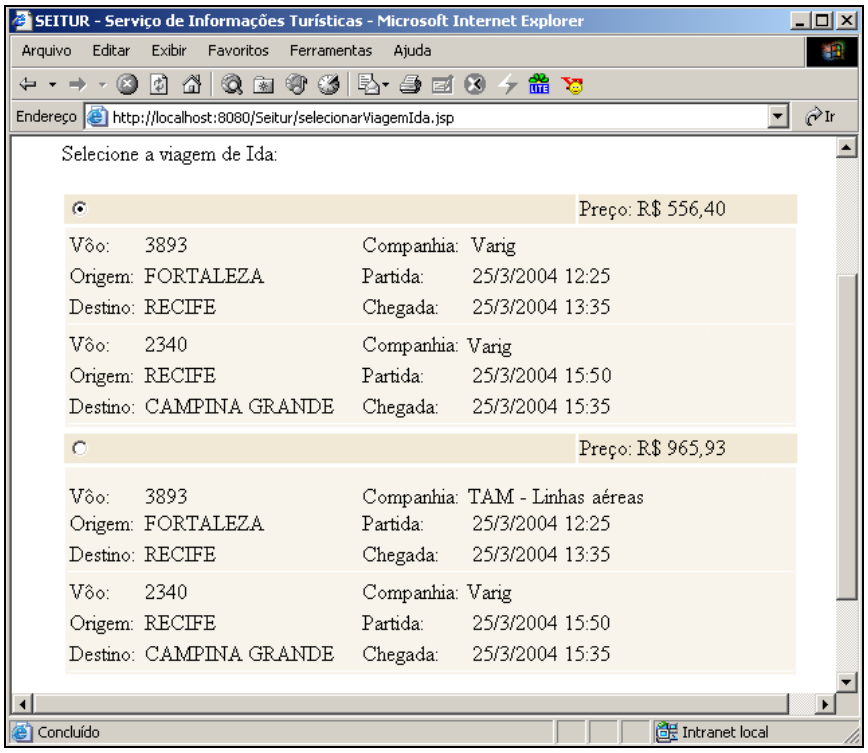

FIGURA 5.4: O usuário Seleciona Viagem de Ida.

Em seguida, o usuário deve selecionar a viagem de volta. Esta interação é semelhante à etapa de selecionar uma viagem de ida. As viagens selecionadas são então inseridas no roteiro. Com isso o sistema passa para a etapa de definição do Padrão de Hospedagem (FIGURA 5.5).

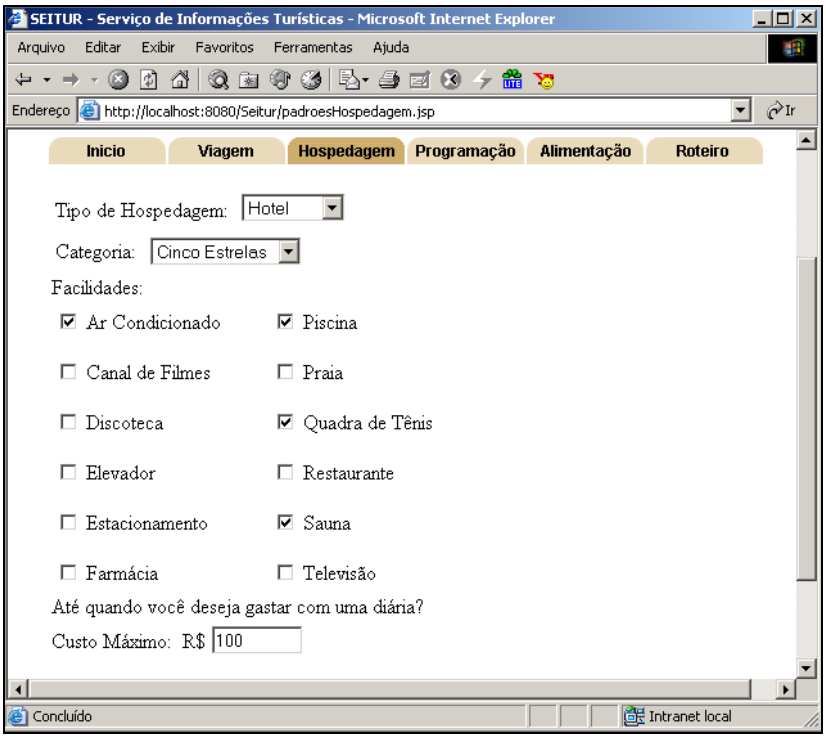

FIGURA 5.5: O usuário define Padrão de Hospedagem.

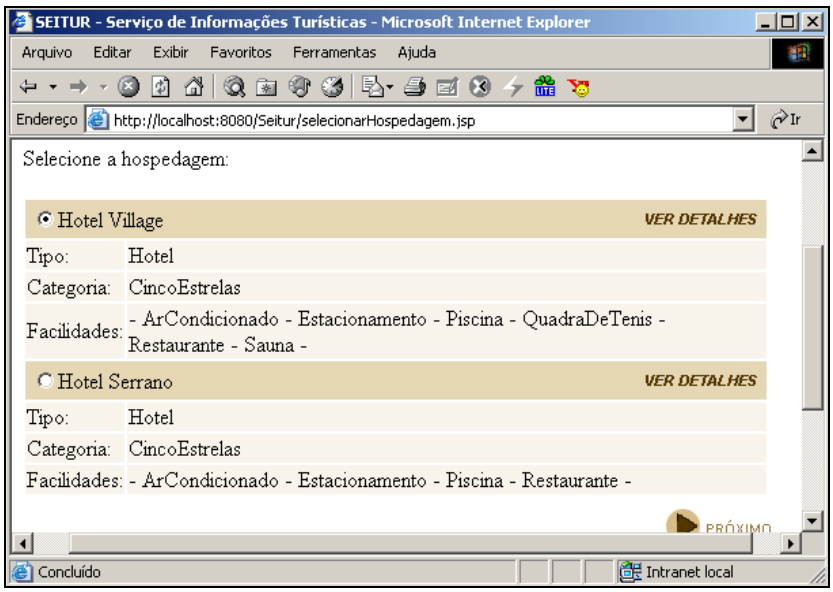

FIGURA 5.6: O usuário Seleciona uma Hospedagem.

Após criado o Padrão de Hospedagem, o sistema sugere as opções disponíveis ao usuário. O usuário pode analisar os detalhes de cada opção e selecionar uma hospedagem que será adicionada ao roteiro.

Em seguida, o usuário informa as atividades que gostaria de fazer em sua viagem. As opções disponíveis neste protótipo são cinema, musica, teatro e excursão. Este informa também suas preferências em relação a cada tipo de evento (FIGURA 5.7).

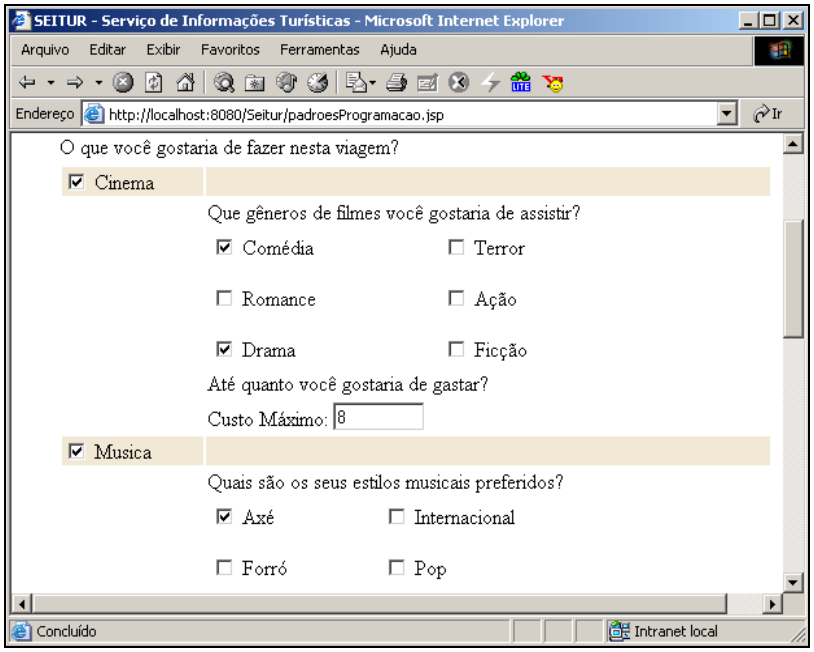

FIGURA 5.7: O usuário define Padrões de Eventos.

Neste exemplo, o usuário informou as preferências por filmes (Comédia e Drama), estilos musicais (Axé e MPB), estilos teatrais (Comédia) e tipos de atividades (Arte, Aventura e Cultura), e custos máximos para os eventos de cinema (R$ 8), show de Musica (R$ 20), teatro (R$ 10) e excursão (R$ 20).

O sistema tenta rearranjar os eventos disponíveis no sistema e que casam com as preferências do usuário, dentro da programação do roteiro. A programação criada com os eventos citados na seção 5.1.4 é mostrada na FIGURA 5.8.

O usuário tem as opções de alterar ou excluir um evento. Alguns eventos da programação podem ser alocados em outro horário e apresentam a opção alterar habilitada, permitindo o usuário escolher outro horário disponível. Eventos que não possuem outros horários disponíveis permitem apenas serem excluídos do roteiro. Desta forma, o usuário pode ajustar a programação proposta pelo sistema. As FIGURAS 5.8 e 5.9 mostram o roteiro completo.

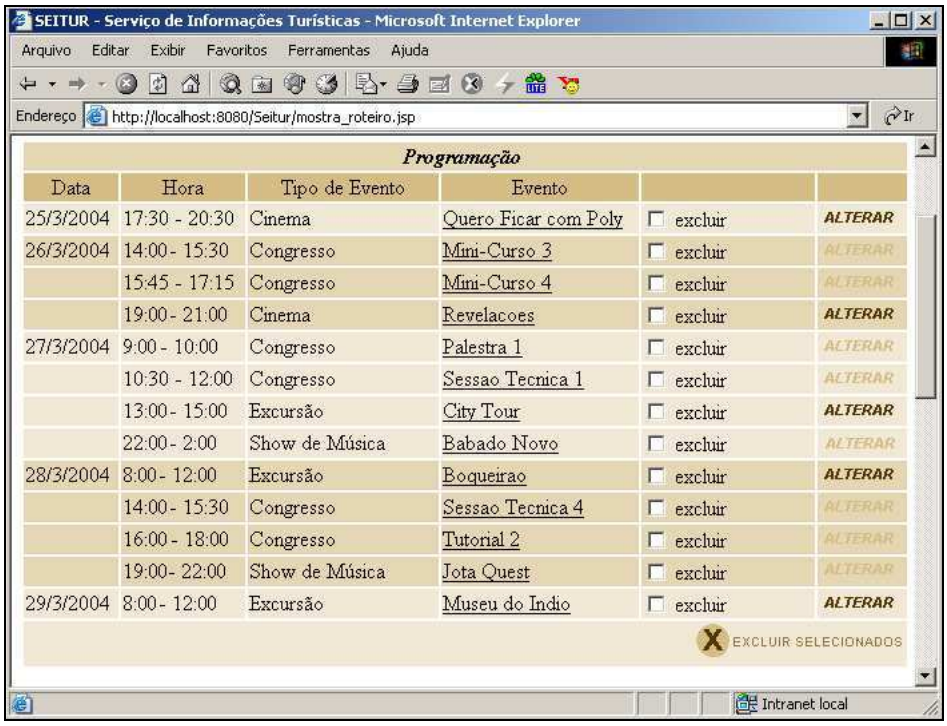

FIGURA 5.8: Programação proposta ao usuário.

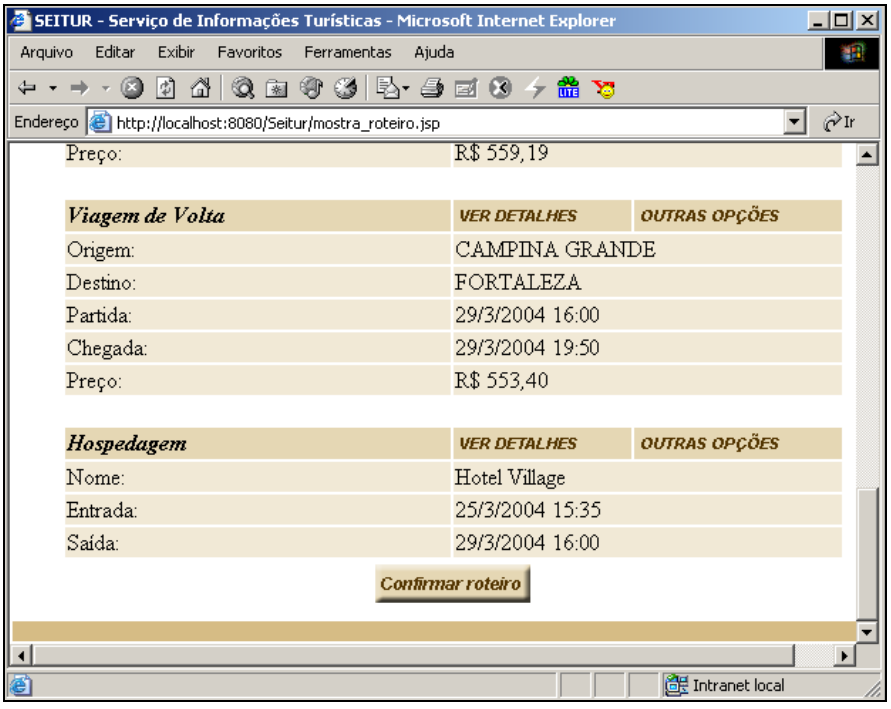

FIGURA 5.9: Itens do roteiro selecionados pelo usuário.

A FIGURA 5.9 mostra todos itens selecionados pelo usuário e inseridos no roteiro. O usuário pode ainda ver os detalhes de cada item, escolher outras opções ou aceitar o roteiro proposto. O processo de confirmação do roteiro implica em fazer as

reservas necessárias dos itens selecionados pelo usuário. Por questões de tempo, esta etapa não foi implementada no protótipo.

5.3 Considerações Finais

Neste capítulo, mostramos um exemplo de utilização do *SEI-Tur*, que foi preparado para ser utilizado também pelos participantes de um congresso fictício realizado na cidade de Campina Grande.

O protótipo implementado não foi completamente concluído. Além da questão de reservas citada acima, também não foi implementado, por questões de tempo, sugestões de Alimentação.

Outra funcionalidade que deve ser adicionada ao protótipo é proporcionar ao usuário a visualização das características de cada item proposto que casam com suas preferências. Deixando o usuário ciente do motivo pelo qual um dado item foi sugerido a ele.

6. Conclusão

Nesta dissertação, apresentamos um Serviço de Informações Turísticas que faz recomendações, auxiliando o usuário a compor um roteiro turístico completo. O sistema pode ser utilizado tanto para organizar viagens de turismo, como de negócios, em particular, a participação em eventos como congressos, conferências, workshops ou festivais.

Várias técnicas de recomendação, como recomendação baseada em conteúdo e colaborativa, permitem filtrar a grande quantidade de informações de modo a satisfazer as necessidades dos usuários.

Sistemas de Recomendação são ferramentas eficientes para a personalização e divulgação de produtos e serviços para o público desejado. Esses sistemas possibilitam que seja oferecido ao cliente o que ele realmente procura, poupando-lhe tempo e esforço em filtrar enormes quantidades de dados. O diferencial do *SEI-Tur* em relação a outros sistemas de recomendação da área de turismo é que ele combina de forma consistente todos componentes envolvidos em uma viagem, ou seja, o transporte, hospedagem e programação local. Na programação local, além dos eventos e facilidades oferecidos pelo local, pode ser incluída a programação de um evento científico. Assim, o sistema pode ser particularmente útil como suporte à organização de congressos, simpósios e festivais.

As necessidades do cliente são registradas na forma de padrões. Estes padrões de roteiro permitem ao sistema calcular a similaridade dos dados disponíveis com as preferências do usuário. A criação de padrões proporciona uma filtragem de dados baseada em conteúdo, onde estes são confrontados com os dados disponíveis do sistema retornando uma medida de similaridade. Os padrões também permitem a criação de regras que permitem manter a consistência temporal entre os elementos do roteiro.

O *SEI-Tur* combina dados cadastrais armazenados em um banco de dados próprio com dados dinâmicos disponíveis na *Web*. A utilização de *Web services*

permitiu acessar dados de outros sistemas que sofrem atualizações freqüentes, como a disponibilidade de quartos em um hotel, para agregar valor às funcionalidades do *SEI-Tur*. A interoperabilidade pôde ser obtida pois estes usam a linguagem XML como um protocolo para troca de informações entre sistemas heterogêneos.

Informações disponíveis em páginas estáticas foram obtidas através da descrição de metadados utilizando a tecnologia RDF. Definimos um esquema RDF para a modelagem de eventos. Esta abordagem acarreta no problema das fontes de dados terem que se adequar a este padrão. No entanto, nosso objetivo é recuperar dados da *Web* utilizando as abordagens existentes. Muitas pesquisas têm sido feitas focando interoperabilidade na *Web*. A arquitetura em camadas proposta pelo *SEI-Tur* permite que novas abordagens de recuperação de informação sejam incorporadas, adaptando-se apenas alguns módulos do sistema.

Desenvolvemos um protótipo visando validar a solução proposta no projeto do *SEI-Tur*. Como não foram encontrados *Web services* reais adequados às necessidades do sistema, foram criados *Web services* hipotéticos de alguns hotéis em Campina Grande. Esta solução permitiu testar as funcionalidades do sistema, independente da existências de serviços reais adequados. Esperamos que a tendência seja que cada vez mais empresas instalem estes serviços, facilitando o acesso público a seus produtos.

No entanto, devido ao tempo, alguns questões não foram implementadas, como por exemplo: sugerir uma programação incluindo opções de alimentação; sugerir outras opções de viagem, além de viagens aéreas; fazer as reservas necessárias dos elementos do roteiro selecionados pelo usuário.

6.1 Limitações da versão atual

O SEI-Tur, em sua versão atual, possui algumas limitações que serão apresentadas a seguir:

- As aplicações que desejam integrar seus dados sobre eventos ao *SEI-Tur* devem adequar-se o esquema de eventos definido na representação de suas instâncias.

- Atualmente o SEI-Tur não é suficientemente genérico para criar um roteiro de viagens abrangendo várias cidades. Está preparado para criar roteiros que consideram apenas uma cidade de destino.
- Os eventos alimentados no sistema são eventos que ocorrerão em uma cidade em intervalos de tempo bastante próximo, não permitindo a um usuário criar uma programação para uma viagem com muito tempo de antecedência.

6.2 **Trabalhos Futuros**

A seguir citamos alguns trabalhos que podem ser desenvolvidos a partir do *SEI-Tur*:

- Aplicar outras metodologias de recomendação que explorem consultas passadas de outros usuários para fazer recomendações.
- Manter a consistência espaço-temporal dos elementos do roteiro, ou seja, considerar o tempo de deslocamento entre dois elementos.
- Permitir que o usuário especifique pesos para suas preferências, ou seja, definir quais atributos são mais importantes e que devem ser priorizados.
- Desenvolver uma GUI para dar suporte às aplicações que desejam integrar-se ao *SEI-Tur*, no momento de inserção de sentenças RDF, obedecendo ao esquema proposto, em suas páginas HTML.
- Realizar uma análise de desempenho e escalabilidade do sistema proposto.
- Implementar as funcionalidades que não foram realizadas, que são: outros tipos de viagem; programa de alimentação; e interagir com outros sistemas realizando reservas e pagamentos.

Referências Bibliográficas

[Allen, 1983] Allen, J. *Maintaining knowledge about Temporal Intervals*. Comunications of ACM, 26(11), 1983, pp. 832-843.

[Austin *et al*, 2002] Austin, D., Barbier, A., Ferris, C., et al. *Web Services Architecture Requirements*, W3C Working Draft, Agosto 2002. - disponível em: http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/

[Berners-Lee e*t al*, 2001] Berners-Lee, T., Hendler, J., Lassila, O. *The Semantic Web.* Scientific American, Maio 2001.

[Braganholo, 2001] Braganholo, V., Heuser, C. *XML Schema, RDF(S) e UML: uma comparação*. In: IDEAS 2001 - 4th Iberoamerican Workshop on Requirements Engineering and Software Environments, Santo Domingo, Heredia, Costa Rica, 2001. Proceedings... page 78-90. - disponível em: http://www.inf.ufrgs.br/~vanessa//artigos/ideas2001.pdf

[Bray, 1999] Bray, T. *XML Namespaces by Example*. - disponível em: http://www.xml.com/pub/a/1999/01/namespaces.html.

[Bray *et al.*, 2000] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation, Outubro 2000.

[Burke, 2002] Burke, R. *Hybrid Recommender Systems: Survey and Experiments*, User Modeling and User-Adapted Interaction, vol. 12 , Issue 4, Novembro 2002. pp. 331-370.

[Curbera *et al*., 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S. *Unraveling the Web Services Web: An*

Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, vol. 6, no. 2, Março/Abril 2002, pp. 86-93.

[Delgado *et al*, 2002] Delgado, J., Davidson, R. *Knowledge bases and user profiling in travel and hospitality recommender systems*. In Proceedings of the ENTER 2002 Conference, Innsbruck, Austria, Janeiro 2002, pp 1-16.

[Dell'Erba *et al*, 2002] Dell'Erba, M., Fodor, O., Ricci, F., Werthner, H. *Harmonise: a solution for data interoperability*, in Proceedings of the 2nd IFIP Conf. on E-Commerce, E-Business & E-Government, Lisboa, Portugal, Outubro 2002.

[Fremantle et al., 2002] Fremantle, P., Weerawarana, S., Klalaf, R. *Enterprise Services - Examining the Emerging Field of Web Services and How it is Integrated into existing enterprise infrastructures*, Communications of the ACM, Outubro 2002, Vol.45, No.10, pp. 77-82.

[Furlan, 1998] Furlan, José Davi. *Modelagem de Objetos UML - The Unified Modeling Language* . São Paulo : Makron Books, 1998.

[Grimnes *et al*, 2003] Grimnes, G. A., Chalmers, S., Edwards, P., Preece, A. D. *GraniteNights - A Multi-agent Visit Scheduler Utilising Semantic Web Technology*. Seventh International Workshop on CIA, Finlandia, 2003, pp. 137-151.

[Izeki, 2001] Izeki, C. A. *Anotações Colaborativas como Hiperdocumentos de Primeira Classe*. Dissertação (Mestrado), Universidade de São Paulo, São Carlos. Outubro 2001.

[Jacobson, 1992] Jacobson, I. et al. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley, Reading/Mass, 1992.

[Kao, 2001] Kao, James. *Developer's Guide to Building XML-based Web Services*. - disponível em: http://www.theserverside.com/articles/article.jsp?l=WebServices-Dev-Guide Acessado em 02/03/2004.

[Larman, 1998] Larman, C. *Applying UML and Patterns.* Prentic-Hall, USA, 1998.

[Lassila & Swich,1999] Lassila, O., Swick, R. R. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, 1999 – disponível em http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ Acessado em 29/02/2004

[Marino, 2001] Marino, M. Teresa. Integração de Informações em Ambientes Científicos na Web: Uma Abordagem Baseada na Arquitetura RDF. Dissertação (Mestrado), Rio de Janeiro: UFRJ/IM/NCE, 2001. - disponível em: http://genesis.nce.ufrj.br/dataware/Metadados/Teses/Teresa/. Acessado em 01/03/2004.

[Menéndez, 2002] Menéndez, Andrés Ignácio Martinez. *Uma ferramenta de apoio ao desenvolvimento de Web Services*. Dissertação (Mestrado), Campina Grande: UFCG/COPIN, 2002.

[Protégé, 2000] Protégé-2000 - disponível em http://protege.stanford.edu/download.html .Acessado em 28/02/2004.

[Ricci *et al*, 2002] Ricci, F., Arslan, B., Mirzadeh, N., Venturini, A. *Itr: a case-based travel advisory system*. In S. Craw, editor, 6th European Conference on Case Based Reasoning, ECCBR 2002, Aberdeen, Scotland, Setembro 2002.

[Ricci *et al*, 2003] Ricci, F., Venturini , A., Cavada , D., Mirzadeh , N.Blaas , D., Nones M. *Product Recommendation with Interactive Query Management and Twofold Similarity.* Proceedings of ICCBR 2003, the 5th Internatinal Conference on Case-Based Reasoning, Trondheim, Norway, Junho 2003.

[Santos, 2002] Santos, Domingos S. A. *RDF na Interoperabilidade entre domínios na Web*. Dissertação (Mestrado), Campina Grande: UFCG/COPIN, 2002.

[Schafer *et al*, 2001] Schafer, J.B., Konstan, J.A., Riedl, J. *ECommerce Recommendation Applications*, Data Mining and Knowledge Discovery, vol. 5, pp. 115-153, 2001.

[Schiel, 96] Schiel, U. *Aspectos Temporais em Sistemas de Informação*, Relatório Técnico 002/96, UFPB, Campina Grande, PB, Brasil, 1996.

[Sun-WS, 2003] Sun Microsystems. *The Java Web Service Tutorial* - disponível em: http://java.sun.com/webservices/docs/1.3 /tutorial/doc/index.html Acessado em 10/03/2004.

[VacationCoach, 2001] VacationCoach. *Using Knowledge Personalization to Sell Complex Products*. White Paper. Novembro 2001 - disponível em http://crm.ittoolbox.com/documents/document.asp?i=2003 Acessado em 10/03/2004.

[W3C] World Wide Web Consortium - disponível em: http://www.w3.org.

[W3C-HTML] Hypertext Markup Language (*HTML*). W3C – disponível em http://www.w3.org/MarkUp/ Acessado em 28/02/2004.

[W3C-RDF] Resource Description Framework (*RDF*), W3C – disponível em http://www.w3.org/RDF/ Acessado em 28/02/2004.

[W3C-RDFSchema] Resource Description Framework Schema (*RDF Schema*) – disponível em http://www.w3.org/TR/rdf-schema/ Acessado em 28/02/2004.

[W3C-SOAP] Simple Object Access Protocol (SOAP) 1.2, W3C - disponível em http://www.w3.org/TR/soap12 Acessado em 03/03/2004.

[Wilson *et al*, 1997] Wilson, D., Martinez, T. *Improved heterogeneous distance functions*. Journal of Artificial Intelligence Research, vol.6, pp. 1-34, 1997.

Apêndice A – Esquema RDF do domínio Eventos

Este anexo mostra o conteúdo, completo, do RDFS gerado pela ferramenta protégé, referente ao diagrama de classes exibido na FIGURA 3.2.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:a="http://protege.stanford.edu/system#" xmlns:rdfs="http://www.w3.org/TR/1999/PR-
    rdf-schema-19990303#"
    xmlns:Evento="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#">
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Artista"
      rdfs:label="Artista">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema"
      rdfs:label="Cinema">
    <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Custo"
      rdfs:label="Custo">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data"
      rdfs:label="Data">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco"
      rdfs:label="Endereco">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento"
      rdfs:label="Evento">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
  </rdfs:Class>
  <rdfs:Class
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoComposto"
      rdfs:label="EventoComposto">
    <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
  </rdfs:Class>
```

```
<rdfs:Class
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoCongresso"
    rdfs:label="EventoCongresso">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
</rdfs:Class>
<rdfs:Class
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimples"
    rdfs:label="EventoSimples">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Excursao"
    rdfs:label="Excursao">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Hora"
    rdfs:label="Hora">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Horario"
    rdfs:label="Horario">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
</rdfs:Class>
<rdfs:Class
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#HorarioEvento"
    rdfs:label="HorarioEvento">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local"
    rdfs:label="Local">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
</rdfs:Class>
<rdfs:Class
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#OcorrenciaEvento"
    rdfs:label="OcorrenciaEvento">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Resource" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Outros"
    rdfs:label="Outros">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#ShowMusical"
    rdfs:label="ShowMusical">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
</rdfs:Class>
<rdfs:Class rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Teatro"
    rdfs:label="Teatro">
   <rdfs:subClassOf
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
      es" />
</rdfs:Class>
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#ano"
    a:maxCardinality="1" a:range="integer" rdfs:label="ano">
   <rdfs:domain
      rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data" />
   <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#Literal" />
```

```
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#anoLancamento"
        a:maxCardinality="1" rdfs:label="anoLancamento">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#apresentador"
        rdfs:label="apresentador">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoCongr
          esso" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#artista"
        rdfs:label="artista">
      <rdfs:range
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Artista" />
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema" />
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#ShowMusical
          " />
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Teatro" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#bairro"
        a:maxCardinality="1" rdfs:label="bairro">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#capacidade"
        a:maxCardinality="1" a:range="integer" rdfs:label="capacidade">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#cep"
        a:maxCardinality="1" rdfs:label="cep">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#cidade"
        a:maxCardinality="1" rdfs:label="cidade">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#custo"
        rdfs:label="custo">
      <rdfs:range
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Custo" />
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#OcorrenciaE
          vento" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#dataFim"
        a:maxCardinality="1" rdfs:label="dataFim">
      <rdfs:range
```

```
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#dataInicio"
        a:maxCardinality="1" rdfs:label="dataInicio">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#descricao"
        a:maxCardinality="1" rdfs:label="descricao">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Excursao" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#dia"
        a:maxCardinality="1" a:range="integer" rdfs:label="dia">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#diasSemana"
        a:allowedValues="Ter" a:range="symbol" rdfs:label="diasSemana">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#HorarioEvent
            o" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
        <a:allowedValues>Dom</a:allowedValues>
        <a:allowedValues>Qua</a:allowedValues>
        <a:allowedValues>Qui</a:allowedValues>
        <a:allowedValues>Sab</a:allowedValues>
        <a:allowedValues>Seg</a:allowedValues>
        <a:allowedValues>Sex</a:allowedValues>
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#endereco"
        a:maxCardinality="1" rdfs:label="endereco">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#estado"
        a:maxCardinality="1" rdfs:label="estado">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#evento"
        rdfs:label="evento">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#eventos"
        rdfs:label="eventos">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoComp
            osto" />
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoSimpl
            es" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#facilidades"
```

```
        a:range="symbol" rdfs:label="facilidades">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
      <a:allowedValues>AcessoDeficientes</a:allowedValues>
      <a:allowedValues>ArCondicionado</a:allowedValues>
      <a:allowedValues>Estacionamento</a:allowedValues>
      <a:allowedValues>SomDolbyDigital</a:allowedValues>
  </rdf:Property>
  <rdf:Property
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#generoFilme"
      a:range="symbol" rdfs:label="generoFilme">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
      <a:allowedValues>Acao</a:allowedValues>
      <a:allowedValues>Comedia</a:allowedValues>
      <a:allowedValues>Drama</a:allowedValues>
      <a:allowedValues>Ficcao</a:allowedValues>
      <a:allowedValues>Romance</a:allowedValues>
      <a:allowedValues>Terror</a:allowedValues>
  </rdf:Property>
  <rdf:Property
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#generoMusical"
      a:range="symbol" rdfs:label="generoMusical">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#ShowMusical
          " />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
      <a:allowedValues>Axe</a:allowedValues>
      <a:allowedValues>Forro</a:allowedValues>
      <a:allowedValues>Internacional</a:allowedValues>
      <a:allowedValues>MPB</a:allowedValues>
      <a:allowedValues>Pagode</a:allowedValues>
      <a:allowedValues>Pop</a:allowedValues>
      <a:allowedValues>Rock</a:allowedValues>
      <a:allowedValues>Sertanejo</a:allowedValues>
  </rdf:Property>
  <rdf:Property
      rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#generoTeatro"
      a:range="symbol" rdfs:label="generoTeatro">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Teatro" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
      <a:allowedValues>Comedia</a:allowedValues>
      <a:allowedValues>Drama</a:allowedValues>
      <a:allowedValues>Ficcao</a:allowedValues>
      <a:allowedValues>Infantil</a:allowedValues>
      <a:allowedValues>Musical</a:allowedValues>
      <a:allowedValues>Romance</a:allowedValues>
  </rdf:Property>
  <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#hora"
      a:maxCardinality="1" a:range="integer" rdfs:label="hora">
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Hora" />
      <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
          19990303#Literal" />
  </rdf:Property>
  <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#horaFim"
      a:maxCardinality="1" rdfs:label="horaFim">
      <rdfs:range
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Hora" />
      <rdfs:domain
          rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Horario" />
```

```
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#horaInicio"
        a:maxCardinality="1" rdfs:label="horaInicio">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Hora" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Horario" />
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#horarioEvento"
        a:maxCardinality="1" rdfs:label="horarioEvento">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#HorarioEvent
            o" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#OcorrenciaE
            vento" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#horarios"
        rdfs:label="horarios">
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Horario" />
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#HorarioEvent
            o" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#local"
        a:maxCardinality="1" rdfs:label="local">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
        <rdfs:range
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#logradouro"
        a:maxCardinality="1" rdfs:label="logradouro">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Endereco" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#mes"
        a:maxCardinality="1" a:range="integer" rdfs:label="mes">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Data" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#minuto"
        a:maxCardinality="1" a:range="integer" rdfs:label="minuto">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Hora" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#nome"
        a:maxCardinality="1" rdfs:label="nome">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Artista" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#nomeEvento"
        a:maxCardinality="1" rdfs:label="nomeEvento">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
```

```
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#nomeLocal"
    a:maxCardinality="1" rdfs:label="nomeLocal">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
</rdf:Property>
<rdf:Property
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#ocorrenciaEvento"
    rdfs:label="ocorrenciaEvento">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Evento" />
  <rdfs:range
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#OcorrenciaE
    vento" />
</rdf:Property>
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#origem"
    a:maxCardinality="1" rdfs:label="origem">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
</rdf:Property>
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#papel"
    a:allowedValues="escritor" a:maxCardinality="1" a:range="symbol" rdfs:label="papel">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Artista" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
  <a:allowedValues>ator</a:allowedValues>
  <a:allowedValues>autor</a:allowedValues>
  <a:allowedValues>cantor</a:allowedValues>
  <a:allowedValues>diretor</a:allowedValues>
</rdf:Property>
<rdf:Property
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#responsavel"
    a:maxCardinality="1" rdfs:label="responsavel">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Excursao" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
</rdf:Property>
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#telefone"
    a:maxCardinality="1" rdfs:label="telefone">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Local" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
</rdf:Property>
<rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#tipo"
    a:maxCardinality="1" a:range="symbol" rdfs:label="tipo">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoCongr
    esso" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
  <a:allowedValues>Mini-curso</a:allowedValues>
  <a:allowedValues>Palestra</a:allowedValues>
  <a:allowedValues>SessaoTecnica</a:allowedValues>
  <a:allowedValues>Tutorial</a:allowedValues>
</rdf:Property>
<rdf:Property
    rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#tipoAtividade"
    a:allowedValues="Relaxar" a:range="symbol" rdfs:label="tipoAtividade">
  <rdfs:domain
    rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Excursao" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
    19990303#Literal" />
  <a:allowedValues>Arte</a:allowedValues>
```

```
        <a:allowedValues>Aventura</a:allowedValues>
        <a:allowedValues>Cultura</a:allowedValues>
        <a:allowedValues>Ecologia</a:allowedValues>
        <a:allowedValues>Esporte</a:allowedValues>
        <a:allowedValues>Gastronomia</a:allowedValues>
        <a:allowedValues>Meio-ambiente</a:allowedValues>
        <a:allowedValues>Paisagem</a:allowedValues>
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#tipoEvento"
        a:maxCardinality="1" rdfs:label="tipoEvento">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Outros" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property
        rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#tipoIngresso"
        a:maxCardinality="1" a:range="symbol" rdfs:label="tipoIngresso">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Custo" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
        <a:allowedValues>arquibancada</a:allowedValues>
        <a:allowedValues>camarote</a:allowedValues>
        <a:allowedValues>individual</a:allowedValues>
        <a:allowedValues>mesa</a:allowedValues>
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#titulo"
        a:maxCardinality="1" rdfs:label="titulo">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#EventoCongr
            esso" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#tituloFilme"
        a:maxCardinality="1" rdfs:label="tituloFilme">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Cinema" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
    <rdf:Property rdf:about="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#valor"
        a:maxCardinality="1" a:range="float" rdfs:label="valor">
        <rdfs:domain
            rdf:resource="http://www.dsc.ufcg.edu.br/~sei/seitur/schemas/Evento#Custo" />
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
            19990303#Literal" />
    </rdf:Property>
</rdf:RDF>
```