

Universidade Estadual de Maringá
Centro de Tecnologia – Departamento de Informática
Especialização em Desenvolvimento de Sistemas para *Web*

**Arquitetura Orientada a Serviços
em Aplicações Web Sensíveis ao Contexto**

Rafael Leonardo Vivian

Profa. Dra. Elisa Hatsue Moriya Huzita
Orientadora

Maringá, 2010

Universidade Estadual de Maringá
Centro de Tecnologia – Departamento de Informática
Especialização em Desenvolvimento de Sistemas para *Web*

Rafael Leonardo Vivian

**Arquitetura Orientada a Serviços
em Aplicações Web Sensíveis ao Contexto**

**Trabalho submetido à Universidade Estadual de Maringá
como requisito para obtenção do título de Especialista
de Desenvolvimento de Sistemas para *Web*.**

Orientadora: Profa. Dra. Elisa Hatsue Moriya Huzita

Maringá, 2010

Universidade Estadual de Maringá
Centro de Tecnologia – Departamento de Informática
Especialização em Desenvolvimento de Sistemas para *Web*

Rafael Leonardo Vivian

**Arquitetura Orientada a Serviços
em Aplicações Web Sensíveis ao Contexto**

Maringá, ____ de _____ de 2010.

Profa. Dra. Elisa Hatsue Moriya Huzita (orientadora) Ass.: _____

Profa. Dra. Tânia Fátima Calvi Tait Ass.: _____

Profa. Me. Ana Paula Chaves Ass.: _____

*Aos meus pais Maria e Delcino Vivian,
pelo amor, dedicação e compreensão.*

AGRADECIMENTOS

Agradeço primeiramente a Deus pela força e proteção que me acompanharam durante todo o meu trabalho e nas minhas viagens.

Aos meus pais Maria e Delcino Vivian por todo o amor, paciência, carinho e principalmente por todo o incentivo. Sem a força e o apoio irrestrito dos meus amados pais, eu não encontraria forças para a realização desta tarefa e morar longe da minha família.

Agradeço, com muito carinho, a minha namorada Dayana por estar sempre ao meu lado em todas as dificuldades e compartilhar momentos de alegria. “*No meio de tudo, você!*”.

Agradeço também ao meu irmão Gabriel pelo apoio e caronas entre Coronel Freitas e Chapecó.

Agradeço em especial a minha orientadora Profa. Dra. Elisa Hatsue Moriya Huzita pela oportunidade, confiança, paciência, dedicação e conhecimentos compartilhados comigo.

Agradeço aos professores participantes da banca examinadora Profa. Dra. Tânia Fátima Calvi Tait e Profa. Me. Ana Paula Chaves.

Também agradeço ao restante da minha família (avós, tios, primos, padrinhos...) por todo apoio e carinho oferecido a mim.

Aos meus amigos (em Santa Catarina: Marcos, Ita, Marcio, Rafaela, Maira, Juninho, Negão, Rudi, Lorenzon, Sediane; no Paraná: Neto, Marcos, Vanessa, Marlon, André, Marcelo, Felipe) agradeço o apoio, a compreensão e o carinho. Aos velhos amigos de SC: “*A distância não separabólica.*”

Agradeço a todos os colegas da turma 2008 da especialização. O companheirismo e a cumplicidade sempre permanecerão.

Agradeço a todos os professores do curso de Especialização em Desenvolvimento de Sistemas para *Web*, principalmente os professores Ayslan Trevizan Possebom, Edson Alves de Oliveira Junior e Flávio Luiz Schiavoni. E também ao coordenador do curso Prof. Dr. Wesley Romão pela paciência e dedicação.

Agradeço a todos os funcionários da secretaria do Departamento de Informática (DIN) da Universidade Estadual de Maringá (UEM).

Por fim, agradeço aos grandes compositores Neil Young, Bob Dylan, Roger Waters, David Gilmour, Humberto Gessinger, Bono Vox... que sempre me acompanharam com suas canções, servindo como inspiração durante a realização deste trabalho.

*“Um pouco de silêncio
e um copo de água pura.”*

HG

RESUMO

Aplicações sensíveis ao contexto adaptam e alteram dinamicamente o seu comportamento com base em informações contextuais, fornecendo outras informações relevantes e proporcionando maior comodidade ao usuário. Entretanto, o desenvolvimento dessas aplicações envolve vários desafios relacionados à captura, representação, armazenamento, processamento e apresentação da informação contextual. Nesse sentido, há a necessidade de uma arquitetura adequada para tal, e assim, apoiar o desenvolvimento de aplicações sensíveis ao contexto usando serviços para a aquisição e apresentação de informações contextuais, particularmente quando se considera aplicações *Web*. Esta monografia investiga a arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto, que por meio de um conjunto de especificações e protocolos promove a aquisição de informações de contexto provenientes de diversos sistemas heterogêneos. Para tal, foi implementado um protótipo que consome serviços de outros sistemas por meio de *Web services* e assim, obtém e apresenta informações de acordo com o contexto do usuário.

Palavras-chave: percepção de contexto, aplicações *Web*, arquitetura orientada a serviços, serviços *Web*.

ABSTRACT

Context-aware applications adapt and change dynamically their behavior based on contextual information, providing other relevant information and providing greater convenience to user. However, the development of these applications involves several challenges related to the capture, representation, storage, processing and presentation of contextual information. In this sense, there is a need for an architecture appropriate to do so, and so, support the development of context-aware applications using services for the acquisition and presentation of contextual information, particularly when considering Web applications. This monograph investigates the service-oriented architecture in Web context-aware applications, that through a set of specifications and protocols promoting the acquisition of context information from various heterogeneous systems. To this end, we implemented a prototype that uses services from other systems through Web services and so, retrieves and displays information in accordance with the user context.

Key-words: context-aware, Web applications, service-oriented architecture, Web services.

LISTA DE ILUSTRAÇÕES

Figura 2.1: Aplicações Tradicionais.....	23
Figura 2.2: Aplicações Sensíveis ao Contexto.....	23
Figura 3.1: Arquitetura básica de Aplicações <i>Web</i>	34
Figura 3.2: Modelo de processo de desenvolvimento de aplicações <i>Web</i> (MURUGESAN e GINIGE, 2005).....	39
Figura 4.1: Papéis de <i>Web services</i> (CERAMI, 2002).....	44
Figura 4.2: Papéis, operações e artefatos de <i>Web services</i> (KREGGER, 2001).....	45
Figura 4.3: Camadas de protocolos de <i>Web services</i> (CERAMI, 2002).....	46
Figura 4.4: Elementos de uma mensagem SOAP (CERAMI, 2002).....	48
Figura 4.5: Mensagem SOAP de uma requisição cliente.....	49
Figura 4.6: Mensagem SOAP de resposta.....	49
Figura 4.7: Elementos da especificação WSDL (CERAMI, 2002).....	50
Figura 4.8: Exemplo de arquivo WSDL.....	51
Figura 5.1: Modelo baseado em <i>context-awareness</i> para o DiSEN (CHAVES, 2009).....	55
Figura 5.2: Exemplo de configuração do <i>Suporte ao processamento</i> (adaptado de COSTA, 2003).....	58
Figura 5.3: Arquitetura orientada a serviços em aplicações <i>Web</i> sensíveis ao contexto.....	60
Figura 6.1: Cenário de exemplo.....	63
Figura 6.2: Gerenciar Viagem Use Case.....	64
Figura 6.3: Arquitetura do <i>Contexttrip</i>	65
Figura 6.4: Trecho de código da classe <i>CapturaContexto</i>	66
Figura 6.5: Representação de informações contextuais.....	67
Figura 6.6: Apresentação de informações contextuais.....	67
Figura A1: Visão de Negócio.....	75
Figura A2: Modelo de Negócio para Organização de Viagem.....	75
Figura A3: Arquitetura Inicial.....	76
Figura A4: Gerenciar Hospedagem Use Case.....	76
Figura A5: Gerenciar Passagem Use Case.....	77
Figura A6: Gerenciar Passeio Use Case.....	77
Figura A7: Gerenciar Perfil Use Case.....	78

Figura A8: Gerenciar Veículo Use Case.....	78
Figura A9: Gerenciar Viagem Use Case.....	79
Figura A10: Modelo de Objeto de Negócio.....	79
Figura A11: Diagrama de Componente.....	80
Figura A12: Diagrama de Classes Análise.....	81
Figura A13: Diagrama de Classes Projeto.....	82

LISTA DE TABELAS

Tabela 2.1: Categorias de Contexto, conforme Schilit et al. (1994).....	22
Tabela 2.2: Dimensões de Software Sensível ao Contexto (adaptado de Shilit et al. 1994).....	25
Tabela 3.1: Categorias de Aplicações Web (GINIGE e MURUGESAN, 2001).....	33

LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
DiSEN	<i>Distributed Software Engineering Environment</i>
DiSEN-CSE	<i>DiSEN-Context Sensitive Environment</i>
FTP	<i>File Transfer Protocol</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
SGML	<i>Standard Generalized Markup Language</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
SOA	<i>Service-Oriented Architecture</i>
SOAP	<i>Simple Object Access Protocol</i>
UDDI	<i>Universal Description, Discovery and Integration</i>
UML	<i>Unified Modeling Language</i>
URL	<i>Uniform Resource Locator</i>
W3C	<i>World Wide Web Consortium</i>
WSDL	<i>Web Services Definition Language</i>
WWW	<i>World Wide Web</i>
XML	<i>Extensible Markup Language</i>

SUMÁRIO

1. Introdução.....	16
1.1. Objetivos.....	18
1.2. Limitações.....	18
1.3. Metodologia.....	18
1.4. Organização da Monografia.....	19
2. Computação Sensível ao Contexto.....	20
2.1. Definição de Contexto.....	20
2.2. Categorias de Contexto.....	21
2.3. Percepção de Contexto.....	23
2.4. Características de Aplicações Sensíveis ao Contexto.....	24
2.5. Aquisição de Contexto.....	26
2.6. Representação de Contexto.....	26
2.7. Requisitos para Desenvolvimento de Aplicações Sensíveis ao Contexto. .	28
2.7.1. Especificação de contexto.....	29
2.7.2. Separação de aquisição e utilização de contexto.....	29
2.7.3. Interpretação de contexto.....	29
2.7.4. Comunicação distribuída e transparente.....	30
2.7.5. Disponibilidade contínua de aquisição de contexto.....	30
2.7.6. Armazenamento de contexto.....	30
2.7.7. Descoberta de recursos.....	30
2.8. Considerações Finais.....	31
3. Aplicações Web.....	32
3.1. Definição.....	32
3.2. Classificação.....	33
3.3. Arquitetura de Aplicações Web.....	34
3.4. Características.....	35
3.5. Engenharia Web.....	36
3.6. Processo de Desenvolvimento.....	38
3.7. Considerações Finais.....	39
4. Arquitetura Orientada a Serviços.....	41
4.1. Definição.....	41

4.2. Princípios da Arquitetura Orientada a Serviços.....	42
4.3. Web Services.....	43
4.4. Arquitetura de Web Services.....	43
4.5. Protocolos e Tecnologias de Web Services.....	46
4.5.1. XML - Extensible Markup Language.....	46
4.5.2. SOAP - Simple Object Access Protocol.....	47
4.5.3. WSDL - Web Service Description Language.....	50
4.5.4. UDDI - Universal Description, Discovery and Integration.....	52
4.6. Considerações Finais.....	53
5. Elementos da Arquitetura Orientada a Serviços em Aplicações Web Sensíveis ao Contexto.....	54
5.1. Visão Geral da Arquitetura.....	54
5.2. Suporte à Captura.....	55
5.3. Representação de Contexto.....	56
5.4. Repositório.....	56
5.5. Suporte ao Processamento.....	57
5.6. Mecanismos de Apresentação.....	59
5.7. A Arquitetura.....	59
5.8. Considerações Finais.....	61
6. Exemplo de Aplicação.....	62
6.1. Cenário de Exemplo.....	62
6.2. Modelagem dos Casos de Uso.....	64
6.3. Arquitetura do Protótipo.....	64
6.4. Implementação.....	65
6.5. Resultados.....	68
6.6. Considerações Finais.....	68
7. Conclusão.....	69
7.1. Trabalhos Futuros.....	69
Referências.....	71
Apêndice.....	74
Apêndice A – Modelagem.....	75

1. INTRODUÇÃO

A crescente adoção da Internet e dispositivos móveis está movendo o paradigma de aplicações desktop tradicionais para o paradigma de computação móvel, aumentando o espaço de trabalho e as mudanças dinâmicas do seu ambiente. Este novo paradigma trouxe a possibilidade de explorar as informações de contexto no qual o usuário está inserido, surgindo assim, uma nova geração de sistemas conhecidos por aplicações sensíveis ao contexto (*context-aware applications*).

Na computação sensível ao contexto as aplicações percebem e usam informações contextuais para assim, fornecer serviços ou outras informações relevantes, proporcionando maior comodidade e tarefas mais simples ao usuário. Desta forma, essas aplicações podem explorar a mobilidade e a “onipresença” da Internet para obter informações contextuais do usuário e do próprio ambiente.

As aplicações sensíveis a contexto são capazes de adaptar o seu comportamento autonomamente em resposta às alterações de contexto (COSTA, 2003). Desta forma, o sistema se adapta de acordo com o seu local de utilização e objetos próximos ao longo do tempo. Assim, esse tipo de sistema possui a capacidade de alterar e ajustar dinamicamente o seu comportamento com base em informações contextuais.

O desenvolvimento de aplicações sensíveis ao contexto envolve vários desafios relacionados à captura, representação, armazenamento, processamento e apresentação da informação contextual. Esses elementos justificam a necessidade de uma arquitetura adequada para tal, e assim, apoiar o desenvolvimento dessas aplicações usando serviços para a aquisição e apresentação de informações contextuais.

A evolução da Internet, que surgiu como uma plataforma para compartilhamento de informações, tem proporcionado que as aplicações *Web* tornem-se sistemas interativos que abrangem um vasto conjunto de serviços. Sistemas e aplicações baseadas na *Web* (*WebApps*) evoluíram para ferramentas computacionais sofisticadas, produzindo uma complexa matriz de conteúdo e funcionalidade para uma ampla população de usuários finais (PRESSMAN, 2006).

Diante dos avanços na mobilidade e tecnologias de rede, as aplicações *Web* estão mudando a forma como sistemas interagem com as pessoas, através de diferentes tipos de dispositivos para acesso em qualquer lugar. Os usuários estão procurando por serviços e aplicações com conteúdo customizáveis, de acordo com as suas preferências e o seu ambiente (CERI *et al.*, 2007). Desta forma, aplicações *Web* adaptáveis podem oferecer conteúdo relevante, selecionando informações disponíveis de acordo com o contexto do próprio usuário.

O desenvolvimento de aplicações *Web* incorpora processos, métodos e tecnologias, além da arquitetura de software, que engloba um conjunto de decisões importantes sobre a organização de um sistema de software (BOOCH, 2001). Sendo assim, a estrutura do sistema e a forma como os componentes de dados e procedimentos colaboram entre si, definem a arquitetura de software (PRESSMAN, 2006).

A Arquitetura Orientada a Serviços, do acrônimo em inglês SOA (*Service-Oriented Architecture*), é uma arquitetura de software que promove funcionalidades da aplicação disponibilizadas na forma de serviços (ERL, 2005). Esses serviços podem ser implementados por meio de um conjunto de especificações e protocolos chamados *Web services*, que definem interfaces e contratos para a comunicação entre aplicações.

Como visto anteriormente, aplicações sensíveis ao contexto possuem a capacidade de adaptar o comportamento e a interface da aplicação de acordo com a situação de contexto do usuário e do seu ambiente, abrangendo uma vasta gama de parâmetros de adaptação (CHAARI *et al.*, 2004). Além do mais, essas informações podem ser combináveis e assim estender esse tipo de aplicação (YOON, 2007). Desta forma, Chaari *et al.* (2004) afirmam que, o desenvolvimento de aplicações sensíveis ao contexto requer a concepção de uma arquitetura que apoie percepção de contexto em tempo de execução, no qual os componentes dedicados à gestão de contexto são separados do núcleo da aplicação.

Como a computação sensível ao contexto é um campo de investigação recente no domínio da Internet (CERI *et al.*, 2007), novas questões e necessidades têm sido abordadas para oferecer apoio à percepção de contexto nesses sistemas e, conseqüentemente lança desafios para os desenvolvedores garantirem adaptação e

extensão em aplicações sensíveis ao contexto. Desta forma, informações de contexto provenientes de diversos sistemas heterogêneos podem ser obtidas de forma transparente ao usuário por meio de serviços. A arquitetura orientada a serviços possui características como abstração, fraco acoplamento, contrato formal e autonomia, que proporcionam a evolução das aplicações por meio da integração de sistemas.

1.1. Objetivos

O objetivo geral deste trabalho é investigar a utilização da Arquitetura Orientada a Serviços em aplicações *Web* sensíveis ao contexto. A partir do objetivo geral foram definidos os seguintes objetivos específicos:

- apresentar e descrever o paradigma de computação sensível ao contexto;
- aplicar a integração da arquitetura orientada a serviços em um modelo *context-awareness*;
- demonstrar como a integração de *Web services* em aplicações *Web* sensíveis ao contexto promovem o apoio para adaptação e extensão; e
- exemplificar o uso da solução proposta com o desenvolvimento de um estudo de caso.

1.2. Limitações

Não faz parte do escopo deste trabalho:

- investigar outras arquiteturas e/ou tecnologias que atendam e forneçam suporte para o desenvolvimento de aplicações *Web* sensíveis ao contexto;
- apontar os conceitos relativos à computação ubíqua; e
- descrever aspectos de segurança em aplicações *Web* sensíveis ao contexto.

1.3. Metodologia

Para atender aos objetivos desse trabalho, a seguinte metodologia foi utilizada. Durante a atividade de Revisão, foram realizados estudos sobre

Computação Sensível ao Contexto, Aplicações *Web* e Arquitetura Orientada a Serviços. Os resultados dessa atividade foram discutidos nos Capítulos 2, 3 e 4. Também foi estudada uma arquitetura para o suporte ao gerenciamento de contexto, discutida no capítulo 5. A atividade de Implementação, discutida no capítulo 6, consistiu em analisar, projetar e implementar um protótipo para aplicar os conceitos estudados. Por fim, a atividade de Redação consistiu em produzir esta monografia.

1.4. Organização da Monografia

Este capítulo apresentou a introdução e os objetivos desta monografia. O restante deste trabalho está organizado da seguinte maneira:

- Capítulo 2: apresenta os conceitos relacionados à computação sensível ao contexto;
- Capítulo 3: apresenta os conceitos relacionados às aplicações *Web*;
- Capítulo 4: apresenta os conceitos relacionados à arquitetura orientada a serviços;
- Capítulo 5: descreve os elementos da arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto;
- Capítulo 6: detalha a proposta desta monografia por meio de um exemplo de aplicação;
- Capítulo 7: discute os resultados obtidos no desenvolvimento desta monografia, além das contribuições e sugestões de trabalhos futuros.

2. COMPUTAÇÃO SENSÍVEL AO CONTEXTO

Este capítulo aborda alguns conceitos pertinentes à computação sensível ao contexto. Primeiramente, são apresentadas algumas definições e categorias de contexto. Em seguida, o termo percepção de contexto (*context-aware*), é apresentado por meio de algumas definições. A seguir, são expostas as características de aplicações sensíveis ao contexto, além das formas de aquisição e representação de contexto. Por fim, são apresentadas as considerações finais deste capítulo.

2.1. Definição de Contexto

O termo contexto é definido por vários autores na literatura (SCHILIT e THEIMER, 1994; BROWN et al., 1997; RYAN et al., 1997; DEY, 1998; BRÉZILLON e POMEROL, 1999). O primeiro trabalho a definir contexto foi dado por Schilit e Theimer (1994), os quais referem-se a contexto como informações de localização, de identidade de pessoas e objetos próximos entre si e das mudanças nesses objetos. De forma similar, Brown et al. (1997) definem contexto como informações de localização, das identidades de pessoas que cercam um usuário, a hora do dia, a estação do ano e a temperatura de um ambiente físico. Ryan et al. (1997) definem contexto como localização do usuário, ambiente, identidade e tempo. Dey (1998) enumera contexto como estado emocional do usuário, foco de atenção, localização e orientação, data e hora, objetos e pessoas no ambiente do usuário. Brézillon e Pomerol (1999) definem contexto como todo o conhecimento usado para resolver um problema sem intervir nele explicitamente.

Estas definições são específicas e difíceis de aplicar, pois são baseadas em exemplos e sinônimos, referenciando contexto como ambiente ou situação (DEY e ABOWD, 2000). Além disso, alguns autores consideram contexto no ambiente do usuário, enquanto outros consideram no ambiente da aplicação. Brown (1996) define contexto como, uma combinação de elementos do ambiente que o computador do usuário conhece. Ward et al. (1997) visualizam contexto como o estado do ambiente da aplicação. Dey et al. (1998) definem contexto como estado físico, social, emocional ou informacional do usuário. Pascoe (1998) define contexto como um

subconjunto de estados físicos e conceituais de interesse de uma entidade particular.

No entanto, a definição apresentada por Dey (2001) é mais genérica e tem sido bem aceita pela comunidade acadêmica. Segundo Dey (2001), contexto é "qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação". Sendo assim, se uma informação pode ser usada para caracterizar a situação de um participante em uma interação, então esta informação é contexto (DEY e ABOWD, 2000).

Apesar destas diversas definições serem caracterizadas dentro de uma área de domínio específico, Vieira (2008) aponta alguns aspectos das definições que os pesquisadores concordam: (i) contexto existe somente quando relacionado a uma entidade (tarefa, interação ou agente, que pode ser humano ou de software); (ii) contexto é um conjunto de itens associados a uma entidade; e (iii) um item é considerado parte do contexto somente se for útil para apoiar a resolução de um problema.

Com base nessas observações, Vieira (2008) distinguiu os termos contexto e elemento contextual, definindo elemento contextual como "qualquer item de dado ou informação que caracteriza uma entidade em um domínio" e que "o contexto de uma interação entre um agente e uma aplicação, com o objetivo de realizar alguma tarefa, é um conjunto de elementos contextuais instanciados que são necessários para apoiar a execução da tarefa". Sendo assim, e com base em Chaves (2009), essas definições foram utilizadas no desenvolvimento deste trabalho por representar claramente a dinamicidade do contexto, relacionando-o com os elementos relevantes para uma determinada interação, neste caso, entre os serviços da aplicação.

2.2. Categorias de Contexto

A categorização de tipos de contexto pode ajudar os desenvolvedores a descobrir quais informações de contexto serão úteis para sua aplicação. Ryan et al. (1997), sugerem tipos de contexto de localização, ambiente, identidade e tempo. Já

Schilit et al. (1994), definiram que os aspectos importantes de contexto são: “onde você está, quem está com você e, quais recursos estão próximos”. Dessa forma, eles classificaram contexto em três categorias, conforme mostra a Tabela 2.1.

Tabela 2.1: Categorias de Contexto, conforme Schilit et al. (1994)

Contexto do Usuário	Perfil do usuário
	Localização do usuário
	Situação social
Contexto Computacional	Conectividade da rede
	Custo da comunicação
	Largura de banda da rede
Contexto Físico	Luminosidade
	Nível de ruído

Há certos tipos de informações de contexto que, na prática, são mais importantes que outras. Para saber se uma informação é importante para ser considerada contexto, pode-se utilizar as diretrizes "quem", "onde", "o que" e "quando" relacionadas a uma entidade para determinar "porquê" uma situação está ocorrendo (DEY e ABOWD, 2000).

De acordo com Dey e Abowd (2000), há tipos de informações que são chamadas de contexto primário e, são classificadas em: localização, identidade, atividade e tempo. A diferença para a lista de Ryan et al. (1997) é o uso de “atividade” no lugar de “ambiente”. Já as categorias definidas por Schilit et al. (1994), incluem somente informações de localização e identidade. Sendo assim, para caracterizar uma situação, também são necessárias as informações de atividade e tempo.

Dey e Abowd (2001) afirmam que, os tipos de contexto primário servem como índice para outras fontes de informação de contexto. Por exemplo, dada a identidade de uma pessoa, pode-se adquirir outras informações relacionadas, tais como, número de telefone, endereço e *e-mail*. Estas informações de contexto, encontradas através do contexto primário, são chamadas de contexto secundário.

2.3. Percepção de Contexto

Muitos sistemas computacionais tradicionais não exploram informações contextuais do usuário e processam sua função baseada somente em entrada explícita (COSTA, 2003), conforme apresenta a Figura 2.1. Estes sistemas não possuem a capacidade de obter informações dinamicamente do ambiente para oferecer serviços personalizados ao usuário.

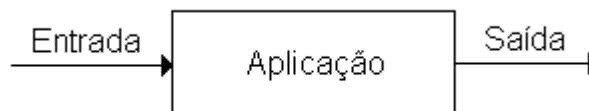


Figura 2.1: Aplicações Tradicionais

Em computação sensível ao contexto, há uma nova geração de aplicações que obtém informações contextuais para oferecer serviços e interação ao usuário. Essas aplicações possuem a capacidade de percepção de contexto e consideram a informação contextual como entrada implícita (COSTA, 2003), como ilustra a Figura 2.2.

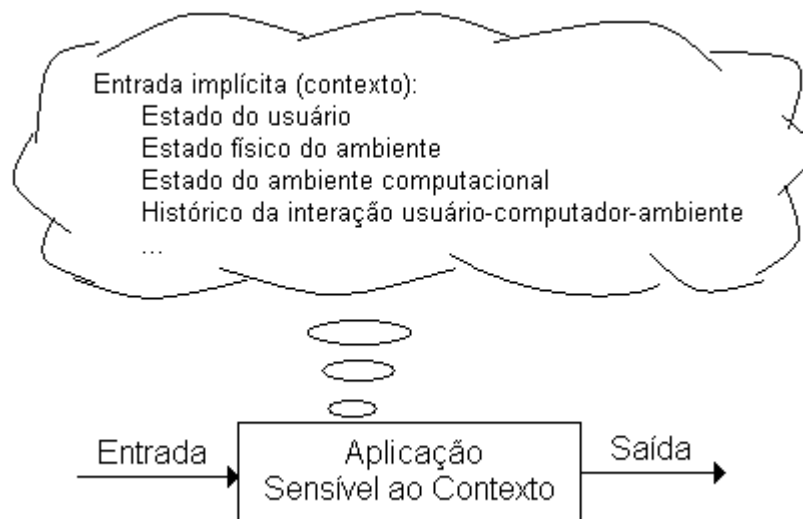


Figura 2.2: Aplicações Sensíveis ao Contexto

O termo percepção de contexto, do inglês *context-aware*, é definido por vários autores (SCHILIT et al., 1994; WARD et al., 1997; BROWN et al., 1997; RYAN et al., 1997; PASCOE, 1998; DEY, 1998; DEY et al., 1998; SALBER et al., 1998), sendo umas das primeiras definições introduzida por Schilit e Theimer (1994). Eles definiram percepção de contexto como, “software que se adapta de acordo com a

sua localização de uso, o conjunto de pessoas e objetos próximos, assim como, as mudanças ocorridas com esses objetos ao longo do tempo”. No entanto, Dey e Abowd (2000) argumentam que essa definição é restrita às aplicações, pois requerem que o seu comportamento seja modificado para a mesma ser considerada sensível ao contexto.

Schilit et al. (1994), Ward et al. (1997) e Brown et al. (1997) definem que, com percepção de contexto as aplicações mudam ou adaptam dinamicamente o seu comportamento baseado no contexto da aplicação e do usuário. Ryan et al. (1997) e Pascoe (1998) definem percepção de contexto como, a habilidade de dispositivos computacionais detectarem e sentirem, interpretar e responderem aos aspectos do ambiente local do usuário e dos próprios dispositivos computacionais. Dey (1998) argumenta que na percepção de contexto, a interação humano-computador impulsiona o conhecimento do contexto do usuário. Dey et al. (1998) partem da noção de adaptação para definir percepção de contexto, baseado no conhecimento do contexto do usuário. Salber et al. (1998) definem percepção de contexto como sendo a habilidade de fornecer flexibilidade máxima de um serviço computacional em tempo real.

Sendo assim, Dey e Abowd (2000) propõem uma definição mais genérica e centrada no usuário, a qual diz que, “um sistema é sensível ao contexto se ele usa informações de contexto para fornecer informações e/ou serviços relevantes para o usuário, no qual a relevância depende das tarefas do usuário”. Desse modo, essa definição foi utilizada no desenvolvimento deste trabalho por não excluir as aplicações sensíveis ao contexto que estão de acordo com as definições prévias propostas, e também, por estar relacionada ao usuário, não limitar a percepção apenas à interface do usuário, não requer que as aplicações realizem serviços automaticamente e nem adquiram contexto em tempo real.

2.4. Características de Aplicações Sensíveis ao Contexto

Para Schilit et al. (1994), há quatro características em aplicações sensíveis ao contexto, resultantes de duas dimensões ortogonais, conforme apresenta a Tabela 2.2: se a tarefa está obtendo informações ou executando um comando e se, a tarefa é realizada manualmente ou automaticamente.

Tabela 2.2: Dimensões de Software Sensível ao Contexto (adaptado de Shilit et al. 1994)

	Manual	Automático
Informação	Seleção próxima e informação contextual	Reconfiguração contextual automática
Comando	Comandos contextuais	Ações de contexto-desencadeado

De acordo com Shilit et al. (1994), em “seleção próxima”, a aplicação recupera informação contextual para o usuário manualmente, no qual uma lista de objetos e lugares é apresentada e, os itens relevantes para o contexto do usuário são destacados e podem ser escolhidos. Em “reconfiguração contextual automática”, a aplicação recupera informação contextual para o usuário automaticamente, criando uma ligação entre os recursos disponíveis através da adição, remoção e alterações de conexões entre componentes. Já em “comandos contextuais”, a aplicação executa comandos para o usuário manualmente, sendo que a execução é modificada de acordo com o contexto do usuário. Finalmente, em “ações de contexto-desencadeado”, a aplicação executa comandos para o usuário automaticamente, realizando serviços combinados do contexto existente e baseadas em regras “se-então”.

Pascoe (1998) apresenta um conjunto de características para identificar e descrever aplicações sensíveis ao contexto, com foco nas características centrais da computação sensível ao contexto. De acordo com Pascoe (1998), a primeira característica é “percepção contextual”, que é a habilidade de detectar informação contextual e apresentá-la ao usuário sob uma forma conveniente. A próxima característica é “adaptação contextual”, que é a habilidade de executar ou modificar um serviço automaticamente baseado no contexto atual. A terceira característica é “descoberta de recursos contextuais”, que permite que aplicações sensíveis ao contexto localizem e explorem recursos e serviços que são relevantes para o contexto do usuário. A última característica é “expansão contextual”, que é a habilidade de associar dados digitais com o contexto do usuário.

Baseados na combinação das ideias apresentadas por Shilit et al. (1994) e Pascoe (1998), Dey e Abowd (2000) propõem três características que aplicações sensíveis ao contexto podem implementar:

- apresentação de informações e serviços para o usuário;
- execução automática de um serviço; e
- união da informação de contexto para recuperação posterior.

A primeira característica, “apresentação”, refere-se às aplicações que, ou apresentam informação de contexto para o usuário ou, usam o contexto para sugerir seleções apropriadas de ações para o usuário. A segunda característica, “execução automática”, descreve aplicações que executam um comando ou reconfigura o sistema em favor do usuário, de acordo com as mudanças do contexto. Na terceira característica, “união”, a aplicação marca dados às informações de contexto relevantes (DEY et al., 2001).

2.5. Aquisição de Contexto

Mostéfaoui et al. (2004) afirmam que, dependendo do campo da aplicação, há vários caminhos para se obter contexto. Eles definiram três formas para a aquisição de contexto, de acordo com a forma como ela é obtida:

Sentida: a informação (e.g. temperatura, pressão, luminosidade e nível de ruído) é adquirida por meio de sensores físicos;

Derivada: a informação contextual (e.g. tempo e data) pode ser obtida em tempo de execução, por meio de regras de inferência;

Provida: a informação (e.g. preferências do usuário) é fornecida explicitamente à aplicação pelo próprio usuário.

Entretanto, a aquisição de contexto não é uma tarefa fácil, principalmente quando a informação é sentida (MOSTÉFAOUI et al., 2004). Isso ocorre devido à grande variedade de tipos sensores e a natureza dinâmica da informação contextual, sendo necessário que a aplicação gerencie esses aspectos.

2.6. Representação de Contexto

A representação de contexto é o mapeamento das informações por meio de um modelo formal e deve ser aplicada em todo o processo de coleta, transferência, armazenamento e interpretação de informações de contexto. De acordo com Held et

al. (2002), a representação de contexto precisa contemplar algumas características, de forma que essa informação possa ser:

Estruturada: fornece meios para filtrar ou extrair eficientemente a informação do contexto que é relevante para a aplicação. Além disso, reduz a possibilidade de ambiguidade de atributos;

Intercambiável: muitas vezes, informação contextual precisa ser trocada entre as aplicações ou entre os diferentes componentes da própria aplicação;

Composta/Decomposta: permite que a informação do contexto seja armazenada e mantida de forma distribuída. Por exemplo, no caso de uma atualização, pode ser enviada apenas aquela parte da informação que foi modificada, evitando que seja enviada novamente toda a informação do contexto de diferentes fontes;

Uniforme: uma representação uniforme da informação de contexto facilita a interpretação durante o processo de adaptação de conteúdo;

Extensível: a representação da informação deve permitir que sejam adicionados novos parâmetros, pois não há um conjunto de atributos que seja identificado hoje e sirva para todas as futuras aplicações;

Padronizada: como a informação provém de diferentes entidades, é fundamental que a informação seja representada de forma padronizada.

Sendo assim, como a representação de contexto influencia o projeto de sistemas sensíveis ao contexto, há na literatura diferentes abordagens para uma melhor representação de contexto, cada qual com suas características particulares. De acordo com Strang e Linnhoff-Popien (2004), os principais modelos são:

Modelo baseado em pares de chave-valor: é a mais simples estrutura de dados para representar informação contextual, em que os serviços são descritos com uma lista de atributos “chave” e “valor”;

Modelo baseado em marcação: é uma estrutura de dados hierárquicas que consiste de *tags* de marcação com atributos e conteúdo, com base no padrão XML (*Extensible Markup Language*);

Modelo baseado em representações gráficas: a estrutura genérica da UML (*Unified Modeling Language*) a torna apropriada para modelagem de contexto, pois aspectos contextuais relevantes são modelados como extensões UML;

Modelo baseado em orientação a objetos: este modelo emprega os principais benefícios de uma abordagem orientada a objeto, tais como, encapsulamento e reutilização, para que os detalhes do processamento de contexto sejam encapsulados em nível de objeto e fornecidos por meio de interfaces especificadas;

Modelo baseado em lógica: com alto nível de formalismo, esse modelo define as condições em que uma expressão ou fato pode ser derivado, por um processo de raciocínio ou inferência, de um conjunto de outras expressões ou fatos;

Modelo baseado em ontologia: com formalismo, esse modelo especifica conceitos e relacionamentos, normalizando e combinando o conhecimento de diferentes domínios por meio de linguagens de ontologia.

Neste trabalho foi adotado o modelo baseado em marcação, com base no padrão XML, pois o mesmo facilita a comunicação por diferentes aplicações. Além disso, Held et al. (2002) afirmam que esse modelo é extensível, flexível e aberto, usado em aplicações sensíveis ao contexto, principalmente para expressar perfis de usuários.

2.7. Requisitos para Desenvolvimento de Aplicações Sensíveis ao Contexto

Para o desenvolvimento de aplicações sensíveis ao contexto, há um conjunto de requisitos que auxiliam durante este processo. Dey (2000) apresenta esse conjunto de características que podem ser utilizadas no desenvolvimento de sistemas sensíveis ao contexto, a saber: especificação de contexto, separação de aquisição e utilização de contexto, interpretação de contexto, comunicação distribuída e transparente, disponibilidade contínua de aquisição de contexto, armazenamento de contexto e descoberta de recursos.

2.7.1. Especificação de contexto

Um requisito importante para o processo de desenvolvimento é um mecanismo que permita especificar quais contextos a aplicação necessita. O mecanismo e a linguagem de especificação devem permitir que os desenvolvedores indiquem as informações de contexto como simples ou múltiplos fragmentos, se há relacionamento entre essas informações e se estas são interpretadas ou não (DEY, 2000).

2.7.2. Separação de aquisição e utilização de contexto

Não há uma forma padrão para adquirir e manipular contexto. Existem duas maneiras comuns para tratar contexto: (i) associar *drivers* de sensores diretamente nas aplicações e (ii) utilizar servidores que ocultam os detalhes dos sensores. Após serem adquiridas as informações de contexto, por meio de mecanismos de aquisição ou notificação, a aplicação verifica a ocorrência e a relevância das modificações nessas informações. Ao separar os processos de aquisição e utilização de contexto, a aplicação utiliza essas informações sem se preocupar com os detalhes de como foram adquiridas (DEY, 2000).

2.7.3. Interpretação de contexto

Existe a necessidade de estender os mecanismos de notificação e de consulta para permitir que aplicações recuperem contexto em ambientes distribuídos. As informações de contexto podem passar por várias camadas de software antes de chegar à aplicação devido à necessidade de abstração (interpretação). Por exemplo, uma aplicação que precisa ser notificada quando uma reunião estiver para acontecer: em um nível mais baixo, informações de localização e identificação podem ser interpretadas para determinar onde várias pessoas se encontram. Em um nível mais alto, essa informação pode ser combinada com os registros da agenda dessas pessoas para determinar se a reunião está acontecendo. Para o desenvolvedor da aplicação, o uso de várias camadas deve ser transparente e dessa forma, o contexto deve ser interpretado antes de ser usado pela aplicação (DEY, 2000).

2.7.4. Comunicação distribuída e transparente

Os dispositivos utilizados para capturar contexto, em sua maioria, não estão alocados no mesmo computador em que as aplicações sensíveis ao contexto estão sendo executadas. Os sensores podem estar fisicamente distribuídos e assim, não estarem diretamente ligados a uma única máquina. Além disso, as aplicações podem utilizar informações que estão localizadas em vários dispositivos. Devido a isso, a comunicação distribuída deve ser transparente tanto para os sensores quanto para as aplicações, e assim, os desenvolvedores não precisam implementar um protocolo de comunicação e um esquema de codificação e decodificação para a transmissão de informações de contexto (DEY, 2000).

2.7.5. Disponibilidade contínua de aquisição de contexto

Os componentes que capturam informações de contexto devem ser executados de maneira independente das aplicações que os usam e também, estarem sempre disponíveis. Como não se sabe quando as aplicações solicitarão informações de contexto, esses componentes devem executar continuamente para permitir que as aplicações os consultem quando necessário (DEY, 2000).

2.7.6. Armazenamento de contexto

Devido à demanda pela constante disponibilidade dos dados, é necessário manter históricos de informações de contexto. Os componentes de aquisição de informações de contexto devem manter um histórico de todo o contexto que obtém, pois ele pode ser usado para estabelecer tendências e prever valores futuros de contexto. Sem o armazenamento de contexto, esse tipo de análise não poderia ser realizada. Componentes de captura devem adquirir informações de contexto mesmo quando nenhuma aplicação está interessada nessas informações naquele momento, pois elas podem ser utilizadas no futuro por uma aplicação que necessite do histórico de um contexto, por exemplo, o histórico de localização de um usuário para prever sua localização futura (DEY, 2000).

2.7.7. Descoberta de recursos

Para que uma aplicação possa se comunicar com dispositivos de captura de contexto, ela deve saber que tipo de informação o sensor pode fornecer, qual a sua

localização e quais os seus protocolos e mecanismos para se comunicar. Assim que uma aplicação é iniciada, são informados os tipos de informações de contexto de seu interesse. Com isso, o mecanismo de descoberta de recursos indica onde encontrar os componentes adequados e descreve os mecanismos de acesso para eles. O mecanismo de descoberta de recursos pode ser também utilizado pela aplicação para determinar se a informação de contexto requisitada está disponível no ambiente (DEY, 2000).

2.8. Considerações Finais

Neste capítulo foram apresentados os principais conceitos pertinentes à computação sensível ao contexto. Verificou-se que há diversas definições na literatura para o termo contexto. Assim como para categorias de contexto, classificações de diferentes autores foram abordadas. Também foram apresentadas algumas definições para percepção de contexto e, suas principais características. Em sequência, foram apontadas algumas formas para a aquisição e representação de contexto. Por fim, foram apresentados os requisitos para o desenvolvimento de aplicações sensíveis ao contexto. O capítulo seguinte apresenta alguns conceitos relacionados às aplicações *Web*.

3. APLICAÇÕES WEB

Este capítulo aborda alguns conceitos pertinentes às aplicações *Web*. Primeiramente, são apresentadas algumas definições e a classificação de aplicações *Web*. Em seguida, a arquitetura e as características de aplicações *Web* são apresentadas. A seguir, são abordados aspectos referentes à engenharia *Web* e processo de desenvolvimento. Por fim, são apresentadas as considerações finais deste capítulo.

3.1. Definição

Originalmente a *Web* foi concebida com o propósito de compartilhar informações científicas entre poucos pesquisadores (GINIGE e MURUGESAN, 2001). De simples sites *Web*, que consistiam em textos interligados por elos (*hyperlinks*), a *Web* teve seu escopo e proposta de uso ampliados, apoiados pelo constante avanço da Internet, padrões e tecnologias *Web* (MURUGESAN e GINIGE, 2005). Empreendimentos, turismo, indústrias hospitalares, bancos, instituições educacionais, entretenimento, negócios e até governos usam aplicações e sistemas baseados na *Web* para aprimorar ou estender suas operações (MURUGESAN e GINIGE, 2005). Muitos sistemas de informações legados têm sido, progressivamente, migrados para a *Web* (MURUGESAN e GINIGE, 2005), produzindo uma complexa combinação de conteúdos e funcionalidades para uma ampla população de usuários (PRESSMAN, 2006).

Holck (2003) afirma que há na literatura discussões sobre as peculiaridades do desenvolvimento *Web*, que geram um número de nomes diferentes para o mesmo fenômeno, tais como: Sistema de Informação Baseado na *Web*, Sistema de Informação *Web*, Sistema Baseado na *Web*, Aplicação Baseada na *Web*, Aplicação *Web*, Aplicação de Software *Web*, Solução *Web*, *Site Web* e Aplicação *Web* Interativa.

Gellersen e Gaedke (1999) definem uma aplicação *Web* como "alguma aplicação de software que depende da *Web* para a sua correta execução". Gnaho (2001) define um sistema de informação *Web* como "um sistema de informação que proporciona facilidades para acessar dados complexos e serviços interativos pela *Web*". Scharl et al. (2001) afirmam que sistemas de informação *Web* "representam

uma subcategoria de sistemas de informação em massa que, tipicamente, apoia a recuperação de informação *on-line* e tarefas rotineiras pelo caminho do próprio serviço para um grande número de usuários ocasionais que estão espalhados em vários locais”. Holck (2003) define um sistema de informação *Web* como “um sistema de informação apoiado por computador, utilizando a tecnologia da WWW (*World Wide Web*), e acessado pela maioria de seus usuários por meio de um *browser*”.

Sendo assim, por ser recente e mais objetiva, neste trabalho foi adotada a definição de Conallen (2002), que afirma que uma aplicação *Web* é “um sistema de software cujas funcionalidades são executadas através da *Web*”.

3.2. Classificação

Atualmente, na literatura, não existe um consenso sobre uma classificação padrão para as aplicações *Web*. Há classificações abrangentes e simples, como a de Pressman (2006), e até classificações mais detalhadas, como a de Ginige e Murugesan (2001).

Na classificação proposta por Pressman (2006), uma aplicação *Web* compreende todo tipo de aplicação existente para a *Web*, desde um simples *site Web* até um portal de comércio eletrônico com intenso processamento de informações. Outros pesquisadores sugerem classificações baseadas nos detalhes técnicos da aplicação ou em suas funcionalidades. Conallen (2002) classificou as aplicações *Web* em: *sites Web*, os quais compreendem sistemas hipermídia distribuídos e interconectados por *links*, tendo apenas o intuito informativo; e, sistemas *Web*, como aqueles que adicionam a lógica de negócio à aplicação. Ginige e Murugesan (2001) apresentam uma classificação levando em consideração o domínio da aplicação, conforme apresenta a Tabela 3.1.

Tabela 3.1: Categorias de Aplicações *Web* (GINIGE e MURUGESAN, 2001)

Categorias	Exemplos
Informacional	Jornais <i>on-line</i> , catálogos de produto, manuais de serviço, classificados <i>on-line</i> , livros eletrônicos
Interativo	Formulários de registro, apresentação de informação customizada, jogos <i>on-line</i>

Transacional	Compras eletrônicas, serviços de bancos
<i>Workflow</i>	Sistemas de planejamento, gerenciamento de inventário, monitoria de <i>status</i>
Ambiente de trabalho colaborativo	Sistemas de autorização distribuídos, ferramentas de projeto colaborativo
Comunidades <i>on-line</i>	<i>Chat</i> , sistemas que recomendam produtos ou serviços
Portais <i>Web</i>	Portais de comércio eletrônico

As sete categorias de aplicações *Web* da Tabela 3.1, mostram que “uma determinada aplicação pode pertencer a mais de uma categoria” (GIGINE e MURUGESAN, 2001). Por exemplo, um *site* de comércio eletrônico é ao mesmo tempo uma aplicação “transacional” e “portal *Web*”.

3.3. Arquitetura de Aplicações *Web*

Para o sucesso do projeto de uma aplicação *Web*, é fundamental a presença de uma arquitetura (BOOCH, 2001). De um modo geral, a arquitetura básica de uma aplicação *Web* inclui três componentes: o navegador cliente, uma conexão de rede e um servidor *Web* (CONALLEN, 1999). O servidor *Web* é um software executado em um computador remoto que responde às solicitações de outro software, o navegador cliente, por meio do protocolo de comunicação HTTP (*Hypertext Transfer Protocol*).

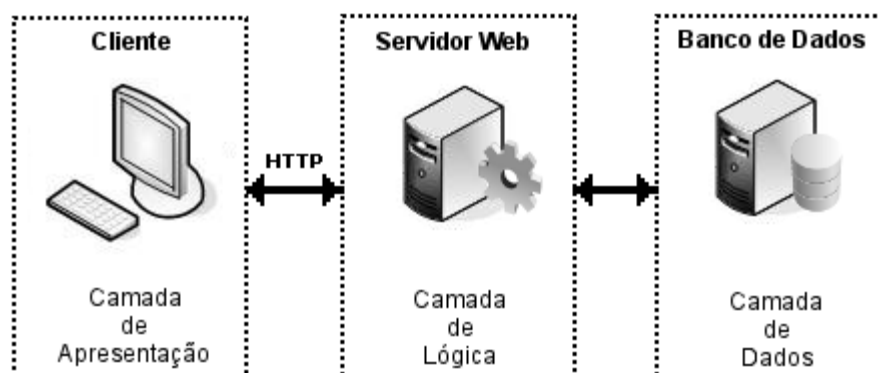


Figura 3.1: Arquitetura básica de Aplicações *Web*

Conforme apresenta a Figura 3.1, os componentes podem ser estruturados em uma arquitetura composta por três camadas: apresentação, lógica e dados. Por meio de um servidor *Web* pode-se ter acesso às funções de lógica de negócio e de

acesso aos dados, enquanto o cliente possui funções de interface com o usuário (CONALLEN, 2002).

Desta forma, Conallen (2002) afirma que a arquitetura da aplicação *Web* influencia todos os passos no processo de desenvolvimento, pois fornece regras e limites para a construção do software e, de certo modo, como o problema deve ser solucionado.

3.4. Características

As aplicações *Web* possuem certas características que as distinguem do software tradicional. Murugesan e Ginige (2005) relacionam as seguintes características:

- as aplicações *Web* evoluem constantemente, pois a sua estrutura e funcionalidades, além das informações apresentadas pelo *site*, modificam ao longo do tempo, especialmente após o sistema ser colocado em uso;
- o conteúdo em aplicações *Web* pode incluir texto, gráficos, imagens, áudio e/ou vídeo, sendo que, a forma como esse conteúdo é apresentado e organizado, tem implicações sobre o desempenho e tempo de resposta do sistema;
- as aplicações *Web* podem ser utilizadas por uma vasta comunidade de usuários, com diferentes exigências, expectativas e conjunto de habilidades. Portanto, a interface do usuário e as características de usabilidade têm que satisfazer as necessidades de uma diversidade de usuários anônimos;
- a maioria dos sistemas baseados na *Web* são dirigidos ao conteúdo e portanto, nesses sistemas há a criação, manutenção e gestão de conteúdo;
- os sistemas baseados na *Web* favorecem a criatividade visual e a incorporação de multimídia na interface da aplicação;
- aplicações *Web* têm um cronograma de desenvolvimento comprimido, sendo assim, não é apropriado um processo de desenvolvimento que seja demorado;

- falhas ou insatisfação dos usuários de aplicações baseadas na *Web* pode ser pior que em sistemas convencionais;
- aplicações *Web* geralmente são desenvolvidas por uma equipe de pessoas com diversas origens, habilidades e conhecimentos, além de percepção da qualidade desses sistemas;
- há constantes avanços nas tecnologias *Web*, linguagens, padrões e ferramentas;
- o desenvolvimento *Web* usa diversas tecnologias e padrões, e integra vários componentes, linguagens de *script*, arquivos HTML (*HyperText Markup Language*), bancos de dados, imagens e outros componentes multimídia como áudio e vídeo, além de complexas interfaces de usuários;
- o meio de distribuição de aplicações *Web* é completamente diferente do software tradicional, pois necessitam apoiar uma variedade de dispositivos de exibição, formatos, hardware, software e redes com velocidades variadas de acesso;
- sistemas baseados na *Web* necessitam de maior segurança e privacidade do que o software tradicional;
- a *Web* apresenta uma maior ligação entre a arte e a ciência do que geralmente é encontrado no desenvolvimento de software convencional.

Sendo assim, Murugesan e Ginige (2005) afirmam que estas características únicas da *Web* e suas aplicações, tornam o desenvolvimento desses sistemas diferentes e mais exigentes que o desenvolvimento de software tradicional.

3.5. Engenharia Web

A Engenharia *Web* é uma forma de desenvolvimento e organização do conhecimento para desenvolver aplicações *Web*, gerenciando a complexidade e a diversidade dessas aplicações (MURUGESAN e GINIGE, 2005). Um sistema baseado na *Web* precisa de uma infraestrutura, criada pela Engenharia *Web*, que permita sua evolução, minimizando os riscos e aumentando a facilidade de manutenção e a qualidade.

A Engenharia *Web* é uma disciplina emergente e apresenta um processo e uma abordagem sistemática para o desenvolvimento de sistemas baseados na *Web*. Sendo assim, Murugesan et al. (2001) afirmam que Engenharia *Web* “é o uso, com base científica, dos princípios e disciplinas de engenharia e gerenciamento, por meio de abordagens sistemáticas para o desenvolvimento, implantação e manutenção de sistemas e aplicações de alta qualidade baseadas na *Web*”.

Há várias atividades ou tarefas que tratam todos os aspectos do desenvolvimento de aplicações *Web*, a partir da concepção, desenvolvimento, implementação, avaliação de desempenho e manutenção contínua. Murugesan et al. (2001) apresentam essas principais atividades:

- especificação e análise de requisitos;
- desenvolvimento de metodologias e técnicas para as aplicações *Web*;
- integração com sistemas legados;
- migração de sistemas legados para o ambiente *Web*;
- desenvolvimento de aplicações *Web* de tempo real;
- teste, verificação e validação;
- avaliação da qualidade, controle e garantia;
- gerenciamento da configuração e projeto;
- “métricas *Web*” para estimativas dos esforços de desenvolvimento;
- especificação e avaliação de desempenho;
- atualização e manutenção;
- modelos, equipes e pessoal de desenvolvimento;
- aspectos humanos e culturais;
- desenvolvimento centrado no usuário, por meio de modelagem, participação e *feedback* do mesmo;
- desenvolvimento de aplicação ao usuário-fim; e
- educação e treinamento.

A Engenharia *Web* adota alguns princípios da Engenharia de Software, entretanto, Murugesan et al. (2001) apresenta algumas diferenças:

- os sistemas baseados na *Web* são orientados a documentos e contém páginas estáticas ou dinâmicas;
- a aparência e a incorporação de elementos multimídia são evidenciados na apresentação e interface com o usuário;
- a maioria do sistemas *Web* são dirigidos ao conteúdo;
- os sistemas baseados na *Web* necessitam atender usuários com diferentes perfis e capacidades para a interação com a interface;
- a *Web* apresenta uma ligação entre a arte e a ciência que não são encontradas no desenvolvimento de software convencional;
- a maioria do sistemas baseados na *Web* precisam ser desenvolvidos a curto prazo; e
- a *Web* é diferente do software convencional em relação ao meio de distribuição;

Sendo assim, a Engenharia *Web* é um campo multidisciplinar, pois engloba diversas áreas, tais como a interação humano-computador, interface com o usuário, análise e projeto de sistemas, engenharia de software, engenharia de requisitos, engenharia de hipermídia, estrutura de informação, teste, modelagem, simulação e gestão de projetos, bem como ciências sociais, artes e projeto gráfico (MURUGESAN et al., 2001).

3.6. Processo de Desenvolvimento

Murugesan e Ginige (2005) apresentam um processo evolutivo que descreve as diferentes etapas e atividades do desenvolvimento de aplicações *Web*. Este processo auxilia os desenvolvedores a entender o contexto no qual a aplicação será implantada e utilizada, ajuda na captura dos requisitos, permite a integração de habilidades em diferentes disciplinas, facilita a comunicação entre os diversos membros envolvidos no processo de desenvolvimento, apoia a evolução contínua e

a manutenção, facilita o gerenciamento do conteúdo da informação e auxilia no gerenciamento da complexidade e diversidade do processo de desenvolvimento.

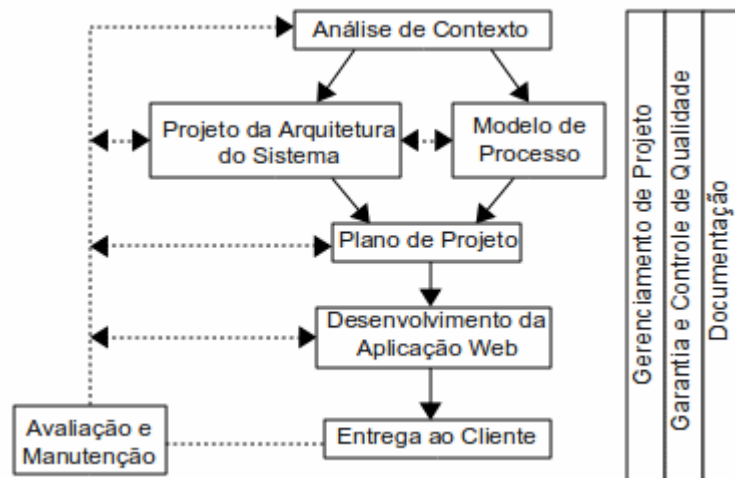


Figura 3.2: Modelo de processo de desenvolvimento de aplicações *Web* (MURUGESAN e GINIGE, 2005)

Nesse processo, como pode ser visto na Figura 3.2, o primeiro passo é a análise de contexto, no qual os objetivos, os requisitos e as necessidades dos usuários e da organização que necessita do sistema são compreendidos. Nesse passo, é importante perceber que os requisitos podem mudar e evoluir. No projeto da arquitetura do sistema, devem ser considerados os componentes do sistema e a forma como eles estão ligados. Juntamente com o projeto da arquitetura do sistema, é criado um modelo de processo para elucidar os principais requisitos e objetivos do sistema. Em seguida, segue a construção de um plano de projeto, para iniciar o desenvolvimento da aplicação. Depois do desenvolvimento, a aplicação é entregue ao cliente, e por fim, a aplicação mantém-se sob constante avaliação e manutenção. Combinados à essas fases, há as atividades de apoio como gerenciamento de projeto, garantia e controle de qualidade e documentação (MURUGESAN e GINIGE, 2005).

3.7. Considerações Finais

Neste capítulo foram apresentados os principais conceitos pertinentes às aplicações *Web*. Verificou-se que há diversas definições na literatura para o termo aplicações *Web*, assim como, para a sua classificação. Também foram

apresentadas a arquitetura e as principais características de tais aplicações. Em sequência, foram apontados os conceitos relativos à Engenharia *Web*. Por fim, foi apresentado um processo de desenvolvimento de aplicações *Web*. O capítulo seguinte apresenta alguns conceitos relacionados à arquitetura orientada a serviços.

4. ARQUITETURA ORIENTADA A SERVIÇOS

Este capítulo aborda alguns conceitos pertinentes à arquitetura orientada a serviços. Primeiramente, são apresentadas a definição e as características dessa arquitetura. Em seguida, a definição, benefícios e arquitetura de *Web services* são apresentados. Posteriormente, são expostos os protocolos e tecnologias usadas para implementar os *Web services*. Por fim, são apresentadas as considerações finais deste capítulo.

4.1. Definição

A arquitetura de software de um programa ou sistema de computação é a estrutura ou estruturas do sistema, que incluem elementos de software, as propriedades externamente visíveis desses elementos e as relações entre eles (BASS *et al.*, 2003). A Arquitetura Orientada a Serviços (SOA – *Service-Oriented Architecture*) tem como elemento fundamental o conceito de serviços. Nesta arquitetura, os recursos do software são empacotadas como serviços, que são bem definidos, independentes de módulos que fornecem funcionalidades de padrões de negócios e do estado ou contexto de outros serviços (PAPAZOGLU e HEUVEL, 2007).

Desta forma, Erl (2005) afirma que a arquitetura orientada a serviços representa um modelo que visa decompor um sistema em uma coleção de pequenas unidades relacionadas entre si. Os serviços são descritos em uma linguagem de definição padrão, tem uma interface publicada, e se comunicam uns com os outros solicitando a execução de suas operações, a fim de apoiar coletivamente uma tarefa comum de negócio ou processo. A arquitetura orientada a serviços proporciona flexibilidade e agilidade para eliminar barreiras na integração de aplicações, apresentando uma maneira lógica de conceber um sistema de software para fornecer serviços (PAPAZOGLU e HEUVEL, 2007).

Segundo Papazoglou e Heuvel (2007), um serviço é vinculado à uma interface e uma implementação do serviço. A interface do serviço define a identidade de um serviço e sua logística de chamada. A implementação do serviço realiza o trabalho que o serviço é designado a fazer. Sendo assim, um cliente a partir de qualquer dispositivo de comunicação utilizando qualquer plataforma computacional,

sistema operacional ou linguagem de programação pode usar o serviço, pois as interfaces são independentes de plataforma.

A arquitetura orientada a serviços cria um conjunto de serviços que podem se comunicar uns com os outros, usando interfaces para passar mensagens de um serviço para outro, ou coordenando uma atividade entre um ou mais serviços (PAPAZOGLU e HEUVEL, 2007). Desta forma, o cliente do serviço não tem acesso direto à sua implementação, exceto indiretamente, por intermédio de sua interface. Sendo assim, Josuttis (2007) apresenta a arquitetura orientada a serviços como um paradigma que melhora a flexibilidade na concepção de uma arquitetura de software e promove a integração de aplicações heterogêneas.

4.2. Princípios da Arquitetura Orientada a Serviços

Um conjunto de princípios que definem as regras básicas para o desenvolvimento, manutenção e uso da arquitetura orientada a serviços é apresentado por Erl (2005):

Reuso: a arquitetura orientada a serviços estabelece um ambiente que promove o reuso de recursos, pois a lógica é dividida em serviços;

Contrato formal: os contratos de serviços definem formalmente as regras e características do serviço, suas operações e mensagens;

Fraco acoplamento: um serviço mantém uma relação com os outros, porém apresenta independência sobre os demais serviços;

Abstração: os serviços funcionam como uma “caixa-preta”, escondendo seus detalhes para os demais serviços;

Composição: um serviço pode ser membro de uma composição de serviços, independente do tamanho e da complexidade da mesma;

Autonomia: os serviços têm um alto nível de controle sobre o seu ambiente e seus recursos básicos;

Estado: os serviços minimizam o consumo de recursos, permanecendo com informações de estado apenas quando necessário;

Descoberta: os serviços são externamente descritos para serem efetivamente encontrados e interpretados.

De acordo com Erl (2005), os princípios de autonomia, fraco acoplamento, abstração e contrato formal, são considerados o núcleo fundamental que forma a base da arquitetura orientada a serviços.

4.3. Web Services

A arquitetura orientada a serviços representa uma arquitetura que promove a orientação a serviços e é composta de serviços que podem ser implementados como *Web services* (ERL, 2005). Segundo Papazoglou e Heuvel (2007), *Web service* é uma tecnologia para implementar a arquitetura orientada a serviços e permite alcançar compartilhamento, reuso e interoperabilidade.

Erl (2005) afirma que os *Web services* apresentam um conjunto de especificações e protocolos que definem as interfaces e contratos dos serviços para a comunicação entre aplicações. Desta forma, há uma padronização para a integração de aplicações (CERAMI, 2002).

De acordo com Cerami (2002), um *Web service* é um serviço que está disponível por meio de uma rede, usa um padrão XML (*Extensible Markup Language*) para a troca de mensagens e não está ligado a qualquer sistema operacional ou linguagem de programação. Além disso, os serviços são descritos por uma gramática XML e descobertos por um mecanismo simples de busca.

Desta forma, os *Web services* estão se tornando rapidamente uma significativa tecnologia na evolução da *Web* e da computação distribuída (NEWCOMER, 2002). Além do mais, as interfaces de *Web services* podem mapear qualquer tipo de programa de software, sistema de *middleware*, sistema de gerenciamento de banco de dados ou aplicação empacotada, unindo diferentes domínios de software.

4.4. Arquitetura de Web Services

A arquitetura de *Web services*, segundo Cerami (2002), pode ser visualizada de duas maneiras. A primeira é analisar os papéis individuais de cada ator do *Web service* e a segunda é analisar a pilha de protocolos de *Web service*.

Em relação aos papéis individuais de cada ator do *Web service*, conforme apresenta a Figura 4.1, há três papéis na arquitetura:

- provedor de serviço: implementa o serviço e torna-o disponível na Internet;
- consumidor de serviço: utiliza um *Web service* existente por meio da abertura de uma conexão de rede e envia uma solicitação XML; e
- registro de serviço: é um diretório logicamente centralizado de serviços. Fornece um local centralizado onde os desenvolvedores podem publicar novos serviços ou encontrar os já existentes.

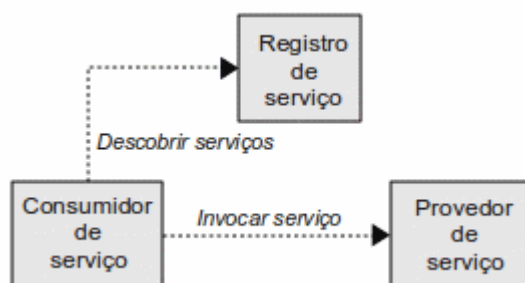


Figura 4.1: Papéis de *Web services* (CERAMI, 2002)

A interação entre os papéis em um *Web service* envolve as operações: publicação, descoberta e ligação (KREGGER, 2001). A Figura 4.2 apresenta essas operações, artefatos e suas interações. As operações são descritas a seguir:

Publicação: para ser acessível, a descrição do serviço deve ser publicada para que o consumidor de serviço possa encontrá-la;

Descoberta: o consumidor de serviço recupera uma descrição de serviço diretamente ou consulta o registro de serviços para encontrar o tipo de serviço desejado;

Ligação: o consumidor de serviços invoca ou inicia uma interação com o serviço em tempo de execução usando os detalhes de ligação na descrição do serviço para localizar, contatar e chamar o serviço.

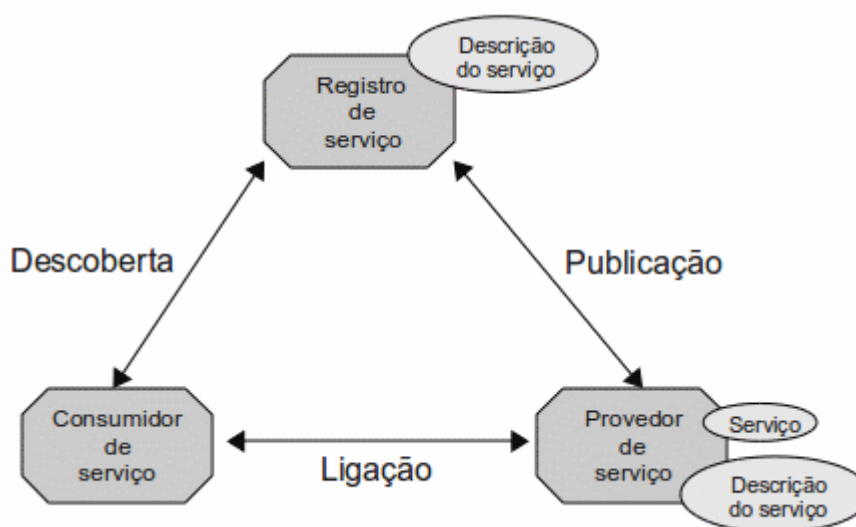


Figura 4.2: Papéis, operações e artefatos de *Web services* (KREGER, 2001)

Os artefatos em um *Web service* são o serviço e a descrição do serviço. De acordo com Kreger (2001), serviço é um módulo de software implantado em plataformas de rede fornecido pelo provedor de serviços e a descrição do serviço contém os detalhes da interface e implementação do serviço.

Em relação à pilha de protocolos de *Web service*, conforme apresenta a Figura 4.3, há quatro camadas principais (CERAMI, 2002). Essas camadas são descritas a seguir:

Transporte: esta camada é responsável pelo transporte de mensagens entre as aplicações. Esta camada inclui o protocolo de transferência de hipertexto (HTTP - *Hypertext Transfer Protocol*), protocolo de transferência de e-mail (SMTP - *Simple Mail Transfer Protocol*), protocolo de transferência de arquivos (FTP - *File Transfer Protocol*), entre outros;

Mensagens XML: esta camada é responsável pela codificação de mensagens em um formato XML comum, de modo que as mensagens possam ser entendidas em cada extremidade. Esta camada inclui o XML-RPC e o protocolo para troca de informações (SOAP - *Simple Object Access Protocol*);

Descrição: esta camada é responsável por descrever a interface pública para um *Web service* específico. Esta camada inclui a linguagem de definição de *Web services* (WSDL - *Web Services Definition Language*);

Descoberta: esta camada é responsável pela centralização dos serviços em um registro comum, proporcionando a publicação e descoberta de serviços. Esta camada inclui o protocolo de descrição, descoberta e integração de serviços (UDDI - *Universal Description, Discovery and Integration*).

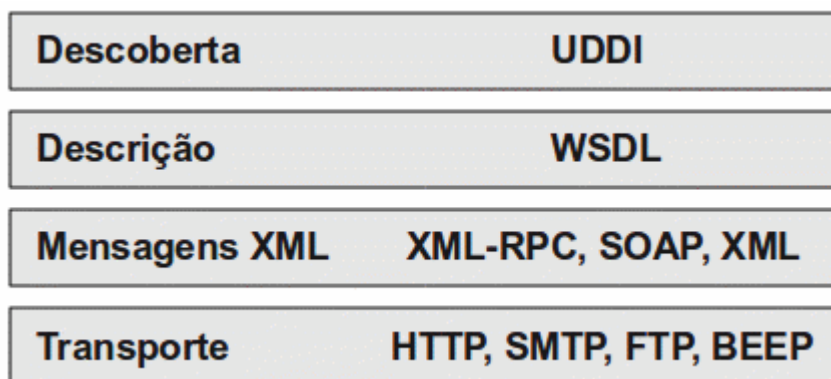


Figura 4.3: Camadas de protocolos de *Web services* (CERAMI, 2002)

Conforme os *Web services* evoluem, novas camadas podem ser adicionadas e outras tecnologias podem ser incorporadas a cada camada. Os protocolos estão detalhados na seção a seguir.

4.5. Protocolos e Tecnologias de Web Services

Para a implementação de *Web services* são necessárias algumas tecnologias e protocolos, os quais são descritos a seguir.

4.5.1. XML - Extensible Markup Language

XML é um simples e flexível formato baseado em texto para representar informação estruturada (W3C, 2009). É derivado de um antigo formato padrão chamado SGML (*Standard Generalized Markup Language*). Foi desenvolvida pela W3C (*World Wide Web Consortium*) e concebida para a troca de uma ampla variedade de dados na *Web*. De acordo com a W3C (2009), as principais características do XML são:

- ser diretamente utilizável na Internet;
- suportar uma grande variedade de aplicações;
- ser compatível com SGML;

- facilidade para escrever programas que processam documentos XML;
- documentos XML devem ser legíveis e claros para os humanos;
- o projeto de XML deve ser formal e conciso; e
- facilidade para criar documentos XML.

No contexto de *Web service*, o XML não é utilizado somente como formato da mensagem, mas também como a forma no qual os serviços são definidos e implementados (NEWCOMER, 2002). Sendo assim, o XML é a base sobre o qual os *Web services* são construídos, fornecendo um formato para descrição, armazenamento e transmissão de dados trocados pelos mesmos.

4.5.2. SOAP - Simple Object Access Protocol

SOAP é um protocolo de comunicação baseado em XML para troca de informações entre computadores (CERAMI, 2002). O transporte é por meio do protocolo HTTP e o SOAP é independente de plataforma, permitindo assim, que diversas aplicações possam se comunicar. De acordo com Cerami (2002), a especificação do protocolo SOAP define três partes principais:

Especificação do envelope SOAP: define regras específicas para encapsular os dados que serão transferidos entre computadores e inclui dados específicos da aplicação, tais como: nome do método a ser invocado, parâmetros do método ou valores de retorno. Também pode incluir informações sobre quem deve processar o conteúdo do envelope e, em caso de falha, como codificar mensagens de erro;

Regras de codificação dos dados: para a troca de dados, os computadores devem chegar a um acordo sobre as regras de codificação e especificação de tipos de dados. O protocolo SOAP apresenta um conjunto de convenções para codificação de tipos de dados, baseadas na especificação XML *Schema* do W3C;

Convenções RPC: o protocolo SOAP pode ser usado por diversos sistemas de mensagens, incluindo unidirecionais e bidirecionais. Isso permite que uma aplicação cliente especifique o nome de um método remoto, incluindo diversos parâmetros, e receba uma resposta do servidor.

A estrutura da mensagem SOAP, apresentada pela Figura 4.4, é definida em documentos XML por meio dos seguintes elementos (CERAMI, 2002):

- *envelope*: identifica o documento XML como uma mensagem SOAP e é responsável por definir o conteúdo da mensagem;
- *header*: contém os dados do cabeçalho;
- *body*: contém as informações de chamada e resposta ao servidor; e
- *fault*: contém as informações dos erros ocorridos no envio da mensagem. Esse elemento só aparece nas mensagens de resposta do servidor.

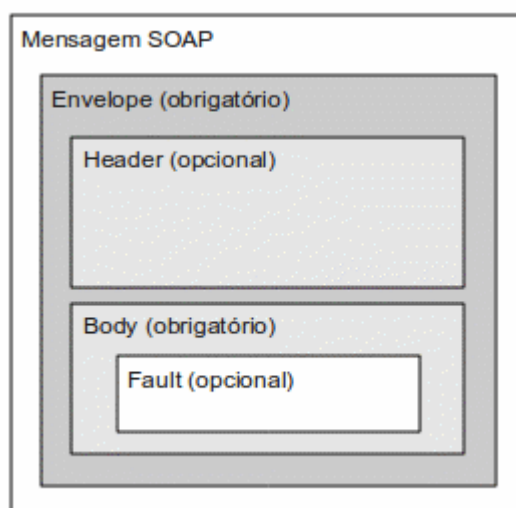


Figura 4.4: Elementos de uma mensagem SOAP (CERAMI, 2002)

Uma mensagem SOAP de uma requisição cliente pode incluir o nome do método à invocar e quaisquer parâmetros exigidos. A Figura 4.5 ilustra uma simples mensagem SOAP de uma requisição cliente contendo o elemento `Envelope`, um bloco opcional `Header` que define os atributos da mensagem e um bloco `Body` que contém a própria mensagem. Neste exemplo, o elemento `getTemp` corresponde ao nome do método remoto localizado na URL: `www.din.uem.br/~rlvivian/services`. Cada parâmetro do método é um subelemento, que neste caso, é o elemento `zipCode` definido como `87020900`.

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Header>
    ...
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:getTemp xmlns:ns1="http://www.din.uem.br/~rlvivian/services/">
      <zipCode>87020900</zipCode>
    </ns1:getTemp>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Figura 4.5: Mensagem SOAP de uma requisição cliente

A Figura 4.6 ilustra a resposta à requisição SOAP apresentada no exemplo anterior. Assim como a requisição, a resposta inclui os elementos `Envelope` e `Body`. Entretanto, desta vez o elemento `Body` contém um elemento `getTempResponse`, que corresponde ao pedido inicial. Esse elemento de resposta inclui um simples elemento de retorno, indicando um tipo de dado `xsd:float`, com a temperatura para o CEP 87020900 de 26 graus centígrados.

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns1:getTempResponse
      xmlns:ns1="http://www.din.uem.br/~rlvivian/services/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="xsd:float">26.0</return>
    </ns1:getTempResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Figura 4.6: Mensagem SOAP de resposta

Sendo assim, o SOAP torna possível que *Web services* troquem dados, não importando onde eles estão localizados no ambiente de rede (NEWCOMER, 2002). Além do mais, este protocolo fornece um mecanismo simples e extensível para o mapeamento de múltiplos tipos de interações de mensagens e sistemas de software subjacente.

4.5.3. WSDL - Web Service Description Language

WSDL é uma especificação que define como descrever *Web services* em uma gramática XML comum, fornecendo uma interface pública para o *Web service* (CERAMI, 2002). Esse protocolo apresenta quatro informações principais:

- informações de interface descrevendo as funções disponíveis;
- informações de tipo de dados para as mensagens de solicitação e resposta;
- informações sobre o protocolo de transporte a ser utilizado; e
- informações de endereço para localizar o serviço especificado.

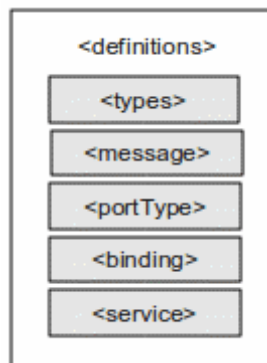


Figura 4.7: Elementos da especificação WSDL (CERAMI, 2002)

A WSDL representa um contrato entre o serviço consumidor e o serviço provedor. Cerami (2002) afirma que “por meio dessa especificação um cliente pode localizar um *Web service* e chamar qualquer uma de suas funções públicas disponíveis”. De acordo com Cerami (2002), a WSDL é dividida em seis elementos principais, conforme apresenta a Figura 4.7:

definitions: é o elemento raiz do documento WSDL e define o nome do *Web service*, declara vários *namespaces* utilizados no documento e contém todos os elementos do serviço;

types: descreve todos os tipos de dados usados nas mensagens entre o cliente e servidor;

message: descreve uma mensagem de sentido único, se é uma mensagem simples de requisição ou de resposta e define o nome da mensagem;

portType: descreve o conjunto abstrato das operações disponibilizadas por um *Web service*;

binding: descreve os detalhes concretos de como o serviço será transmitido;

service: define o endereço para invocar o serviço especificado, incluindo uma URL para o serviço SOAP.

A Figura 4.8 apresenta um exemplo de arquivo WSDL.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
  targetNamespace="http://www.din.uem.br/~rlvivian/wSDL/HelloService.wSDL"
  xmlns="http://schemas.xmlsoap.org/wSDL/"
  xmlns:soap="http://schemas.xmlsoap.org/wSDL/soap/"
  xmlns:tns="http://www.din.uem.br/~rlvivian/wSDL/HelloService.wSDL"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>
  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>
  <binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding
      style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </output>
    </operation>
  </binding>
  <service name="Hello_Service">
    <documentation>Arquivo WSDL para HelloService</documentation>
    <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
        location="http://localhost:8080/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>
```

Figura 4.8: Exemplo de arquivo WSDL

O elemento `definitions` especifica que este documento é o `HelloService`. Ele também inclui vários *namespaces* que referenciam várias especificações externas, como o WSDL, SOAP e XML *Schema*. Dois elementos `message` são definidos. O primeiro representa uma mensagem de requisição `SayHelloRequest`, e o segundo representa uma mensagem de resposta `SayHelloResponse`. Cada uma destas mensagens contém um elemento `part`. Para a requisição o `part` especifica os parâmetros da função `firstName`, e para a resposta especifica o valor de retorno da função `greeting`. O elemento `portType` define a operação `sayHello`, que consiste de uma mensagem `input` (`SayHelloRequest`) e uma mensagem `output` (`SayHelloResponse`). O elemento `binding` fornece detalhes sobre como a operação `portType` será transmitida. O elemento `service` especifica o local (`http://localhost:8080/soap/servlet/rpcrouter`) do serviço.

Sendo assim, a WSDL fornece um mecanismo para a definição de interfaces para *Web services* (NEWCOMER, 2002). Além do mais, essa especificação contém uma descrição dos tipos de dados e estruturas usadas em mensagens de *Web services*, bem como as informações necessárias para o mapeamento da definição dos mesmos.

4.5.4. UDDI - Universal Description, Discovery and Integration

UDDI é uma especificação técnica para descrever, descobrir e integrar *Web services*, por meio de um diretório distribuído de negócios e serviços (CERAMI, 2002). De acordo com Cerami (2002), os dados registrados no UDDI são divididos em três categorias principais:

Páginas brancas (*white pages*): incluem informações gerais (e.g. nome, descrição, contato, endereço e telefone) sobre uma empresa;

Páginas amarelas (*yellow pages*): incluem dados de classificação geral para a empresa ou serviço oferecido;

Páginas verdes (*green pages*): contém as informações técnicas sobre um *Web service*, incluindo um ponteiro para uma especificação externa ou um endereço para chamar o *Web service*.

Cerami (2002) afirma que, o UDDI inclui um XML *Schema* que descreve quatro tipo de informações principais.

businessEntity: esse elemento apresenta informações sobre a empresa que publica o *Web service*. Esta informação contém o nome da empresa, descrição, endereço e contato para informações;

businessService: esse elemento inclui informações sobre um único *Web service* ou um grupo de *Web services* relacionados, oferecendo informações sobre a descrição de cada serviço em termos de negócios;

bindingTemplate: esse elemento apresenta informações sobre como obter o acesso e quais os pontos de acesso a um *Web service*;

tModel: esse elemento contém informações que descrevem uma especificação técnica do serviço.

Desta forma, o UDDI fornece um mecanismo flexível, poderoso e extensível para registrar e descobrir informações de negócios por meio da Internet (NEWCOMER, 2002).

4.6. Considerações Finais

Neste capítulo foram apresentados os principais conceitos e princípios pertinentes à arquitetura orientada a serviços. Na sequência, foram apresentados os conceitos sobre *Web services* e sua arquitetura. Por fim, foram abordados os protocolos e tecnologias para o desenvolvimento de *Web services*. O capítulo seguinte apresenta os elementos da arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto.

5. ELEMENTOS DA ARQUITETURA ORIENTADA A SERVIÇOS EM APLICAÇÕES WEB SENSÍVEIS AO CONTEXTO

Este capítulo aborda os elementos da arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto. Primeiramente, é apresentada a visão geral da arquitetura. Em seguida, são detalhados os elementos: *Suporte à captura*, *Representação de contexto*, *Suporte ao processamento* e *Mecanismos de apresentação*, além do *Repositório*. Por fim, são apresentadas a arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto e as considerações finais deste capítulo.

5.1. Visão Geral da Arquitetura

A arquitetura de software modela a estrutura de um sistema e a maneira pela qual componentes de dados e procedimentos colaboram entre si (PRESSMAN, 2006). Desta forma, Chaari et al. (2004) afirmam que para o desenvolvimento de aplicações sensíveis ao contexto, são necessários dois objetivos à serem alcançados:

- projetar uma arquitetura que apoie a percepção de contexto em tempo de execução; e
- projetar uma aplicação com a finalidade de ser sensível ao contexto.

Dey (2000) destaca quatro elementos essenciais à percepção de contexto (*context-awareness*): captura, representação, armazenamento e disseminação das informações contextuais. Sendo assim, Chaves (2009) apresenta o modelo DiSEN-CSE (*DiSEN-Context Sensitive Environment*) aplicado ao ambiente de desenvolvimento distribuído de software DiSEN (*Distributed Software Engineering Environment*), para permitir a disseminação de informações contextuais entre os diversos *workpaces* envolvidos em um trabalho cooperativo. A Figura 5.1 detalha o modelo DiSEN-CSE.

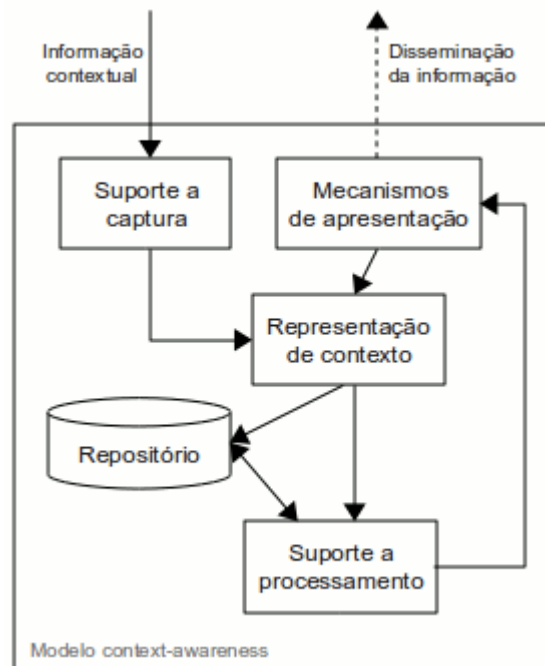


Figura 5.1: Modelo baseado em *context-awareness* para o DiSEN (CHAVES, 2009)

De acordo com Chaves (2009), esse modelo possui quatro elementos essenciais: *Suporte à captura*, *Representação de contexto*, *Suporte ao processamento* e *Mecanismos de apresentação*, além de um *Repositório* para o armazenamento das informações contextuais, que são descritos nas seções a seguir.

5.2. Suporte à Captura

O *Suporte à captura* é responsável por reconhecer as mudanças que ocorrem no contexto do ambiente. Essas mudanças podem ser capturadas de um usuário capaz de fornecer as informações referentes ao seu contexto atual ou de dispositivos físicos como sensores (CHAVES, 2009). Sendo assim, a captura de contexto pode ser transparente para o usuário ou pode ser obtido diretamente por meio da inserção manual do próprio usuário.

Quanto à origem das informações, há três tipos de fontes (CHAVES, 2009): **manual**, quando os usuários registram o seu próprio contexto a partir de uma determinada interface; **local**, quando estão armazenadas em arquivos de configuração, interfaces gráficas ou derivadas de outros sensores locais; e

repositório, quando estão armazenadas em algum repositório de dados comum, como um servidor de banco de dados central.

5.3. Representação de Contexto

O elemento *Representação de contexto* é responsável por receber as informações contextuais capturadas e representá-las formalmente (CHAVES, 2009). Sendo assim, as informações provenientes do *Suporte à captura*, devem ser mapeadas pelo elemento *Representação de contexto* por meio de um modelo formal e, relacionadas com as demais informações contextuais que estão no *Repositório*, com base no modelo de representação definido.

Há na literatura diferentes abordagens de modelos de representação de contexto (STRANG e LINNHOFF-POPIEN, 2004), entre eles, pares chave-valor, modelos baseados em marcação, em representações gráficas, em orientação a objetos, entre outros, conforme discutido na Seção 2.6. Para a arquitetura proposta foi adotado o modelo baseado em marcação, com base no padrão XML, pois além de ser extensível, flexível e aberto (HELD et al., 2002), esse modelo facilita a comunicação por diferentes aplicações.

5.4. Repositório

O *Repositório* é o elemento responsável por armazenar as informações contextuais. Para o modelo proposto, as informações contextuais capturadas e representadas podem ser persistentes ou transientes (CHAVES, 2009). De acordo com Chaves (2009), as informações transientes “são importantes para um determinado momento, mas não serão utilizadas no futuro, podendo ser disparadas para os *Mecanismos de apresentação*, sem a necessidade de armazenamento”. Ao contrário, as informações persistentes “são aquelas que devem permanecer em um repositório de dados para consultas futuras, formando um histórico de informações de contexto”. Esse repositório, quando necessário, pode ser acessado pelo *Suporte ao processamento* para a recuperação das informações nele contidos.

O *Repositório* mantém **tipos de entidades**, seus **atributos** e **tipos de contextos**. Exemplos de **tipos de entidades** são usuário, hotel, restaurante, teatro,

entre outros. Exemplos de **atributos** são idade, endereço, entre outros, e exemplos de **tipos de contexto** são localização, identidade, tempo, entre outros.

Esse elemento armazena também o **perfil do usuário**, pois dados significativos podem ser obtidos diretamente do mesmo. Informações contidas no perfil podem ser consideradas como informação contextual, no sentido que descreve o ambiente em que os usuários desejam operar (COSTA, 2003). De acordo com Costa (2003), o **perfil do usuário** pode apresentar os seguintes conceitos:

- identidade: formas para identificar o usuário;
- características: aspectos pessoais, que são objetivos e independente do contexto, como data de nascimento, sexo, primeiro nome, entre outros;
- preferências: configurações subjetivas pessoais;
- interesses: descreve como um usuário está interessado em certos conceitos, como gosto musical por ópera;
- avaliações: indicações explícitas de como um objeto específico é interessante para o usuário; e
- histórico: registro de todas as ações tomadas pelo usuário ao usar o sistema.

Costa (2003) afirma que as características sobre o **perfil do usuário**, são importantes para permitir a adaptação de sistemas sensíveis ao contexto de acordo com as necessidades e preferências dos usuários.

5.5. Suporte ao Processamento

O elemento *Suporte ao processamento* é um mecanismo de raciocínio capaz de deduzir contextos implícitos nas informações recebidas e corrigir possíveis inconsistências, com base nos relacionamentos existentes entre os conjuntos de informação, criados na *Representação de contexto* (CHAVES, 2009). As informações persistentes são obtidas do *Repositório*, enquanto as informações transientes são recebidas diretamente da *Representação de contexto*. Entretanto, Chaves (2009) afirma que todas as informações precisam ser processadas antes de serem enviadas para os *Mecanismos de apresentação*, para que as informações disseminadas sejam interpretadas e tenham, assim, maior relevância.

Dessa forma, as informações contextuais capturadas pelo *Suporte à captura* e modeladas pela *Representação de contexto*, são reunidas e uniformemente disponíveis para a plataforma por meio de interpretação semântica e, em seguida, com base nessa interpretação, podem ser inferidos novos contextos. Assim, o contexto de baixo nível é traduzido para uma representação de alto nível. Segundo Costa (2003), o *Suporte ao processamento* apresenta os seguintes recursos:

Coleta de contexto: o *Suporte ao processamento* é capaz de coletar contexto do elemento *Representação de contexto*;

Agregação de contexto: o *Suporte ao processamento* é capaz de agregar contexto, quando necessário;

Inferência de contexto: o *Suporte ao processamento* é capaz de inferir contexto de outro contexto.

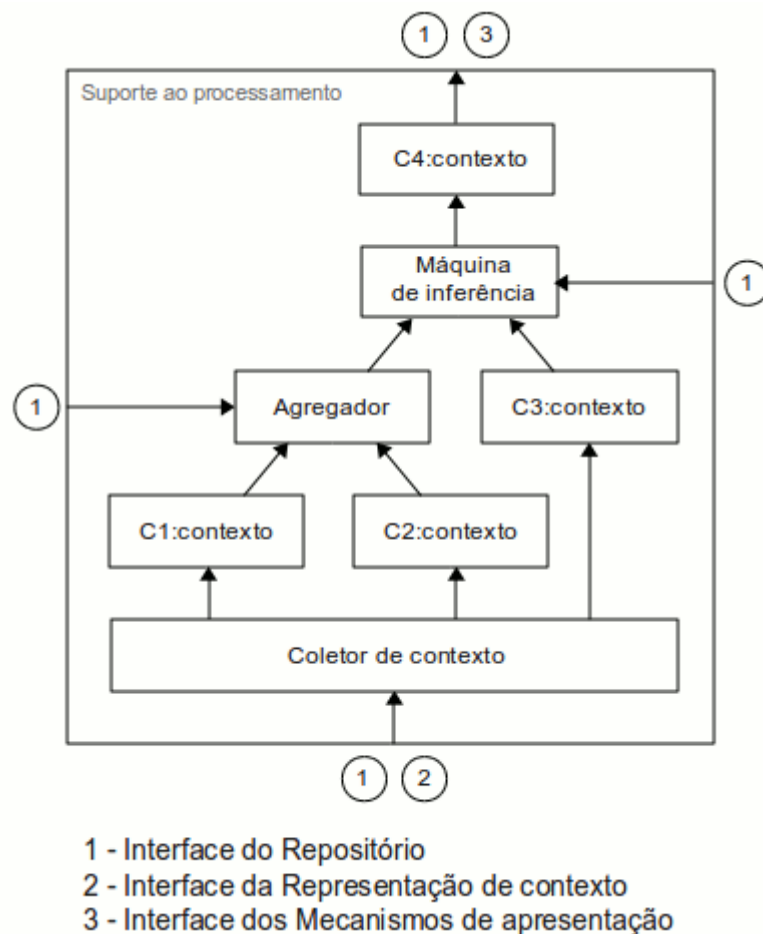


Figura 5.2: Exemplo de configuração do *Suporte ao processamento* (adaptado de COSTA, 2003)

A Figura 5.2 apresenta uma possível configuração para o *Suporte ao processamento*. O **coletor de contexto** é responsável por coletar contexto dos elementos *Repositório* e *Representação de contexto*. O **agregador** e a **máquina de inferência** são responsáveis por agregar contexto e inferir contexto, respectivamente. Este exemplo de configuração apresenta três contextos sendo coletados pelo **coletor de contexto** (C1, C2 e C3). Os contextos C1 e C2 são agregados porque eles estão associados com a mesma entidade (por exemplo, tempo e localização do usuário). Estes contextos agregados e o contexto C3 são passados para a **máquina de inferência** avaliar uma certa regra de inferência. O resultado desta regra de inferência é o contexto C4, que pode ser fornecido para o *Repositório*.

O contexto pode ser derivado de outro contexto (COSTA, 2003). Assim, para inferir novos contextos, são necessárias regras de inferência na **máquina de inferência**. Uma regra de inferência define relações lógicas entre as informações (de contexto ou não), a fim de obter uma nova informação que não é possível obter diretamente do ambiente (COSTA, 2003). Por exemplo, se a localização atual de um usuário é dentro de seu local de trabalho, podemos inferir que o usuário está trabalhando.

5.6. Mecanismos de Apresentação

Após serem representadas e processadas, as informações contextuais estão prontas para serem disponibilizadas para os *Mecanismos de apresentação* (CHAVES, 2009). Esse elemento constitui uma interface entre o sistema de gerenciamento de contexto e o sistema que se adaptará, exibindo os dados pertinentes para cada serviço da aplicação. Durante a execução, os serviços podem “extrair” os valores de contexto cada vez que eles precisarem ou o *Mecanismo de apresentação* pode “lançar” o contexto cada vez que ele é atualizado.

5.7. A Arquitetura

Baseada nos elementos essenciais à percepção de contexto: captura, representação, armazenamento e disseminação das informações contextuais (DEY, 2000) e no modelo DiSEN-CSE (CHAVES, 2009), foi definida uma arquitetura que

visa integrar aplicações *Web* sensíveis ao contexto à outros sistemas por meio de serviços. Conforme apresenta a Figura 5.3, os elementos do modelo DiSEN-CSE, que foram discutidos nas seções anteriores, estão em uma camada separada do núcleo e da interface do usuário na aplicação *Web*.

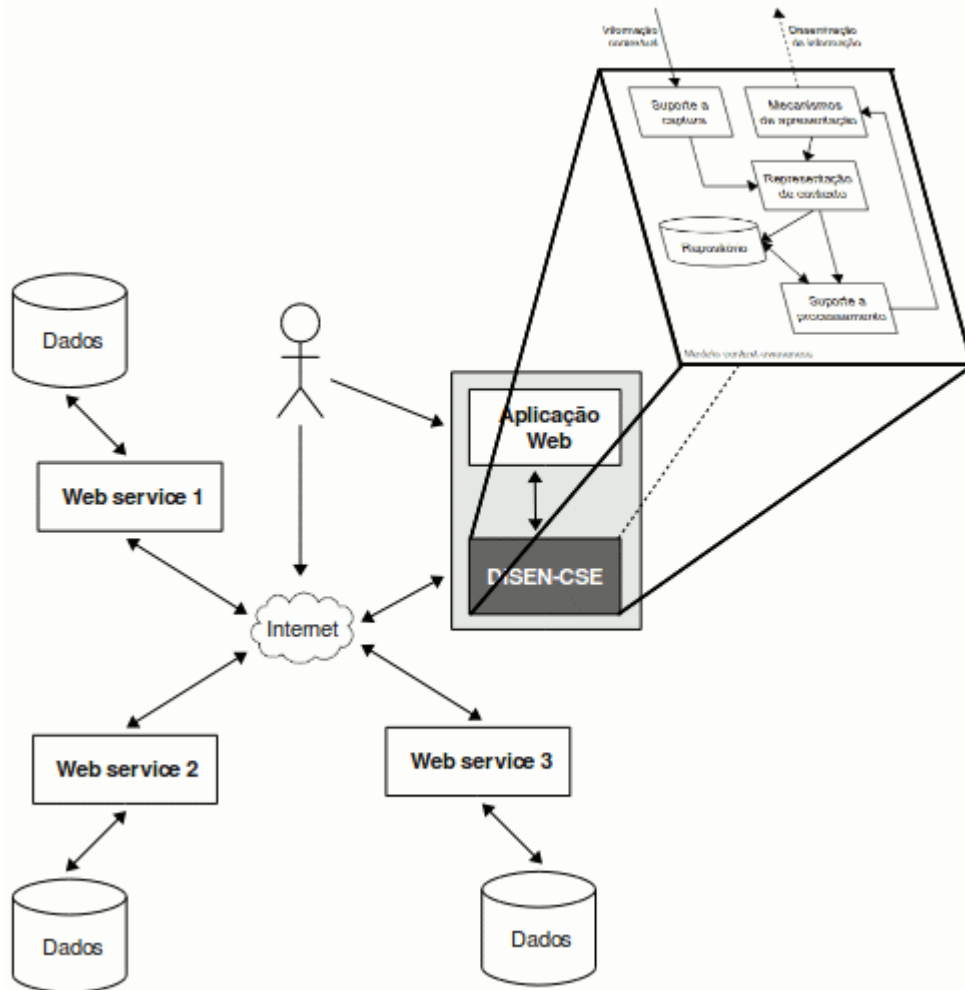


Figura 5.3: Arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto

Nesta arquitetura os *Web services* obtêm informações de outros sistemas e transmitem ao gerenciador de contexto da aplicação *Web*. Desta forma, a aplicação *Web* e os sistemas de terceiros podem ser desenvolvidos em diferentes linguagens e tecnologias, pois a interoperabilidade é garantida pela arquitetura orientada a serviços.

5.8. Considerações Finais

Neste capítulo foi apresentada uma arquitetura orientada a serviços como o ambiente para aplicações sensíveis ao contexto. Esta arquitetura possui alguns elementos de gerenciamento de contexto que foram detalhados, como *Suporte à captura*, *Representação de contexto*, *Repositório*, *Suporte ao processamento* e *Mecanismos de apresentação*. Desta forma, foi apresentada uma arquitetura que integra aplicações *Web* sensíveis ao contexto com outras aplicações heterogêneas. O capítulo seguinte apresenta a modelagem e a implementação da arquitetura proposta em um cenário como exemplo de aplicação.

6. EXEMPLO DE APLICAÇÃO

Este capítulo apresenta um exemplo de aplicação para um sistema *Web* sensível ao contexto, considerando-se a arquitetura orientada a serviços. Primeiramente, é apresentado o cenário em que foi aplicado. Em seguida, são detalhados a modelagem dos casos de uso e a arquitetura do protótipo. A implementação e os resultados são abordados na sequência. Por fim, são apresentadas as considerações finais deste capítulo.

6.1. Cenário de Exemplo

Para a implementação do protótipo foi considerado um cenário de viagens e passeios. Uma típica viagem de negócios ou lazer envolve várias questões, tais como: aviões, trens, táxis, hotéis, restaurantes, passeios, locação de carros, dentre outras. Tal cenário provê informações contextuais que podem ser usadas em favor do usuário.

Desta forma, foi implementado uma aplicação *Web*, chamada *Contexttrip*, que apresenta ao usuário informações úteis sobre o plano de viagem, baseadas no contexto do próprio usuário. Essa ferramenta funciona como um “agente pessoal de viagens”, organizando informações sobre a viagem como o hotel a se hospedar, informações de passagens aéreas, restaurantes, locação de carros, teatros, *shows*, parques, museus, igrejas, passeios, enfim, um mapa de atrações locais, previsão do tempo e outras informações, “empacotadas” em um itinerário.

O *Contexttrip* consome serviços de outros sistemas por meio de *Web services*. Assim, a aplicação pode integrar *Web services* de sistemas de hospedagem de hotéis, sistemas de passagem de companhias aéreas e sistemas de locação de automóveis que podem fornecer informações importantes para o contexto de uma viagem. Esse cenário é apresentado pela Figura 6.1.

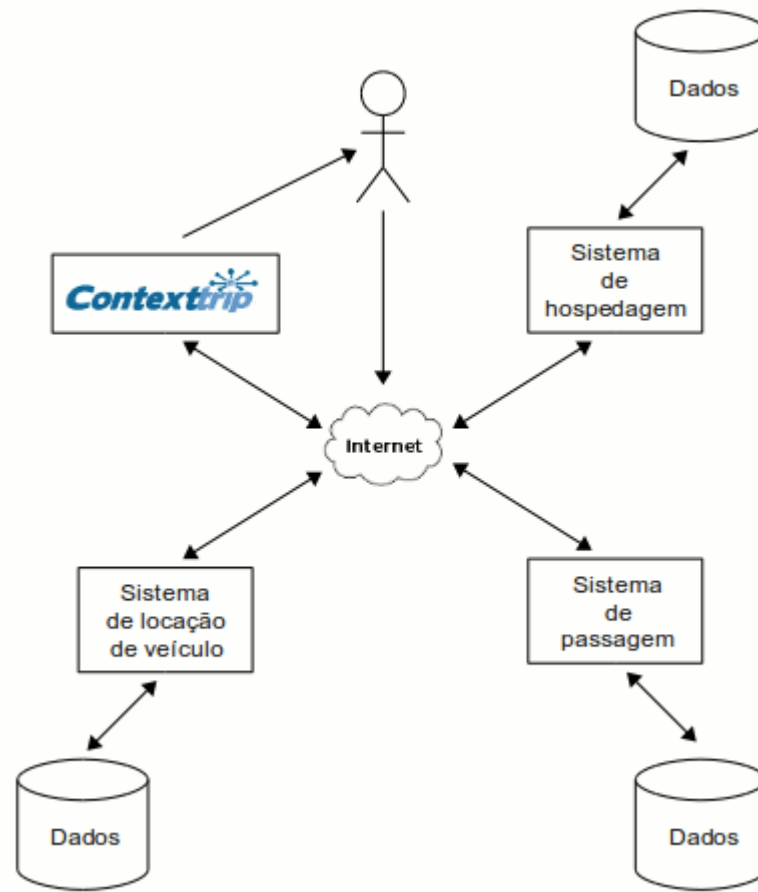


Figura 6.1: Cenário de exemplo

Um usuário pode, por exemplo, adicionar no *Contexttrip* uma viagem para a cidade de Gramado, com data de início e fim da viagem. Baseado nessas informações contextuais, a aplicação acessa *Web services* de sistemas de hospedagem e apresenta para o usuário os hotéis ou pousadas disponíveis para aquele destino e período. Além disso, podem ser apresentadas as opções de companhias aéreas e de locação de automóveis, assim como, opções de passeios, apresentações culturais, restaurantes, teatros, entre outros, de acordo com as informações contextuais de suas preferências e interesses.

Desta forma, esta aplicação *Web* usa essas informações contextuais para fornecer serviços ou outras informações relevantes ao usuário. Assim, há a adaptação de comportamento da aplicação em resposta às alterações de contexto, reduzindo a realização de tarefas pelo usuário.

6.2. Modelagem dos Casos de Uso

O caso de uso *Gerenciar Viagem*, é realizado pelo ator *Usuário* que pode gerenciar as informações relacionadas à sua viagem. No caso de uso *Gerenciar Destino*, o usuário insere o nome da cidade que pretende visitar. No caso de uso *Gerenciar Início e Fim*, o usuário insere o período que ficará naquela cidade. Essas informações servirão como informações contextuais para a aplicação.

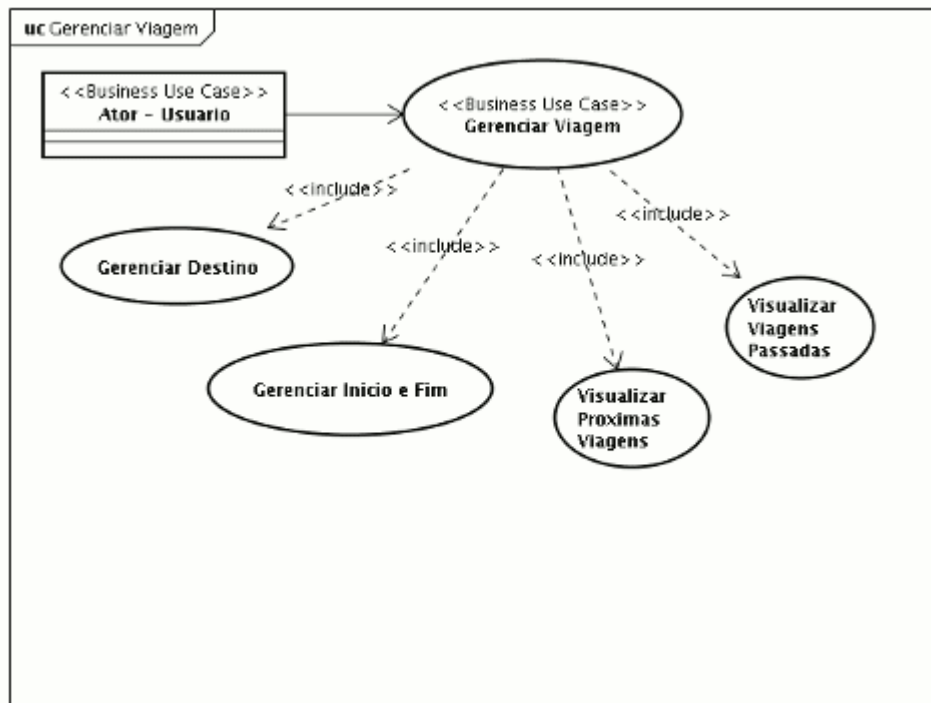


Figura 6.2: Gerenciar Viagem Use Case

Além disso, há os casos de uso *Visualizar Próximas Viagens* e *Visualizar Viagens Passadas*, conforme apresenta a Figura 6.2. A documentação de todos os casos de uso e os diagramas de classes pode ser encontrada no Apêndice A.

6.3. Arquitetura do Protótipo

A Figura 6.3 apresenta uma visão da arquitetura do protótipo *Contexttrip*. O usuário pode acessar a aplicação baseada na *Web* por meio de um dispositivo que tenha conexão à Internet. Nessa aplicação o gerenciador de contexto está em uma camada separada do núcleo e interface do usuário, baseado no modelo *context-awareness* DiSEN-CSE proposto por Chaves (2009), discutido no Capítulo 5.

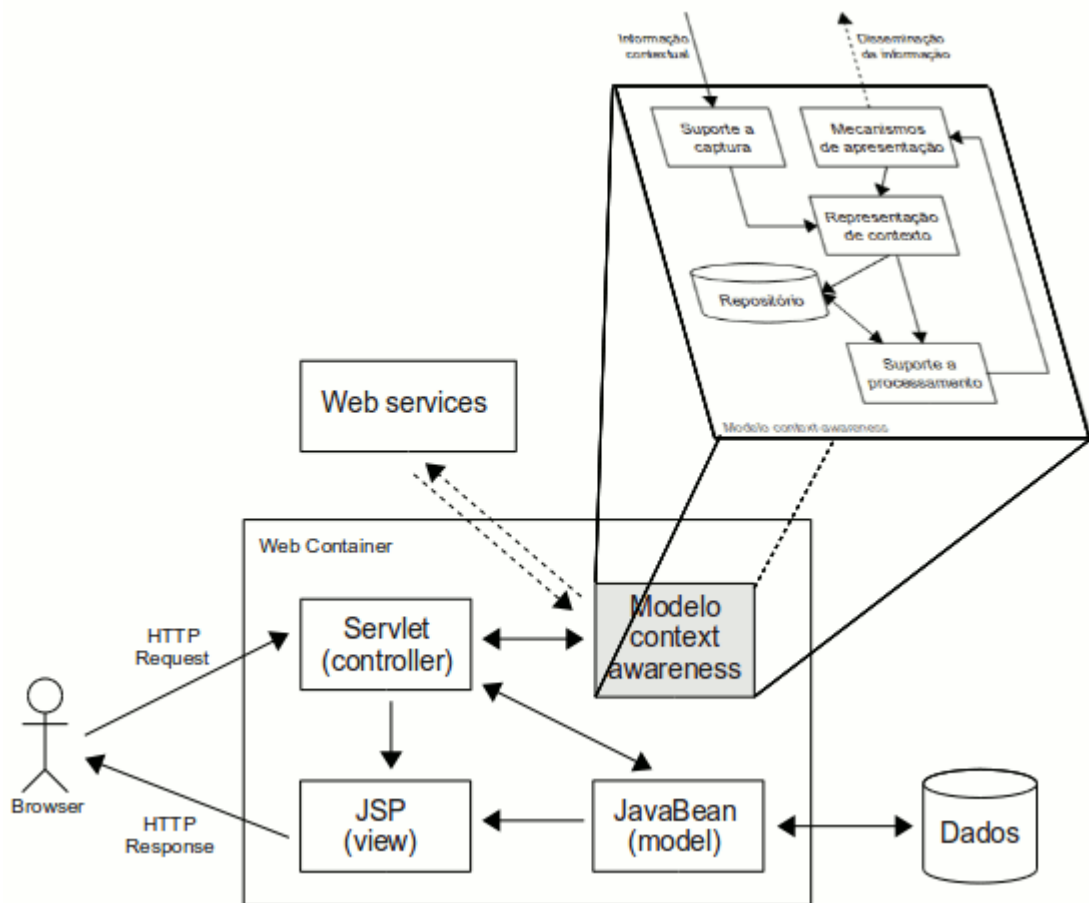


Figura 6.3: Arquitetura do *Contexttrip*

O *Contexttrip* é uma aplicação que explora *Web services* e sendo assim, pode obter informações de sistemas heterogêneos e repassá-las ao gerenciador de contexto, para que possam ser processadas e transmitidas à interface do usuário também por meio de *Web services*. Desta forma, o núcleo do *Contexttrip*, o modelo *context-awareness* e os sistemas de terceiros podem ser desenvolvidos em diferentes linguagens e tecnologias, pois a arquitetura orientada a serviços garante a interoperabilidade por meio dos padrões abertos dos *Web services*.

6.4. Implementação

O desenvolvimento do protótipo tem por objetivo demonstrar os conceitos relacionados à arquitetura orientada a serviços em uma aplicação *Web* sensível ao contexto aplicados no cenário de exemplo. Desta forma, foram usadas tecnologias de *Web services* e a linguagem Java para a implementação do protótipo. Além disso, foi usado o Axis2 para gerar automaticamente o WSDL das classes Java. O

Axis2 é uma API (*Application Programming Interface*) que fornece apoio para mapear WSDL para Java e vice-versa.

O *Contexttrip* fornece uma interface gráfica para o usuário adicionar as viagens que pretende realizar. Essas informações sobre a viagem (destino, data de início e fim) servirão como informações de contexto para a aplicação. Assim, por meio do componente *Suporte à captura* que apresenta um cliente *Web service*, acessa informações sobre hotéis de acordo com as informações contextuais sobre a viagem.

Conforme apresenta a Figura 6.4, o cliente faz chamadas por meio de um *stub*, que é um objeto local ao cliente que funciona como um intermediário (*proxy*) para o serviço remoto. Dessa forma, esse cliente faz uma requisição ao serviço disponível *ConsultaHospedagem*, passando como parâmetro o *destino* e em seguida a *ConsultaHospedagemResponse* recebe a resposta da solicitação.

```
public class CapturaContexto {  
    .  
    .  
    ConsultaHospedagemWSSStub stub = new ConsultaHospedagemWSSStub();  
    ConsultaHospedagemWSSStub.ConsultaHospedagem req = new ConsultaHospedagemWSSStub.ConsultaHospedagem();  
    req.setCidade(destino);  
    ConsultaHospedagemWSSStub.ConsultaHospedagemResponse res = stub.consultaHospedagem(req);  
    resposta = res.get_return();  
    .  
    .  
}
```

Figura 6.4: Trecho de código da classe *CapturaContexto*

Após as informações contextuais em relação à hospedagem serem capturadas, estas são mapeadas e representadas no modelo baseado em marcação XML. A Figura 6.5 apresenta um exemplo de representação de algumas informações contextuais capturadas. Essas informações contextuais, após representadas, são armazenadas no repositório.

```

<?xml version="1.0" encoding="UTF-8"?>
<contextProfile id="0001">
  <contextType name="atividade">
    <contextEntity name="viagem">
      <contextAttribute name="nome">Minha primeira viagem</contextAttribute>
      <contextAttribute name="destino">Gramado</contextAttribute>
      <contextAttribute name="inicio">2010-03-12</contextAttribute>
      <contextAttribute name="fim">2010-03-21</contextAttribute>
    </contextEntity>
  </contextType>
  <contextType name="estabelecimento">
    <contextEntity name="hospedagem">
      <contextAttribute name="nome">Hotel Serra Azul</contextAttribute>
      <contextAttribute name="tipo">Hotel</contextAttribute>
      <contextAttribute name="endereco">R. Garibaldi, 152</contextAttribute>
      <contextAttribute name="cidade">Gramado</contextAttribute>
      <contextAttribute name="estado">RS</contextAttribute>
      <contextAttribute name="nivel">4 estrelas</contextAttribute>
    </contextEntity>
  </contextType>
</contextProfile>

```

Figura 6.5: Representação de informações contextuais

No *Contexttrip* os detalhes sobre a viagem adicionadas podem ser visualizadas. Quando o usuário solicitar, o cliente *Web service* faz chamadas por meio do objeto local *stub* para o serviço remoto, conforme apresenta a Figura 6.6. Dessa forma, esse cliente faz uma requisição ao serviço disponível *ApresentaContexto*, e em seguida o *ApresentarResponse* recebe a resposta da solicitação.

```

.
.
.
ApresentaContextoStub stub = new ApresentaContextoStub();
ApresentaContextoStub.ApresentarResponse res = stub.apresentar();
resposta = res.get_return();
.
.
.

```

Figura 6.6: Apresentação de informações contextuais

Esse serviço extrai as informações contextuais por meio do *Mecanismo de apresentação*. Assim, as informações, obtidas do *Repositório*, são processadas pelo *Suporte ao processamento* e enviadas ao serviço do *Mecanismo de apresentação*.

6.5. Resultados

O trabalho detalhado na seção anterior mostrou que as aplicações sensíveis ao contexto podem se beneficiar das características da arquitetura orientada a serviços para a aquisição de informações contextuais. Por meio de *Web services* há o intercâmbio de informações de contexto na plataforma *Web*.

Aplicações de terceiros podem cooperar no compartilhamento de informações, desta forma, deve-se considerar como promover o intercâmbio de informações de contexto, como garantir a padronização na representação e na comunicação com aplicações e serviços. A arquitetura orientada a serviços apoia, devido ao reuso e fraco acoplamento, a aquisição de informações de contexto de aplicações de terceiros por aplicações sensíveis ao contexto, provendo flexibilidade e interoperabilidade.

Sendo assim, as características proveniente das especificações e tecnologias envolvidas, colocam a arquitetura orientada a serviços como uma solução interessante em cenários com diversidade de aplicações.

6.6. Considerações Finais

Neste capítulo foi abordado um cenário de exemplo de aplicação. Foram apresentadas a modelagem dos casos de uso e a arquitetura do protótipo. Na sequência, foi abordada a implementação do protótipo, demonstrando alguns serviços na aquisição de informações contextuais. O capítulo seguinte apresenta as conclusões, contribuições e trabalhos futuros.

7. CONCLUSÃO

O objetivo desta monografia foi investigar a utilização da arquitetura orientada a serviços em aplicações *Web* sensíveis ao contexto. Essa arquitetura foi aplicada para obter informações de contexto provenientes de diversos sistemas heterogêneos por meio de serviços. As características, como o fraco acoplamento, da arquitetura orientada a serviços proporciona a evolução das aplicações por meio da integração de sistemas.

Desta forma, esta monografia propõe o uso de *Web services* como abordagem para apoiar o compartilhamento de informações de contexto por meio de uma arquitetura padronizada e aberta. Além do mais, *Web services* fornecem interoperabilidade entre aplicações e promovem o compartilhamento de informações em plataformas diferentes e heterogêneas. Assim, a utilização da arquitetura orientada a serviços foi inserida em um cenário de viagens, por meio da integração de uma aplicação sensível ao contexto com outros sistemas de terceiros na plataforma *Web*.

As lições aprendidas na utilização da arquitetura orientada a serviços para a integração e interação entre aplicações é a principal contribuição do trabalho, particularmente em aplicações *Web* sensíveis ao contexto. Esse aprendizado é útil para o desenvolvimento de outros cenários que demandam interação e interoperabilidade entre aplicações heterogêneas, bem como o intercâmbio de informações a serem compartilhadas, que no âmbito de aplicações sensíveis ao contexto possuem grande valor contextual.

7.1. Trabalhos Futuros

A partir dos resultados obtidos, pode-se identificar algumas questões importantes para o aperfeiçoamento e extensão deste trabalho. Seguem algumas sugestões de funcionalidades ainda incompletas ou não implementadas que podem ser realizadas em trabalhos futuros:

- explorar uma abordagem baseada em ontologia para a representação de contexto para este domínio de aplicação;

- o elemento *Suporte ao processamento* poderia ser mais explorado, especialmente do que diz respeito à aplicabilidade de regras de inferência;
- explorar aspectos de segurança e privacidade nas informações contextuais;
- investigar repositórios de *Web services* e estratégias para descoberta e contratação de serviços; e
- incorporar a adaptação de *Web services* de acordo com o contexto.

REFERÊNCIAS

- BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. *Software Architecture in Practice*, 2. ed. Boston: Addison Wesley, 2003.
- BOOCH, Grady. The architecture of Web applications. *DeveloperWorks*, jun. 2001.
- BRAMBILLA, Marco; COMAI, Sara; FRATERNALI, Piero. Hypertext Semantics for Web Applications. *SEBD Italian National Conference on DataBase Systems*. 2002.
- BRÉZILLON, P.; POMEROL, J. C. Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain*. 1999.
- BROWN, P. J. The stick-e document: a framework for creating context-aware applications. *Electronic Publishing*, v.8, p. 259-272, jun/set. 1996.
- BROWN, Peter J.; BOVEY, John D.; CHEN, Xian. Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications*. 1997.
- CERAMI, Ethan. *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*. O'Reilly, 2002.
- CERI, Stefano; FRATERNALI, Piero; BONGIO, Aldo. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. *9th international World Wide Web Conference, Elsevier*. 2000.
- CHAARI, T.; LAFOREST, F.; CELENTANO, A. Design of Context-Aware Applications Based on Web Services. *Technical Report CS-2004-5*. 2004.
- CHAVES, Ana Paula. *Um Modelo Baseado em Context-awareness para Disseminação de Informações em um Ambiente de Desenvolvimento Distribuído de Software*. Maringá, julho, 2009. p. 149. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Maringá.
- CONALLEN, Jim. *Building Web Applications with UML*. 2. ed. Boston: Addison Wesley, 2002.
- CONALLEN, Jim. Modeling Web Application Architectures with UML. *Communications of the ACM*, v.42, n.10, p. 63-70, out. 1999.
- COSTA, Patrícia Dockhorn. *Towards a Services Platform for Context-Aware Applications*. Enschede, The Netherlands, agosto, 2003. p. 112. Dissertação (Mestrado em Ciência da Computação) - University of Twente.
- DEY, Anind K. Context-Aware Computing: The CyberDesk Project. *AAAI 1998 Spring Symposium on Intelligent Environments*, Technical Report SS-98-02. 1998.
- DEY, Anind K. Understanding and Using Context. *Personal and Ubiquitous Computing*, London, p. 4-7. 2001.

DEY, Anind K. *Providing Architectural Support for Building Context-Aware Applications*. Georgia, novembro, 2000. p. 188. Tese (Doutorado em Ciência da Computação) - Georgia Institute of Technology.

DEY, Anind K.; ABOWD, Gregory D. Towards a Better Understanding of Context and Context-Awareness. *Proceedings of the What, Who, Where, When, and How of Context-Awareness Workshop, CHI 2000 Conference on Human Factors in Computer Systems*, New York. 2000.

DEY, Anind K.; ABOWD, Gregory D.; SALBER, Daniel. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, v.16, p. 97-166. 2001.

DEY, Anind K.; ABOWD, Gregory D.; WOOD, Andrew. CyberDesk: a framework for providing self-integrating context-aware services. *Knowledge-Based Systems 11*, p. 3-13. 1998.

ERL, Thomas. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

GINIGE, Athula; MURUGESAN, San. Web Engineering: An Introduction. *IEEE Multimedia*, p. 14-18. 2001.

GRIFFITHS, G.; HEBBRON, B. D.; LOCKYER, M. A.; OATES, B. J. A Simple Method & Tool for Web Engineering. *SEKE'02: 14th international conference on Software engineering and knowledge engineering*. ACM Press. 2002.

HELD, Albert; BUCHHOLZ, Sven; SCHILL, Alexander. Modeling of Context Information for Pervasive Computing Applications. *6th World Multi-conference on Systemics, Cybernetics and Informatics*. 2002.

HOLCK, Jesper. 4 Perspectives on Web Information Systems. *36th Hawaii International Conference on System Sciences*. 2003.

JOSUTTIS, Nicolai M. *SOA in Practice*. O'Reilly Media, 2007.

KREGGER, Heather. Web Services Conceptual Architecture. *IBM Software Group*. 2001.

KOCH, Nora Parcus. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. Munich, dezembro, 2000. p. 371. Tese (Doutorado em Ciência da Computação) – Ludwig-Maximilians University.

MOSTÉFAOUI, Ghuita Kouadri; PASQUIER-ROCHA, Jacques; BRÉZILLON, Patrick. Context-Aware Computing: A Guide for the Pervasive Computing Community. *IEEE Network*. 2004.

MURUGESAN, San; DESHPANDE, Yogesh; HANSEN, Steve; GINIGE, Athula. Web Engineering: A New Discipline for Development of Web-Based Systems. *WebEngineering 2000, LNCS 2016*. 2001.

- MURUGESAN, San; GINIGE, Athula. Web Engineering: Introduction and Perspectives. *Idea Group Publishing*. 2005.
- NEWCOMER, Eric. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, 2002.
- PAPAZOGLU, Mike P.; HEUVEL, Willem-Jan van den. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*. 2007.
- PASCOE, Jason. Adding Generic Contextual Capabilities to Wearable Computers. *2nd International Symposium on Wearable Computers*. 1998.
- PRESSMAN, Roger S. *Engenharia de Software*. 6. ed. São Paulo: McGraw-Hill, 2006.
- RYAN, Nick; PASCOE, Jason; MORSE, David. Enhanced Reality Fieldwork: the Context Aware Archaeological Assistant. *Computer Applications in Archaeology*. 1997.
- SALBER, D.; DEY, A, K.; ABOWD, G. D. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. *GVU Technical Report GIT-GVU-98-01*. 1998.
- SCHILIT, Bill N.; ADAMS, Norman; WANT, Roy. Context-Aware Computing Applications. *IEEE Workshop on Mobile Computing Systems and Applications*. 1994.
- SCHILIT, Bill N.; THEIMER, Marvin M. Disseminating Active Map Information to Mobile Hosts. *IEEE Network*. 1994.
- STRANG, Thomas; LINNHOF-POPIEN, Claudia. A Context Modeling Survey. *Workshop on Advanced Context Modelling, Reasoning and Management, in 6th International Conference on Ubiquitous Computing*. 2004.
- VIEIRA, Vaninha. *CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive System*. Recife, outubro, 2008. p. 187. Tese (Doutorado em Ciência da Computação) – Universidade Federal de Pernambuco.
- W3C. Extensible Markup Language. Obtido via Internet: <http://www.w3.org/XML>. Acesso em: nov. 2009.
- WARD, Andy; JONES, Alan; HOPPER, Andy. A New Location Technique for the Active Office. *IEEE Personal Communications*. 1997.
- YOON, Hoijin. A Convergence of Context-Awareness and Service-Oriented Computing in Ubiquitous Computing. *International Journal of Computer Science and Network Security*, v.7, n.3, p. 253-257, mar. 2007.

APÊNDICE

APÊNDICE A – MODELAGEM

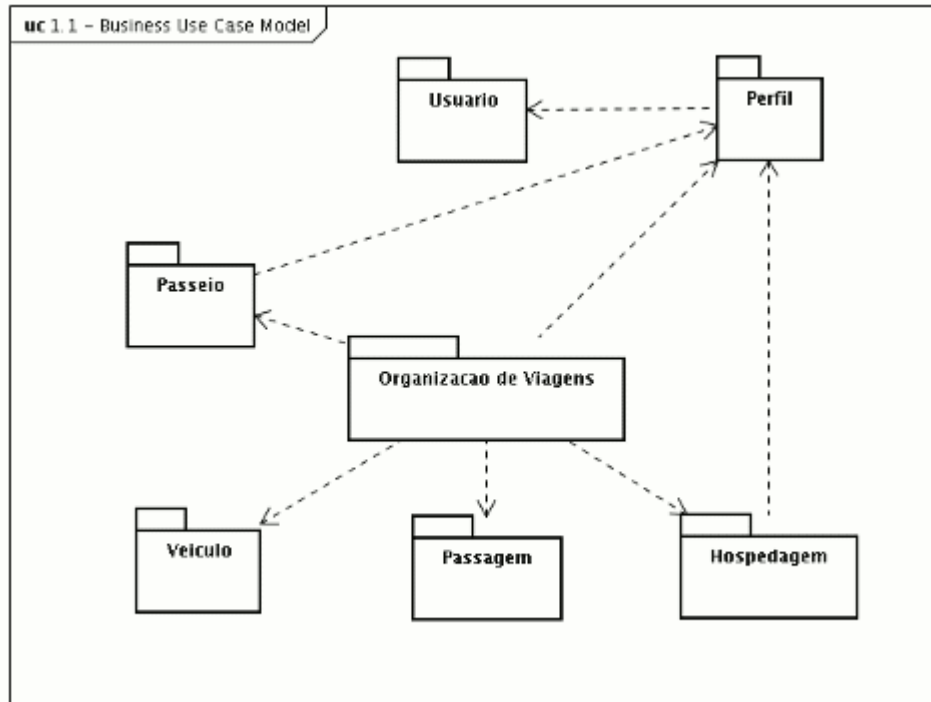


Figura A1: Visão de Negócio

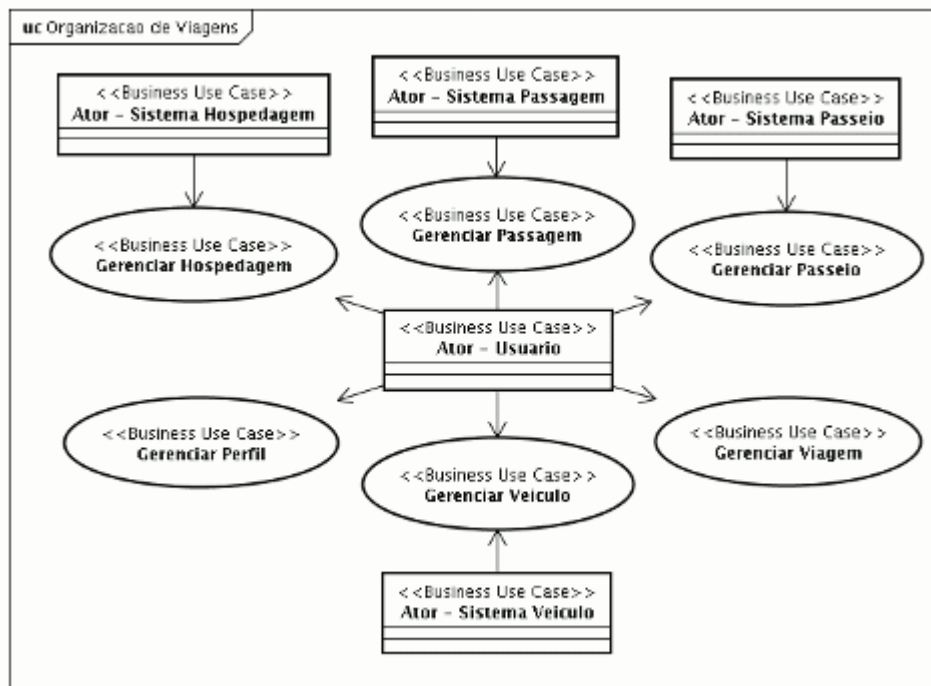


Figura A2: Modelo de Negócio para Organização de Viagem

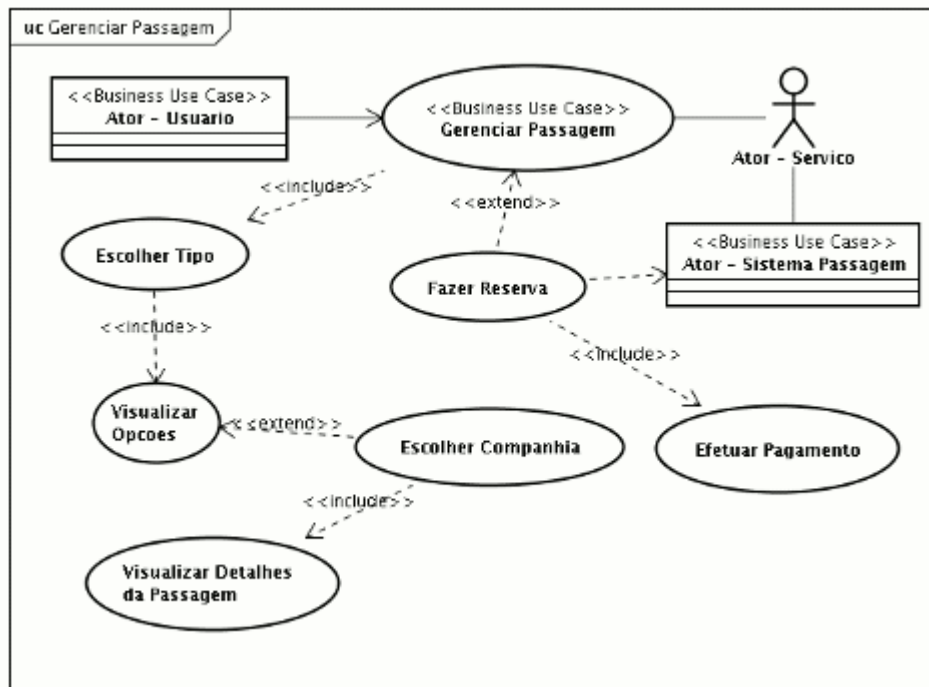


Figura A5: Gerenciar Passagem Use Case

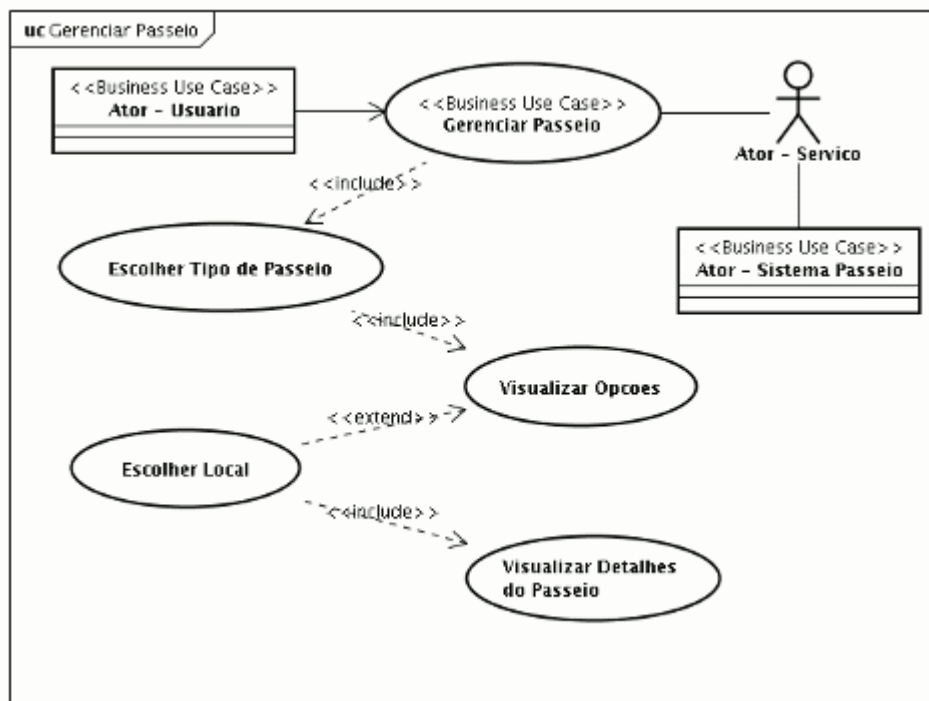


Figura A6: Gerenciar Passeio Use Case

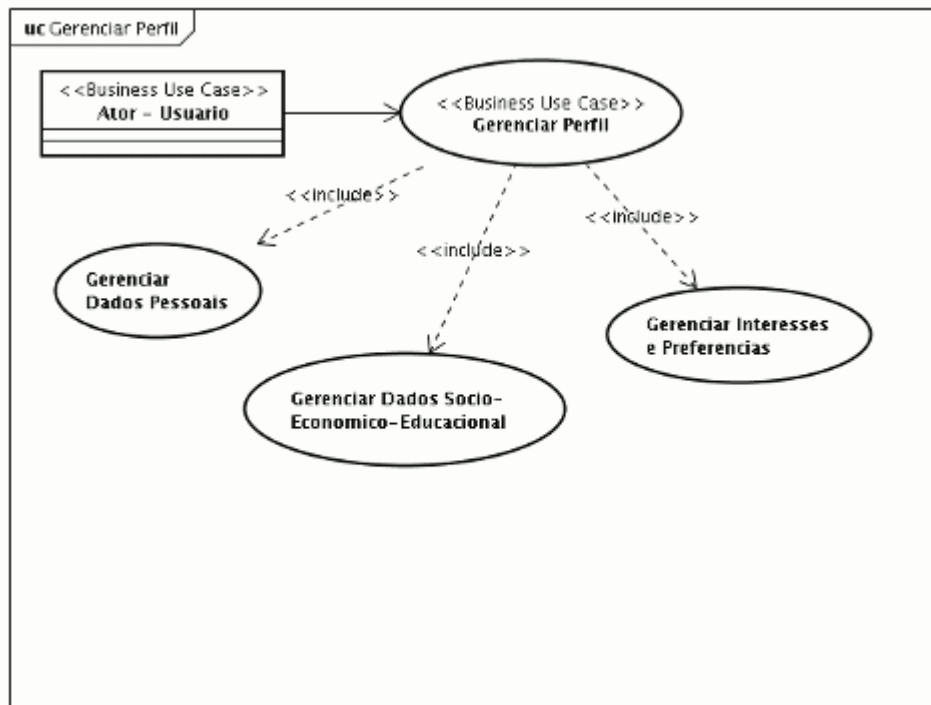


Figura A7: Gerenciar Perfil Use Case

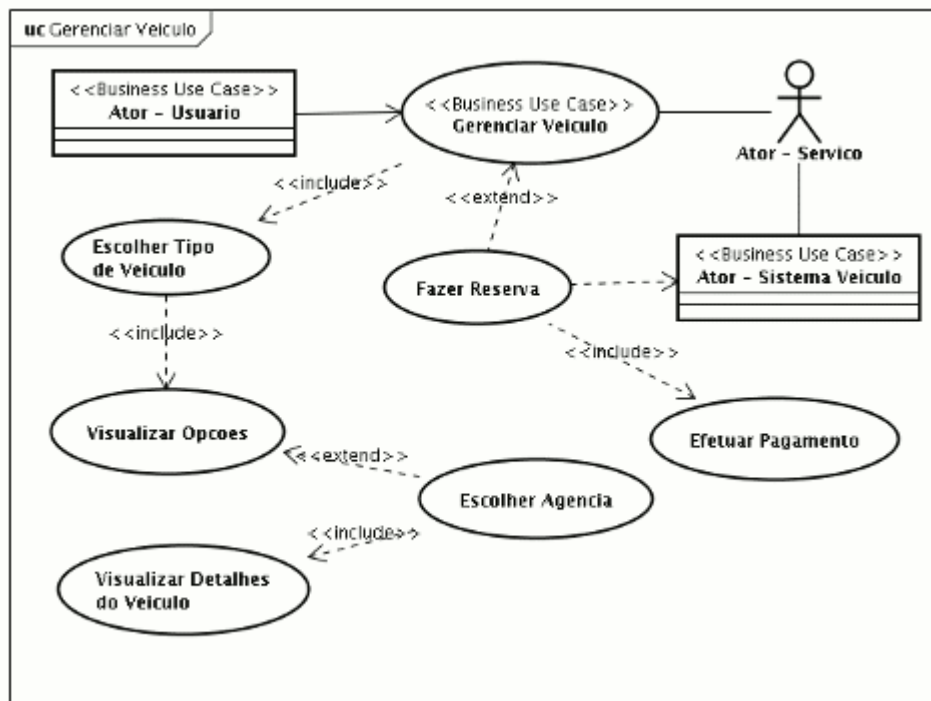


Figura A8: Gerenciar Veículo Use Case

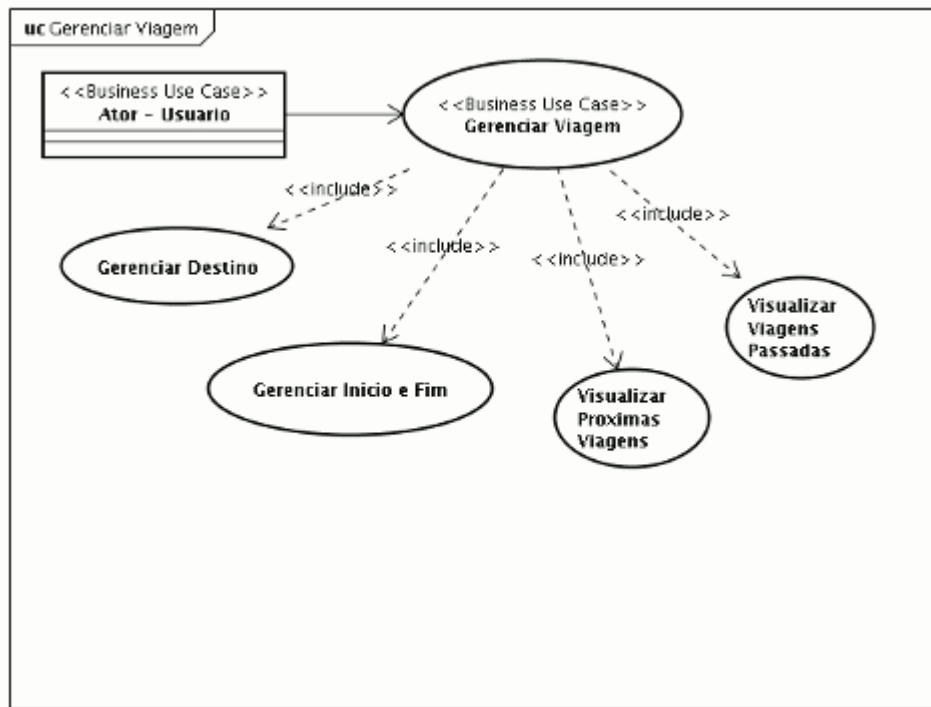


Figura A9: Gerenciar Viagem Use Case

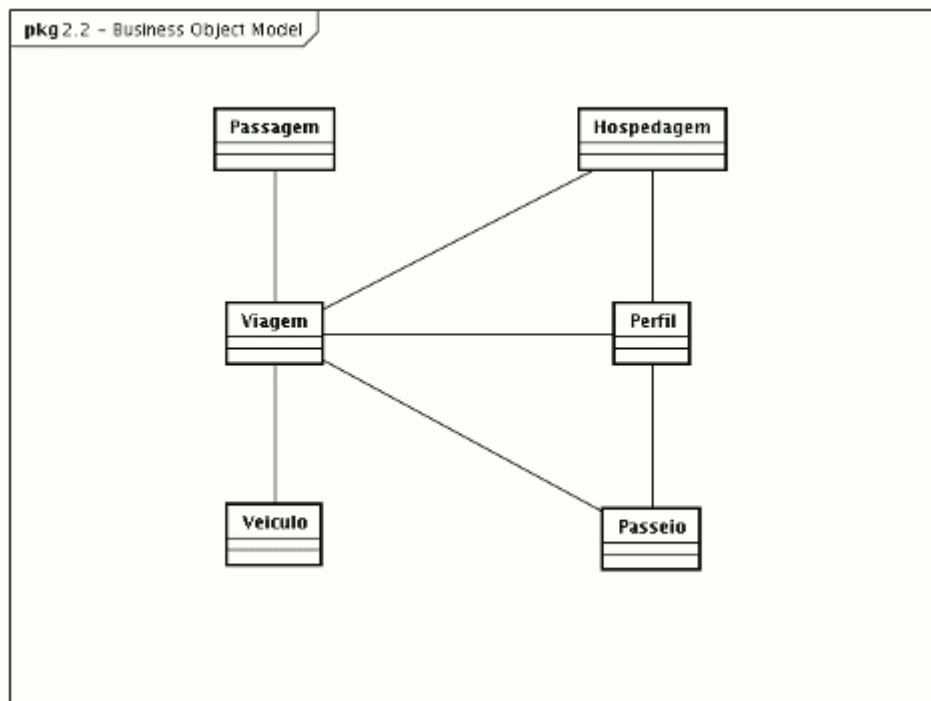


Figura A10: Modelo de Objeto de Negócio

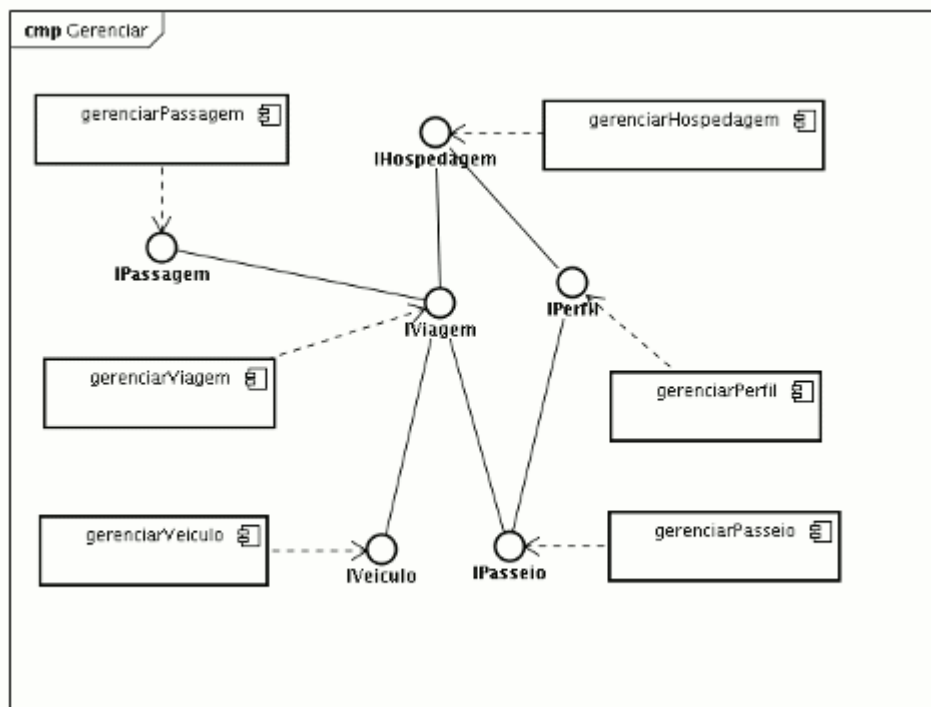


Figura A11: Diagrama de Componente

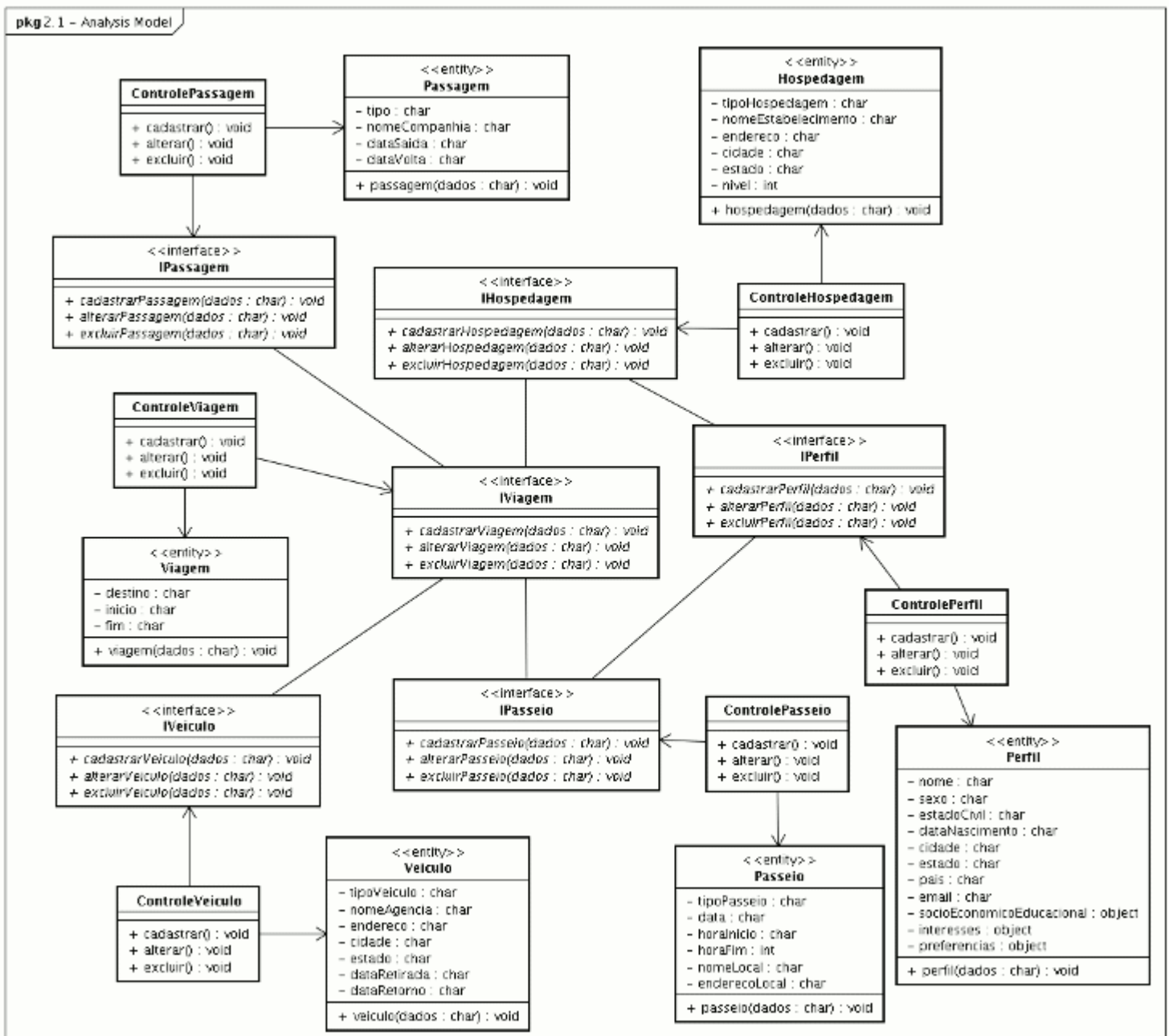


Figura A12: Diagrama de Classes Análise

