Um processo de software e um modelo
ontológico para apoio ao desenvolvimento de
aplicações sensíveis a contexto

Renato de Freitas Bulcão Neto

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 16.11.2006

Assinatura: _____

Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto

Renato de Freitas Bulcão Neto

Orientadora: *Profa. Dra. Maria da Graça Campos Pimentel*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional.

USP – São Carlos
Novembro/2006

Dedicatória

*Aos meus pais Luís e Ilca, por terem me dado a luz da vida;
às minhas irmãs Renatha e Roberta, por me darem força para lutar;
ao meu futuro sobrinho, que rezo para nascer com saúde;
à minha orientadora Graça, pela oportunidade de chegar a este momento;
aos meus amigos, por terem colaborado com minha formação;
e à minha esposa Taciana, pelo seu amor e apoio incondicionais.*

Agradecimentos

Agradeço à professora e orientadora Graça Pimentel pelos ensinamentos passados ao longo de todos esses anos. Mostrou-me o “caminho das pedras” para várias situações de vida em que me encontrei: nos estudos, na vida profissional e mesmo na vida pessoal. Agradeço-te de coração por tornar-me alguém melhor que fui ontem, e também por preparar-me para ser amanhã alguém melhor que sou hoje.

Agradeço aos meus pais por todo o esforço que fizeram para que eu atingisse os meus objetivos. Obrigado por acreditarem em mim! Sempre!

Agradeço à minha esposa, Taciana, que me acompanhou por toda a longa jornada do doutorado, que sofreu junto comigo quando tudo parecia nebuloso. Agradeço-te por também ter estado comigo quando colhi vitórias em cada artigo aceito, em cada auxílio financeiro concedido, em cada capítulo escrito.

Agradeço aos meus amigos, colegas e ex-colegas do ICMC-USP pela ajuda, incentivo e companhia: Graça Pimentel, Rudinei Goularte, Alessandra Macedo, Dilvan Moreira, Édson Moreira, Renata Pontin, Rodrigo Mello, José “Toño” Camacho, Carlos “Patrão” Jardim, Carlos Arruda “Juninho”, Carlos “Billy” Rocha, Otávio, Elaine, Débora, Flávia, Luciana, Cláudia Mello, Renata Porto, Daniel Lobato, Renan, Valter, Hélder, Anselmo, Juliana, Pedro, Jane, Silvana, Claudia Izeki, Laércio, Izabella, Marcela, Felipe, Matheus, Mário e Cássio.

Agradeço aos colegas César Teixeira, Elizete e Daniel Pires do núcleo UFSCar e Faculdades COC, bem como aos alunos das Faculdades COC das turmas de Engenharia de Software, Qualidade de Software e Informática e Sociedade por terem compartilhado comigo este momento.

Aos meus amigos de “república” João Carlos e Mário Meireles pelos momentos de diversão, fraternidade e eterna amizade.

À colônia maranhense que passou (ou tem passado) longas férias em São Carlos: Omar Cortês, Érika Höhn, Nilson Costa, Fernando Tanaka e Bruno Feres.

Aos amigos que fiz na *Hewlett Packard* em Bristol, Inglaterra: Steve Battle, Yathiraj Udupi, Javier Esplugas, Anastasia Krithara, Viral Parekh e Nazareno de Andrade.

Agradeço à FAPEMA pelo apoio financeiro às minhas pesquisas (nº 03/345).

Por último, porém não menos importante, quero agradecer a Deus pela força a mim concedida para enfrentar todas as dificuldades que encontrei.

Resumo

Aplicações sensíveis a contexto utilizam informações de contexto para fornecer serviços adaptados a usuários na realização de suas tarefas. Informação de contexto é qualquer informação considerada relevante para caracterizar entidades de uma interação usuário-computador, como a identidade e a localização de usuários. Esta tese trata a carência de uma abordagem que considere, em termos de processo de software, a complexidade de desenvolvimento de software sensível a contexto. O problema em questão é tratado por meio de três linhas de investigação: modelagem de informação contextual, serviços para tratamento de informação contextual e processo de software para computação sensível a contexto. As contribuições desta tese incluem: (i) o processo de software *POCAp* (*Process for Ontological Context-aware Applications*) para apoiar a construção de aplicações sensíveis a contexto baseadas em ontologias; (ii) o modelo de informações de contexto *SeCoM* (*Semantic Context Model*) baseado em ontologias e em padrões da Web Semântica; (iii) a infra-estrutura de serviços configuráveis *SCK* (*Semantic Context Kernel*) para interpretar informações de contexto apoiadas por modelos ontológicos de informação contextual, como o modelo *SeCoM*; (iv) uma instanciación do processo *POCAp* correspondente à extensão de uma aplicação com informações de contexto apoiadas pelo modelo *SeCoM*, e sua integração com serviços da infra-estrutura *SCK*; e (v) a identificação de questões de projeto relacionadas à inferência sobre informação contextual ontológica.

Abstract

In order to provide adaptive services according to users' tasks, context-aware applications exploit context information, which is any information used to characterize entities of a user-computer interaction such as user identity or user location. This thesis deals with the lack of a software process-based approach to supporting the inherent complexity of developing context-aware systems. The work reported in this thesis followed three main lines of investigation: context information modeling, services for processing context information, and a software process for context-aware computing. The contributions of this thesis include: (i) the Process for Ontological Context-aware Applications (*POCAp*) to support the development of context-aware applications based on ontologies; (ii) the Semantic Context Model (*SeCoM*) based on Semantic Web standards and ontologies; (iii) the Semantic Context Kernel (*SCK*) services infrastructure for interpreting ontological context information models such as the *SeCoM* model; (iv) an implementation of the *POCAp* process for the extension of an application with context information based on the *SeCoM* model, and its integration with services of the *SCK* infrastructure; and (v) the identification of design issues related to the inference over ontology-based context information.

Publicações

A seguir é apresentada a lista de publicações originadas a partir deste trabalho:

Capítulos de Livro

- **Bulcão Neto, R. F.**, Prazeres, C. V. S., and Pimentel, M. G. C. (2006). Web Semântica: Teoria e Prática. Cap. 2, pp. 47–86. Tópicos em Sistemas Interativos e Colaborativos. SBC Press. Texto referente a mini-curso ministrado no Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06), Natal-RN, Brasil.

Artigos completos em Conferência Internacional (com *referee*)

- **Bulcão Neto, R. F.**, Teixeira, C. A. C., and Pimentel, M. G. C. (2005). A Semantic Web-based infrastructure for supporting context-aware applications. In *Proceedings of the IFIP International Conference on Embedded and Ubiquitous Computing (EUC'05)*, LNCS 3824, pp. 900–909, Nagasaki, Japan. Springer. http://dx.doi.org/10.1007/11596356_89.
- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2005). Toward a domain-independent semantic model for context-aware computing. In *Proceedings of the 3rd Latin American Web Congress (LA-Web'05)*, pp. 61–70, Buenos Aires, Argentina. IEEE CS Press. <http://dx.doi.org/10.1109/LAWEB.2005.43>.

Artigos completos em Conferência Nacional (com *referee*)

- **Bulcão Neto, R. F.**, Kudo, T. N., and Pimentel, M. G. C. (2006). Using a software process for ontology-based context-aware computing: A case study. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06)*, Natal-RN, Brasil. ACM Press. 10 páginas.
- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2006). Performance evaluation of inference services for ubiquitous computing. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06)*, Natal-RN, Brasil. ACM Press. 8 páginas.

- **Bulcão Neto, R. F.**, Macedo, A. A., Camacho-Guerrero, J. A., and Pimentel, M. G. C. (2005). Configurable semantic services leveraging applications context-aware. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'05)*, Poços de Caldas-MG, Brasil, ACM Press. 9 páginas. <http://doi.acm.org/10.1145/1114223.1114233>.
- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2003). Interoperabilidade semântica entre aplicações cientes de contexto. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'03)*, pp. 371–385, Salvador-BA, Brasil. <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianConferencePapers/BulcaoPimentel-Webmedia-2003.pdf>.

Artigos resumidos em Conferência Nacional (com referee)

- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2003). Interoperabilidade semântica entre aplicações cientes de contexto: Uma abordagem ontológica. In *Anais do Workshop de Teses e Dissertações, Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'03)*, pp. 577–580, Salvador-BA, Brasil. <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianWorkshopPapers/BulcaoPimentel-WkspTeses-2003.pdf>.

Artigos resumidos em Workshop Local (sem referee)

- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2004). Explorando conceitos de Web Semântica em Computação ciente de contexto. In *Workshop de Ontologias*, ICMC-USP, São Carlos-SP, Brasil. http://tidia-ae.incubadora.fapesp.br/portal/publications/other_documents/BulcaoPimentel-WkspOnto-2004.pdf.

Relatórios Técnicos

- **Bulcão Neto, R. F.**, Kudo, T. N., and Pimentel, M. G. C. (2006). POCAp: A software process for ontology-based context-aware applications. In *Relatório Técnico 273*, ICMC-USP, São Carlos-SP, Brasil. 16 páginas.
- **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2006). Performance evaluation of ubiquitous inference services: reasoning-related issues. In *Relatório Técnico 272*, ICMC-USP, São Carlos-SP, Brasil. 17 páginas.

A seguir é apresentada a lista de publicações indiretamente relacionadas ao contexto deste trabalho:

Artigos completos em Periódico Internacional (com referee)

- Jardim, C. H. O., **Bulcão Neto, R. F.**, Godoy, R. P., Ribas, H. M. B., Arruda Jr., C. R. E., Munson, E. V. and Pimentel, M. G. C. (2005). Web Services enabling ubiquitous computing applications: Lessons learned by integrating ubiquitous e-learning applications. In *International Journal of Web Services Practices*, 1(1-2):142-152. ISSN: 1738-6535. <http://nwesp.org/ijwsp/2005/vol1/13.pdf>.

Artigos completos em Conferência Internacional (com referee)

- Jardim, C. H. O., **Bulcão Neto, R. F.**, Ribas, H. M. B., Munson, E. V. and Pimentel, M. G. C. (2005). Web Services enabling context-aware applications: Lessons learned by integrating e-learning applications. In *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05)*, pp. 400-405, Seoul, Korea. IEEE CS Press. <http://dx.doi.org/10.1109/NWESP.2005.85>.
- Macedo, A. A., **Bulcão Neto, R. F.**, Camacho-Guerrero, J. A., Jardim, C. H. O., Cattelan, R. G., Inácio Jr, V. R. and Pimentel, M. G. C. (2005). Linking everyday presentations through context information. In *Proceedings of the 3rd Latin American Web Conference (LA-Web'05)*, pp. 130-139, Buenos Aires, Argentina. IEEE CS Press. <http://dx.doi.org/10.1109/LAWEB.2005.21>.
- **Bulcão Neto, R. F.**, Jardim, C. H. O., Camacho-Guerrero, J. A. and Pimentel, M. G. C. (2004). A Web Service approach for providing context information to CSCW applications. In *Proceedings of 2nd Latin American Web Congress (LA-Web'04)*, pp. 78-85, Ribeirão Preto-SP, Brazil. IEEE CS Press. <http://dx.doi.org/10.1109/WEBMED.2004.1348145>.

Artigos completos em Conferência Nacional (com referee)

- **Bulcão Neto, R. F.**, Jardim, C. H. O., Camacho-Guerrero, J. A., Lobato, D. C. and Pimentel, M. G. C. (2004). A context-based Web Service approach to communities of practice. In *Anais do XXXI Seminário Integrado de Software e Hardware, (SEMISH'04)*, Salvador-BA, Brasil. 15 páginas. <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianConferencePapers/BulcaoEtAl-SEMISH-2004.pdf>.

Artigos completos em Workshop Internacional (com referee)

- Arruda Jr., C. R. E., **Bulcão Neto, R. F.** and Pimentel, M. G. C. (2003). Open context-aware storage as a Web Service. In *Proceedings of the International Workshop on Middleware for Pervasive and Ad-Hoc Computing in conjunction with the ACM/IFIP/USENIX International Middleware Conference (Middleware'03)*, pp. 81–87, Rio de Janeiro-RJ, Brazil. <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedInternationalWorkshopPapers/ArrudaJrEtAl-MPAC-2003.pdf>.

Artigos resumidos em Workshop Internacional (com referee)

- **Bulcão Neto, R. F.**, Udupi, Y. B. and Battle, S. (2004). Agent-based mediation in Semantic Web Service Framework. In *Proceedings of First AKT Workshop on Semantic Web Services (AKT-SWS'04)*, pp. 5–8, Milton Keynes, United Kingdom. CEUR-WS. <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-122/paper2.pdf>.

Sumário

1	Introdução	1
1.1	Motivação	4
1.2	Objetivos	6
1.3	Metodologia	8
1.4	Contribuições	11
1.5	Estrutura da tese	12
2	Computação Sensível a Contexto	15
2.1	Conceitos básicos	16
2.2	Características de aplicações sensíveis a contexto	17
2.3	Dimensões semânticas de informação de contexto	18
2.4	Requisitos para construção de software sensível a contexto	21
2.4.1	Especificação de informações de contexto	21
2.4.2	Separar aquisição de utilização de informações de contexto	21
2.4.3	Interpretação de informações de contexto	22
2.4.4	Comunicação distribuída e transparente	22
2.4.5	Disponibilidade contínua de componentes de captura de informações de contexto	22
2.4.6	Armazenamento de informações de contexto	23
2.4.7	Descoberta de recursos	23
2.5	Exemplos de aplicações sensíveis a contexto	23
2.5.1	Conference Assistant	24
2.5.2	CARE TM	25
2.5.3	Co-occupant Awareness	26
2.5.4	IM4Sports	27
2.5.5	ContextContacts	28
2.5.6	Friend Locator	29

2.6	Considerações finais	30
3	Web Semântica	31
3.1	Padrões de metadados	32
3.1.1	Padrão vCard	33
3.1.2	Padrão iCalendar	33
3.1.3	Padrão Dublin Core	34
3.2	Ontologias	34
3.2.1	Vantagens do uso de ontologias	35
3.2.2	Engenharia de ontologias	36
3.3	Arquitetura da Web Semântica	42
3.3.1	Camada básica de dados	42
3.3.2	Camada de descrição sintática	43
3.3.3	Camada de descrição estrutural e semântica	43
3.3.4	Camada de descrição semântica e lógica	45
3.3.5	Camada de descrição lógica	49
3.3.6	Camada de prova e confiança	51
3.4	Ontologias da Web Semântica	52
3.4.1	SUMO	52
3.4.2	OpenCyc	52
3.4.3	FOAF	53
3.4.4	SWEET	53
3.4.5	CC/PP	53
3.4.6	OWL-Time	54
3.5	Aplicações da Web Semântica	54
3.5.1	Swoggle	55
3.5.2	SWOOP	56
3.5.3	Photocopain	57
3.5.4	Semantic Wikipedia	58
3.6	Considerações finais	59
4	Modelo Semântico de Informações de Contexto	61
4.1	Visão geral do modelo SeCoM	62
4.2	Ontologia Time	64
4.2.1	Descrição semântica	65
4.2.2	Metodologia de desenvolvimento	67
4.3	Ontologia Temporal Event	68
4.3.1	Descrição semântica	69
4.3.2	Metodologia de desenvolvimento	69
4.4	Ontologia Spatial	70

4.4.1	Descrição semântica	70
4.4.2	Metodologia de desenvolvimento	72
4.5	Ontologia Spatial Event	73
4.5.1	Descrição semântica	73
4.5.2	Metodologia de desenvolvimento	74
4.6	Ontologia Actor	74
4.6.1	Descrição semântica	75
4.6.2	Metodologia de desenvolvimento	76
4.7	Ontologia Device	76
4.7.1	Descrição semântica	77
4.7.2	Metodologia de desenvolvimento	82
4.8	Ontologia Activity	83
4.8.1	Descrição semântica	83
4.8.2	Metodologia de desenvolvimento	85
4.9	Avaliação do modelo SeCoM	86
4.10	Considerações finais	88
5	Serviços para Semântica de Informações de Contexto	91
5.1	Diretrizes de projeto	92
5.2	Arquitetura da infra-estrutura SCK	93
5.3	Implementação dos serviços da infra-estrutura SCK	95
5.3.1	Serviço de Armazenamento de Contexto	96
5.3.2	Serviço de Consulta de Contexto	99
5.3.3	Serviço de Inferência de Contexto	101
5.4	Avaliação do serviço de inferência de contexto	106
5.4.1	Configuração do experimento	106
5.4.2	Ontologias SeCoM como dados de teste	106
5.5	Considerações finais	109
6	Um Processo para Aplicações Sensíveis a Contexto	111
6.1	Uma visão geral do processo POCAp	112
6.2	Atividade de análise e especificação (a1)	114
6.2.1	Análise e especificação de requisitos (a1.1)	114
6.2.2	Análise e especificação de informação contextual (a1.2)	115
6.2.3	Análise e especificação de reúso do modelo (a1.3)	115
6.2.4	Análise e especificação de extensão do modelo (a1.4)	116
6.3	Atividade de projeto (a2)	116
6.3.1	Projeto de reúso de serviços (a2.1)	117
6.3.2	Projeto de extensão de serviços (a2.2)	118
6.3.3	Projeto de novos serviços (a2.3)	118

6.4	Atividade de desenvolvimento (a3)	118
6.5	Atividade de verificação e validação (a4)	119
6.6	Considerações finais	120
7	Instanciação do Processo <i>POCAp</i>	123
7.1	Os artefatos SeCoM e SCK no processo <i>POCAp</i>	123
7.2	Aplicação WebMemex: Estudo de caso do processo <i>POCAp</i>	126
7.2.1	Aplicação WebMemex	126
7.2.2	Atividade de análise e especificação (a1)	127
7.2.3	Atividade de projeto (a2)	133
7.2.4	Atividade de desenvolvimento (a3)	136
7.2.5	Atividade de verificação e validação (a4)	138
7.3	Considerações finais	141
8	Trabalhos Relacionados	143
8.1	Context Toolkit	144
8.2	ConFab	146
8.3	Gaia	149
8.4	one.world	151
8.5	Cooltown	154
8.6	QSI	156
8.7	CoBrA	161
8.8	SOCAM	164
8.9	Comparação com o trabalho proposto	168
8.9.1	Comparação com a infra-estrutura SCK	168
8.9.2	Comparação com o modelo SeCoM	170
8.9.3	Comparação com o processo <i>POCAp</i>	172
8.10	Considerações finais	173
9	Conclusão	175
9.1	Problemas	175
9.1.1	Modelagem de informação contextual	175
9.1.2	Serviços para tratamento de informação contextual	176
9.1.3	Processo de software para computação sensível a contexto	176
9.2	Contribuições	177
9.2.1	Modelo <i>SeCoM</i> (Modelo de Contexto Semântico)	177
9.2.2	Infra-estrutura <i>SCK</i> (Núcleo de Contexto Semântico)	178
9.2.3	Processo <i>POCAp</i> (Processo para Aplicações Sensíveis a Contexto Baseadas em Ontologias)	178
9.3	Limitações	179

9.3.1	Limitações do modelo <i>SeCoM</i>	179
9.3.2	Limitações da infra-estrutura <i>SCK</i>	180
9.3.3	Limitações do processo <i>POCAp</i>	182
9.4	Trabalhos futuros	182
A	Ontologias de Apoio do Modelo SeCoM	185
A.1	Ontologia Contact	185
A.2	Ontologia Role	187
A.3	Ontologia Relationship	187
A.4	Ontologia Knowledge	188
A.5	Ontologia Document	189
A.6	Ontologia Project	191
A.7	Avaliação das ontologias de apoio	192
A.8	Considerações finais	193

Lista de Figuras

2.1	(a) Mapa da residência da <i>Elite Care</i> com destaque para a localização de residentes com temperatura corpórea acima do normal. (b) Gráficos relacionados às atividades de um residente que servem para diagnóstico de sua saúde física e mental. Adaptado de [Elite Care, 2006].	25
2.2	(a) Mapa com a localização de pessoas em um prédio de um campus universitário. (b) Janela com as informações de contato das pessoas que se encontram na mesma sala que o usuário corrente, no caso <i>John Zee</i> [Heer et al., 2003a].	26
2.3	As diferentes fases de utilização da aplicação <i>IM4Sports</i> incluem um programa de treinamento ou preparação, a seleção de músicas e o <i>playback</i> personalizado de músicas. Adaptado de [Wijnalda et al., 2005].	27
2.4	(a) Lista de contatos, cada qual com informações de localização passada e atual, status de operação do telefone (ícone de mão à esquerda), pessoas na proximidade e perfil de alarme do telefone. (b) Ao clicar em um contato, o usuário obtém mais detalhes e uma explicação sobre ícones e atalhos do sistema [Raento et al., 2005].	28
2.5	(a) Usuário consultando a aplicação <i>Friend Locator</i> para encontrar um amigo. (b) Interface da aplicação que mostra a localização corrente do usuário (<i>You</i> no mapa) e a que distância seu amigo (<i>David</i>) se encontra (250 metros) na direção de um local chamado <i>Pampas</i> [Olofsson et al., 2006].	29
3.1	Arquitetura da Web Semântica. Adaptado de Berners-Lee [2000].	42
3.2	Declaração RDF sob a representação de grafo.	44
3.3	(a) Interface do sistema <i>Swoogle</i> para a consulta de ontologias. (b) Resultado da consulta com os metadados do documento da ontologia encontrada.	55

3.4	(a) Interface do ambiente <i>SWOOP</i> em que elos de hipertexto permitem edição e navegação entre conceitos de uma ontologia. (b) Mecanismo de validação de tipo de dialeto OWL utilizado na ontologia corrente, classificada como <i>OWL Full</i>	56
3.5	Interface de anotação do sistema <i>Photocopain</i> para permitir que usuários insiram suas próprias anotações [Tuffield et al., 2006].	57
3.6	(a) Interface da enciclopédia <i>Semantic Wikipedia</i> com uma descrição semântica da cidade de Londres. (b) Código-fonte da respectiva descrição na versão original da <i>Wikipedia</i> . (c) Código-fonte da mesma descrição na <i>Semantic Wikipedia</i> . Adaptado de [Völkel et al., 2006].	58
4.1	Visão geral do modelo <i>SeCoM</i> . Setas representam o inter-relacionamento entre ontologias via mecanismo de importação. Ovais escuras representam as ontologias principais do modelo, enquanto que as ovas claras auxiliam na descrição de perfis de atores. Adaptado de Bulcão Neto & Pimentel [2006a].	63
4.2	Ilustração da ontologia Time.	65
4.3	Ilustração da ontologia Temporal Event.	68
4.4	Ilustração da ontologia Spatial.	71
4.5	Ilustração da ontologia Spatial Event.	73
4.6	Ilustração da ontologia Actor.	75
4.7	Ilustração da ontologia Device.	77
4.8	Ilustração das classes para armazenamento secundário de um dispositivo computacional da ontologia Device.	80
4.9	Ilustração das classes para interface de rede de um dispositivo computacional da ontologia Device.	81
4.10	Ilustração das classes para componente de software de um dispositivo computacional da ontologia Device.	82
4.11	Ilustração da ontologia Activity.	84
5.1	Arquitetura da infra-estrutura Semantic Context Kernel. Adaptado de Bulcão Neto et al. [2005b].	93
6.1	O processo de software <i>POCAp</i> . Adaptado de Bulcão Neto et al. [2006b].	113
6.2	A atividade de análise e especificação do processo <i>POCAp</i> . Adaptado de Bulcão Neto et al. [2006b].	115
6.3	A atividade de projeto do processo <i>POCAp</i> . Adaptado de Bulcão Neto et al. [2006b].	117

7.1	A aplicação <i>WebMemex</i> , em primeiro plano, oferece ao usuário corrente a possibilidade de recomendar a página Web, apresentada em segundo plano, a grupos de usuários — representados por uma <i>combo box</i> — por meio do botão <i>Send it!</i> [Bulcão Neto et al., 2005a].	127
7.2	Representação gráfica da ontologia da aplicação <i>WebMemex</i> gerada por um <i>plugin</i> [ezOWL, 2006] do editor de ontologias Protégé [Gennari et al., 2003].	132
7.3	A arquitetura de componentes da aplicação <i>WebMemex</i> integrada aos serviços da infra-estrutura SCK. Sobre cada componente estão as informações de contexto que esse componente gerencia. Adaptado de Bulcão Neto et al. [2005a].	135
7.4	Múltiplas configurações do serviço de inferência de contexto da infra-estrutura SCK sobre diferentes bases de informações de contexto da aplicação <i>WebMemex</i> . Adaptado de Bulcão Neto & Pimentel [2006a]. . .	140
8.1	Interações típicas entre aplicações e componentes da arquitetura do <i>Context Toolkit</i> . Aplicações podem obter informações de contexto de sensores diretamente via widgets, ou após processamento realizado por agregadores ou interpretadores. Adaptado de Dey et al. [2001].	144
8.2	Arquitetura da infra-estrutura de software <i>ConFab</i> : sensores físicos e serviços básicos são distribuídos entre aplicações e a infra-estrutura. Adaptado de Hong & Landay [2001].	147
8.3	Componentes da infra-estrutura <i>Gaia</i> . Adaptado de Román et al. [2002].	150
8.4	Visão geral da arquitetura <i>one.world</i> . Os serviços de base e os serviços de sistema constituem o núcleo da arquitetura <i>one.world</i> , enquanto que aplicações, bibliotecas e utilitários de sistema executam no espaço do usuário. Adaptado de Grimm et al. [2004].	152
8.5	Infra-estrutura de presença na Web do projeto <i>Cooltown</i> . Adaptado de Kindberg et al. [2002].	154
8.6	Arquitetura em camadas da infra-estrutura <i>QSI</i> . Comunicação síncrona é representada por uma seta contínua, enquanto que comunicação assíncrona, por uma seta tracejada. Adaptado de Henriksen & Indulska [2006].	157
8.7	Instância de modelo de contexto na notação da linguagem CML. A relação <i>located near</i> representa uma informação de contexto derivada, enquanto que a relação <i>engaged in</i> tem relação de dependência com a relação <i>located at</i> . Adaptado de Henriksen & Indulska [2004b].	158

8.8	Um processo para a construção de software sensível a contexto. Os artefatos utilizados em cada passo são exibidos entre parênteses. Os passos A3, T2 e T3 podem solicitar reiterações para o passo A2. Adaptado de Henricksen & Indulska [2006].	160
8.9	Arquitetura do <i>middleware</i> CoBrA. O intermediador de contexto adquire informações de contexto de diversas fontes e as combina em um modelo compartilhado entre entidades computacionais de um mesmo espaço físico. Adaptado de Chen et al. [2004b].	162
8.10	Arquitetura da ontologia de nível superior SOUPA. Ontologias específicas podem ser construídas a partir das ontologias que compõem o núcleo SOUPA, cada qual em um diferente espaço de nomes XML. Adaptado de Chen et al. [2004c].	163
8.11	Arquitetura do <i>middleware</i> SOCAM orientado a serviços sensíveis a contexto. Setas mais espessas indicam fluxo de dados; caso contrario, indicam fluxo de controle. Adaptado de Gu et al. [2005].	165
8.12	Hierarquia de classes da ontologia de nível superior CONON. Adaptado de Gu et al. [2004].	167
A.1	Ilustração da ontologia Contact.	186
A.2	Ilustração da ontologia Role.	187
A.3	Ilustração da ontologia Relationship.	188
A.4	Ilustração da ontologia Knowledge.	188
A.5	Ilustração da ontologia Document.	189
A.6	Ilustração da ontologia Project.	191

Lista de Tabelas

3.1 Critérios para escolha de dialeto OWL e sua respectiva complexidade computacional no pior caso. Adaptado de Lacy [2005].	46
4.1 Relações temporais entre indivíduos A e B da classe <i>IntervalThing</i>	66
4.2 Complexidade lógica e caracterização estrutural das ontologias principais do modelo <i>SeCoM</i> . Adaptado de Bulcão Neto & Pimentel [2006a]. . .	86
4.3 Notação para expressividade lógica de ontologias baseadas em Lógica de Descrições [Baader et al., 2003].	87
4.4 Ordem crescente de complexidade computacional das ontologias principais do modelo <i>SeCoM</i>	88
5.1 Tempo médio de sub-processos de inferência sobre ontologias do modelo <i>SeCoM</i> (em ms). Adaptado de Bulcão Neto & Pimentel [2006a].	107
7.1 Caracterização da ontologia da aplicação <i>WebMemex</i> . Adaptado de Bulcão Neto & Pimentel [2006a].	133
7.2 Tempo médio de cada sub-processo de inferência sob a ontologia da aplicação <i>WebMemex</i> (em ms). Adaptado de Bulcão Neto & Pimentel [2006a].	139
8.1 Comparação entre a infra-estrutura <i>SCK</i> e trabalhos relacionados.	169
8.2 Comparação entre o modelo <i>SeCoM</i> e modelos semânticos de informação contextual baseados em ontologias.	172
8.3 Comparação entre o processo <i>POCAp</i> e processos de software sensível a contexto.	173
A.1 Expressividade lógica e caracterização estrutural das ontologias de apoio do modelo <i>SeCoM</i> . Adaptado de Bulcão Neto & Pimentel [2006a].	192

A.2 Tempo médio de sub-processos de inferência sobre ontologias de apoio do modelo *SeCoM* (em ms). Adaptado de Bulcão Neto & Pimentel [2006a]. 193

Lista de Abreviaturas

ABox: *Assertional Box*
ANSI: *American National Standards Institute*
API: *Application Programming Interface*
CC/PP: *Composite Capabilities/Preferences Profile*
CML: *Context Modeling Language*
CoBrA: *Context Broker Architecture*
CONON: *Context Ontology*
CSCW: *Computer-Supported Cooperative Work*
CSL: *Context Specification Language*
DAML: *DARPA Agent Markup Language*
DARPA: *Defense Advanced Research Projects Agency*
DC: *Dublin Core*
DCMI: *Dublin Core Metadata Initiative*
DL: *Description Logics*
DOI: *Digital Object Identifier*
EXIF: *Exchangeable Image File Format*
FOAF: *Friend of a Friend*
GEON: *Geosciences Network*
GPS: *Global Positioning System*
HTML: *Hypertext Markup Language*
HTTP: *Hypertext Transfer Protocol*
IDF: *International DOI Foundation*
IEEE: *Institute of Electrical and Electronics Engineers*
InCA-SERVE: *Infrastructure for Capture and Access - Infrastructure for Storage, Extension, Retrieval and Visualization of Evolutionary Information*
InkML: *Ink Markup Language*
iROS: *Interactive Room Operating System*
J2SE: *Java Standard Edition*
JDBC: *Java Database Connectivity*
ONIONS: *Ontologic Integration Of Naive Sources*
OWL: *Web Ontology Language*

PDA: *Personal Digital Assistant*

POCAp: *Process for Ontological Context-Aware Applications*

Prolog: *Programming Logic*

QSI: *Queensland's software infrastructure*

RDF: *Resource Description Framework*

RDFS: *RDF Vocabulary Description Language*

RDQL: *RDF Data Query Language*

RFID: *Radio Frequency Identification*

RuleML: *Rule Markup Language*

SALT: *Speech Application Language Tags*

SCK: *Semantic Context Kernel*

SeCoM: *Semantic Context Model*

SOAP: *Simple Object Access Protocol*

SOCAM: *Service-Oriented Context-Aware Middleware*

SOUPA: *Standard Ontology for Ubiquitous and Pervasive Applications*

SPARQL: *SPARQL Query Language for RDF*

SPEM: *Software Process Engineering Metamodel*

StRES: *Storing, Retrieving and Extending Service*

SUMO: *Suggested Upper Merged Ontology*

SWRL: *Semantic Web Rule Language*

SWEET: *Semantic Web for Earth and Environmental Terminology*

TAGGER: *Team-Aware Acquisition Guide for Goals, Entities, and Relationships*

TCP/IP: *Transmission Control Protocol / Internet Protocol*

TBox: *Terminological Box*

TIDIA: *Tecnologia da Informação para o Desenvolvimento da Internet Avançada*

TIDIA-AE: *Tecnologia da Informação para o Desenvolvimento da Internet Avançada - Aprendizado Eletrônico*

TOVE: *Toronto Virtual Enterprise*

UML: *Unified Modeling Language*

URI: *Universal Resource Identifier*

URL: *Universal Resource Locator*

xInCA: *Extended InCA*

XML: *Extensible Markup Language*

XMI: *XML Metadata Interchange*

XSD: *XML Schema Definition*

W3C: *World Wide Web Consortium*

Web: *World Wide Web*

WebMemex: *Web Memory Extender*

Wi-Fi: *Wireless Fidelity*

Introdução

O visionário pesquisador Mark Weiser idealizou ambientes físicos com dispositivos computacionais integrados — por exemplo, sensores — que auxiliariam indivíduos na realização de suas tarefas cotidianas ao fornecer-lhes informações e serviços de forma contínua e transparente. À área de pesquisa em que se estuda a integração de tecnologia às atividades de indivíduos de forma transparente, quando e onde for necessário, Weiser deu o nome de *Computação Ubíqua* [Weiser, 1991].

Para o estabelecimento da computação ubíqua, Weiser previu que a interação entre usuários e computadores se distanciaria dos dispositivos tradicionais — teclado, mouse e monitor — e se aproximaria do paradigma de interação em que seres humanos falam, gesticulam e escrevem para se comunicarem uns com os outros [Weiser, 1993]. Alguns autores, como Helal [2005] e Streitz & Nixon [2005], têm apontado vários avanços tecnológicos que têm contribuído para essa mudança de paradigma, como os avanços na micro-eletrônica, nas tecnologias de sensores, de redes sem fio e de redes de alta velocidade, o aumento contínuo do poder de processamento computacional e a proliferação de novos dispositivos de interação, como telefones celulares, *handhelds*, *laptops* e superfícies eletrônicas em geral.

Weiser [1993] propôs que, para investigar o uso desses novos dispositivos de interação, as pesquisas em computação ubíqua podem explorar o desenvolvimento de aplicações. O desenvolvimento de aplicações de computação ubíqua inclui, entre outros, quatro temas de pesquisa principais: interfaces naturais, captura e acesso automatizados de atividades humanas, computação sensível a contexto [Abowd & Mynatt, 2000] e computação no cotidiano [Abowd et al., 2002]. Cada um desses temas de estudo em computação ubíqua é apresentado a seguir.

Aplicações de interfaces naturais têm como objetivo facilitar a capacidade de comunicação entre usuários e computadores ao fornecer suporte a formas naturais de comunicação humana — por exemplo, voz, escrita e gestos — e utilizar as ações implícitas e explícitas que ocorrem nessa comunicação como dados de entrada para sistemas de computação ubíqua [Reeves et al., 2004]. Como evolução das pesquisas que exploram interfaces baseadas na interação exclusivamente por voz [Back et al., 2001; McGlashan et al., 2004], escrita [Landay & Davis, 1999; Chee et al., 2004], gestos [Starner et al., 2000; Westeyn et al., 2003] e reconhecimento de face [Zhao et al., 2003; Shakhnarovich & Moghaddam, 2005], as interfaces multimodais processam vários tipos de entrada do usuário de maneira combinada e coordenada com a saída multimídia de um sistema computacional [Oviatt, 2002]. Pesquisas têm investigado o desenvolvimento de padrões de descrição de informações provenientes de interfaces multimodais [SALT Forum, 2002; Larson et al., 2003], a intersecção entre esse tipo de interface e sistemas biométricos [Jain & Ross, 2004], o desenvolvimento de arcabouços de software para prototipação de interfaces multimodais [Flippo et al., 2003], novas técnicas de integração de múltiplas entradas do usuário [Lee & Yeo, 2005; Oviatt et al., 2005] e a implantação de interfaces multimodais em ambientes diversos, como automóveis [Pieraccini et al., 2004] e residências [Assad et al., 2005].

Aplicações de captura e acesso são aquelas que registram de modo automático informações de experiências do cotidiano humano para acesso posterior [Abowd et al., 2002]. A necessidade desse tipo de aplicação advém da dificuldade humana de registrar todas as informações de seu interesse em experiências diárias como aulas expositivas (*SmartClassroom* [Shi et al., 2003]), reuniões (*TAGGER* [Geyer et al., 2005]), operações cirúrgicas (*ActiveTheatre* [Hansen & Bardram, 2005]) e visitas a museus (*Exploratorium* [Hsi & Fait, 2005]). Aplicações de captura e acesso registram os diversos fluxos de informação dessas experiências — por exemplo, documentos, áudio, vídeo e anotações — tornando explícitas as inter-relações existentes entre cada fluxo, como relações espaciais e temporais. Para exemplificar a viabilidade do registro desses fluxos, Goularte et al. [2004] apresentam o sistema *M4Note* que permite a criação simultânea de anotações multimídia — via caneta em superfícies eletrônicas — sobre imagens sendo capturadas. Macedo et al. [2005; 2006] apresentam seu trabalho em indexação e recuperação de informação para a criação automática de ligações hipermídia entre fluxos de informação capturados do ambiente *iClass* [Cattelan et al., 2003] de captura e acesso de sessões ao vivo correspondentes a aulas, seminários, etc. Os sistemas *M4Note* e *iClass* foram construídos a partir da infra-estrutura de software *xInCA-StRES* (*Extended Infrastructure for Capture and Access – Storing, Retrieving and Extending Service*) de apoio à construção de aplicações de captura e acesso, como descrito por Pimentel et al. [2006].

Aplicações sensíveis a contexto (do inglês *context-aware*) são aquelas que utilizam informações de contexto para fornecer serviços e informações relevantes a usuários e a outras aplicações na realização de alguma tarefa [Dey et al., 2001]. Segundo a definição clássica dada por Dey [2001], *contexto* é qualquer informação — por exemplo, identidade e localização — que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar, ou um objeto físico ou computacional considerados relevantes em uma interação usuário-aplicação. Usando redes de sensores, o sistema *PROACT* [Philipose et al., 2004] monitora como pessoas em estágio inicial de perda de habilidades cognitivas realizam suas atividades diárias em um asilo. Outro exemplo de sistema sensível a contexto, o sistema *OntoNav* [Tsetsos et al., 2005] assiste pessoas com necessidades especiais a se locomoverem em ambientes fechados ao explorar seus respectivos perfis e localizações físicas como informações de contexto. Pessoa et al. [2006] desenvolvem um sistema que monitora sinais vitais de pacientes com problemas cardíacos crônicos. Quando uma situação de emergência é detectada em um paciente, como uma taquicardia, o sistema notifica equipes de serviço mais adequadas para a situação, como médicos de plantão, serviço de ambulância, entre outros.

Aplicações de computação no cotidiano (do inglês *everyday computing*), para Abowd et al. [2002], devem fornecer suporte computacional contínuo — 24 horas por dia, 7 dias por semana — a atividades que podem ocorrer de forma concorrente, sem início e término bem definidos, e que podem ser interrompidas repentinamente. Por exemplo, Rode et al. [2004] observam como pessoas de sexos diferentes programam aparelhos domésticos para realizar ações programadas, como vídeo cassetes, e aqueles para executar tarefas repetitivas, como telefones celulares. Com estudos desse tipo, é possível construir sistemas domésticos customizados que facilitem as tarefas do cotidiano de homens e mulheres. Há sistemas de computação no cotidiano que auxiliam pessoas em situações de distração e eventual esquecimento de detalhes de uma tarefa sendo realizada. O sistema *Cook's Collage* [Tran et al., 2005] auxilia na recordação de tarefas realizadas em uma cozinha munida de sensores de movimento que detectam onde atividades do usuário ocorrem, ou quando certos itens são utilizados, como recipientes de açúcar e sal. Quando disparados, tais sensores ativam câmeras que capturam imagens da cozinha para criar seqüências de imagens, que podem ser consultadas pelo usuário para lembrar a(s) atividade(s) realizada(s) recentemente. Contribuindo com pesquisas em computação no cotidiano, Blom et al. [2005] investigam como usuários utilizam dispositivos de telefonia móvel e, a partir desse estudo, apontam os desafios de tratar a mobilidade de usuários entre ambientes físicos, principalmente entre espaços públicos e privados, bem como tratar a dificuldade de adotar técnicas de projeto de dispositivos e serviços que amenizem as diferenças culturais de forma efetiva.

1.1 Motivação

Helal [2005] observa que pesquisas em computação ubíqua têm relatado, em geral, infra-estruturas de software e protótipos com o intuito de demonstrar como esse novo paradigma pode beneficiar variados domínios de aplicação, como educação, entretenimento e segurança doméstica. Ainda segundo Helal [2005], esses esforços carecem de abrangência devido, sobretudo, à complexidade e à demanda de tempo para o desenvolvimento desses sistemas. Para Gu et al. [2005], tais observações são válidas também para pesquisas em computação sensível a contexto.

Quanto à modelagem de informação de contexto, os modelos existentes, em geral, apresentam diferentes graus de expressividade, formalidade, uso de padrões, técnicas de modelagem, dimensão de informação de contexto e um domínio de aplicações-alvo restrito. Cada um desses aspectos referentes à modelagem de informação contextual é discutido a seguir.

Quanto maior a *expressividade* de um modelo, maior é a sua capacidade de representar a estrutura e a semântica dos conceitos manipulados por um sistema. Quanto mais *formal* o modelo, maior é a capacidade de sistemas sensíveis a contexto realizarem inferências sobre informações de contexto. Como exemplo, o trabalho de Dey et al. [2001] utiliza um modelo de informação contextual baseado na sintaxe da linguagem XML, que carece de formalidade e expressividade para a interpretação de informações de contexto. Por outro lado, a aplicação de bate-papo *ConChat* [Ranganathan et al., 2002] é capaz de inferir a atividade de usuários a partir de informação contextual que segue um modelo baseado em lógica formal de primeira ordem [Barwise & Etchemendy, 2002]. A aplicação *OntoMail* [Khedr & Karmouch, 2004], por sua vez, infere sobre as preferências e redes de contatos de usuários com base em um modelo híbrido composto de ontologias [Gruber, 1993] e lógica nebulosa [Zadeh, 1965].

Com relação ao *uso de padrões*, modelos de informação contextual devem utilizar padrões de representação que facilitem não apenas o processamento dos modelos, mas também que contribuam para o compartilhamento de informações contextuais e, conseqüentemente, promova a interoperabilidade entre sistemas. Hong & Landay [2001] utilizam o padrão XML [Bray et al., 2006] para a definição de uma linguagem de especificação de informações de contexto que também serve para o intercâmbio de mensagens entre sistemas.

Com respeito a *técnicas de modelagem*, técnicas tradicionais podem ser utilizadas para o desenvolvimento de modelos de informação contextual como os modelos de entidades e relacionamentos [Chen, 1976] e a linguagem UML [Rumbaugh et al., 2005], conforme discutido, respectivamente, por Wu et al. [2002] e Henricksen & Indulska [2004a]. Com uma diferente técnica de modelagem, Chen et al. [2004d] utilizam

metadados baseados em ontologias para descrever o perfil de dispositivos de acesso e habilitar a cooperação transparente entre dispositivos de acesso heterogêneos.

Dimensões de informação contextual descrevem as classes de informações de contexto envolvidas em uma interação usuário-computador. Há sistemas que exploram a identidade e a localização de usuários, como o sistema *Family Intercom* [Nagel et al., 2001]. Outros sistemas tratam um conjunto mais rico de informações de contexto, como em Wang et al. [2004a], de modo a manipular, além da identidade e localização de usuários, as atividades que estes realizam em um ambiente doméstico. Tan et al. [2005] desenvolveram um interpretador de contexto baseado em eventos após identificada a relevância de representar em seu modelo de informação contextual os eventos que descrevem a situação em que usuários se encontram.

Ainda em relação à modelagem de informação contextual, os modelos pioneiros eram construídos para *aplicações específicas*, como os sistemas *ParcTab* [Schilit et al., 1993] e *Active Badge* [Want et al., 1992]. Com a adoção de *técnicas de modelagem* e *padrões* de representação de informação o escopo de aplicações-alvo tem aumentado, embora soluções genéricas sejam ainda uma necessidade.

A partir do levantamento realizado sobre aspectos de modelagem de informação contextual, este trabalho identifica a necessidade de modelos que apresentem avanços com respeito a sua expressividade, formalidade, uso de padrões de representação de informação, técnica de modelagem, dimensão semântica e escopo de aplicações-alvo.

Progressos em modelagem de informação contextual podem, entretanto, aumentar a complexidade no desenvolvimento de aplicações sensíveis a contexto. Nesse sentido, várias pesquisas, como Hong & Landay [2001], Román et al. [2002] e Arruda Jr. et al. [2003], sugerem uma delimitação das funções que devem ser desempenhadas por aplicações e por infra-estruturas de software de apoio à construção dessas aplicações. Essa separação de interesses é necessária devido à diversidade e à complexidade das tarefas que sistemas sensíveis a contexto desempenham. Abowd [1999] generaliza as tarefas de sistemas sensíveis a contexto como segue:

1. Aquisição de uma grande e diversificada quantidade de informações de um ambiente de interação;
2. Organização dessa gama de informações em uma estrutura de representação eficiente para recuperação e consulta;
3. Análise dessas informações como variáveis independentes, ou por meio da combinação dessas informações com outras registradas no passado ou presente;
4. Transmissão da interpretação das informações envolvidas a aplicações que realizam alguma ação com base nessa análise;

5. Repetição de todo o processo de forma automática, transparente e adaptada segundo as mudanças nas informações de contexto do ambiente.

Para dar apoio a esse conjunto de tarefas, pesquisas como as de Kindberg & Fox [2002] e Grimm et al. [2004] têm sugerido que infra-estruturas de software devem fornecer um leque de serviços básicos a aplicações sensíveis a contexto, que incluem: captura, distribuição, representação, armazenamento, interpretação e acesso a informações de contexto, notificação de eventos, e adaptação de serviços e informações.

O presente trabalho identifica, portanto, a necessidade de infra-estruturas de software que ofereçam serviços a aplicações sensíveis a contexto com base nas características de um modelo de informação contextual subjacente.

Pesquisas em computação sensível a contexto têm também identificado a demanda por técnicas de Engenharia de Software que abordem a complexidade do processo de desenvolvimento de aplicações sensíveis a contexto, conforme descrito por Helal [2005] e Kortuem et al. [2006]. Embora existam iniciativas nesse sentido, como o processo de Dey et al. [2001] para aplicações sensíveis a informações de contexto capturadas por sensores, e o processo de Weis et al. [2006] para aplicações customizáveis, tais esforços são informais no que diz respeito à definição do processo e, em particular, em termos das fases clássicas de processos de software — análise e especificação de requisitos, projeto, desenvolvimento e validação e verificação.

Portanto, processos de software mais abrangentes para aplicações sensíveis a contexto constituem um ponto de investigação pouco explorado na literatura. Tais processos devem ser estruturados de maneira formal a ponto de explicitar os diferentes fatores relacionados ao desenvolvimento de aplicações sensíveis a contexto, desde a fase de levantamento de requisitos até as fases de validação e de verificação.

1.2 Objetivos

O objetivo geral¹ deste trabalho consiste em desenvolver, em termos de processos de software, uma solução que trate a complexidade de desenvolvimento de software sensível a contexto.

Para atingir o objetivo geral deste trabalho, é proposta a subdivisão do problema de complexidade de desenvolvimento de software sensível a contexto em três frentes de investigação diretamente relacionadas a este: a modelagem de informação contextual, a construção de serviços para processamento de informação contextual e a definição de um processo de software para computação sensível a contexto.

¹Este trabalho foi inicialmente proposto no Workshop de Teses e Dissertações do Simpósio Brasileiro de Sistemas Multimídia e Web [Bulcão Neto & Pimentel, 2003b].

Em relação à modelagem de informação contextual, tem-se como objetivo específico o desenvolvimento de um modelo que represente a semântica de informações de contexto com as seguintes características:

- alto grau de expressividade e formalidade quanto à representação de conceitos e relações envolvidos em cenários de computação sensível a contexto;
- diversidade de informações de contexto com o intuito de atender a vários domínios de aplicação sensível a contexto;
- e conformidade com padrões de representação de informação para facilitar o intercâmbio, o reúso e o compartilhamento de informações de contexto entre aplicações sensíveis a contexto.

Com respeito à segunda linha de investigação deste trabalho, uma vez desenvolvido um modelo de informação contextual com as características supracitadas, determina-se como objetivo específico o desenvolvimento de uma infra-estrutura de serviços que seja habilitada a:

- fornecer operações básicas a aplicações sensíveis a contexto;
- interpretar a semântica explícita do referido modelo de informação contextual;
- e oferecer diversidade de configuração no sentido de atender às demandas de diferentes aplicações sensíveis a contexto.

Referente à terceira linha de investigação deste trabalho, outro objetivo específico é a definição de um processo de software para computação sensível a contexto com as seguintes características:

- que faça uso conjunto de um modelo ontológico de informação contextual e de uma infra-estrutura de serviços para dar suporte ao desenvolvimento de aplicações sensíveis a contexto;
- que apóie diversas fases do ciclo de vida de aplicações sensíveis a contexto;
- e que seja representada por uma notação padrão para modelagem de processos com a finalidade de facilitar a compreensão por seus usuários.

1.3 Metodologia

Esta seção apresenta a metodologia utilizada para a realização deste trabalho.

1. Para tratar a característica de *diversidade de informação contextual* do modelo proposto, foram escolhidas as dimensões semânticas de identificação, localização, tempo e atividade, discutidas por Abowd & Mynatt [2000], e de modo de captura e acesso a dados, discutida por Truong et al. [2001].
2. Para tratar os aspectos de *formalidade* e *expressividade* do modelo proposto, a técnica de modelagem de ontologia foi escolhida para a construção do modelo. Uma ontologia é um vocabulário de termos cuja descrição formal expressa a semântica desses termos, assim como delimita a interpretação e a utilização dos mesmos [Gruber, 1993].
3. Escolhida a técnica de modelagem, foi tratado o aspecto de *uso de padrões* do modelo. O modelo *Semantic Context Model*² (*SeCoM*) [Bulcão Neto & Pimentel, 2005] é baseado em padrões da Web Semântica [Berners-Lee et al., 2001] para representação sintática, estrutural e semântica de informações de contexto. Em particular, o modelo é baseado na linguagem de ontologia OWL (*Web Ontology Language*) [Bechhofer et al., 2004], que permite adicionar a semântica formal que descreve recursos e as relações existentes entre estes. Utilizar padrões de Web Semântica atende também às características de *formalidade* e *expressividade* do modelo *SeCoM*.
4. O passo seguinte foi a escolha de metodologias de apoio à construção de ontologias. Foram escolhidas metodologias que exploram o reúso de definições de ontologias existentes na Web, tais como as ontologias *FOAF* (*Friend of a Friend*) [Brickley & Miller, 2005], *SWEET* (*Semantic Web for Earth and Environmental Terminology*) [NASA, 2006], *OWL-Time* [Hobbs & Pan, 2004] e *CC/PP* (*Composite Capabilities/Preferences Profile*) [Klyne et al., 2004].
5. O modelo *SeCoM* foi construído como um conjunto *modular* de ontologias inter-relacionadas baseadas nas dimensões semânticas selecionadas. A organização das ontologias que compõem o modelo *SeCoM* segue uma abordagem em duas camadas: a camada superior de ontologias é representada pelo modelo em si, enquanto a camada inferior de ontologias deve ser construída pelo projetista de uma aplicação sensível a contexto. Assim, o projetista deve estender o modelo *SeCoM* com o conhecimento que é particular de sua aplicação.

²Ao longo de todo o texto o modelo *Semantic Context Model* será referenciado pela sigla *SeCoM*.

6. Para validar o modelo *SeCoM*, foi projetada e desenvolvida uma infra-estrutura de serviços para o gerenciamento de informações de contexto baseadas em ontologias. Foi definido um conjunto básico de serviços para tratar informações de contexto instanciadas a partir de quaisquer modelos ontológicos de informação contextual, como o modelo *SeCoM*. Os serviços que compõem a infra-estrutura *Semantic Context Kernel*³ (SCK) [Bulcão Neto et al., 2005b] podem ser customizados para atender a diferentes demandas de aplicações quanto a armazenamento, consulta e inferência a partir da semântica de informações de contexto.
7. Para validar a infra-estrutura de serviços SCK, o autor estendeu uma aplicação com informação de contexto instanciada a partir do modelo ontológico *SeCoM* [Bulcão Neto et al., 2005a]. A aplicação *WebMemex* [Macedo et al., 2003] em questão captura a navegação Web de usuários e permite que estes recomendem páginas Web entre si com base em suas comunidades *on-line*. Bulcão Neto et al. [2005a] descrevem como as ontologias do modelo *SeCoM* são reusadas e estendidas para uso da aplicação *WebMemex*, e também descrevem como essa aplicação faz uso dos serviços que compõem a infra-estrutura SCK.
8. Em seguida, o autor avaliou o serviço de inferência da infra-estrutura SCK configurado com máquinas de inferência diferentes sobre um conjunto crescente de repositórios da aplicação *WebMemex* [Bulcão Neto & Pimentel, 2006a]. Como resultado, observou-se que máquinas de inferência com baixa expressividade para ontologias OWL podem atender às necessidades de inferência da aplicação *WebMemex* com o melhor tempo total de inferência dentre as máquinas experimentadas. Os resultados dessa avaliação corroboraram a viabilidade de um serviço de inferência, assim como o serviço de inferência da infra-estrutura SCK, configurável quanto às linguagens de ontologia, bem como quanto às respectivas máquinas de inferência possíveis de serem usadas.
9. Outra etapa deste trabalho incluiu a avaliação do modelo *SeCoM* quanto a sua *expressividade* e aspectos de *desempenho* em processo de inferência:
 - A *expressividade* das ontologias do modelo *SeCoM* foi medida por meio da máquina de inferência *Pellet* [Sirin et al., 2006]. Bulcão Neto & Pimentel [2006a] mostram que algumas ontologias têm o grau máximo de expressividade de ontologias codificadas em OWL. Assim, para inferir sobre informações instanciadas a partir do modelo *SeCoM*, deve-se prever o uso de máquinas de inferência que atendam a tal expressividade;

³Ao longo de todo o texto a infra-estrutura *Semantic Context Kernel* será referenciada pela sigla SCK.

- A *viabilidade* da abordagem de um modelo ontológico em duas camadas foi comprovada por meio de medições do tempo de importação das ontologias do modelo *SeCoM* a partir da ontologia da aplicação *WebMemex* [Bulcão Neto & Pimentel, 2006a;b]. O tempo médio de importação das ontologias do modelo *SeCoM* não consome mais que 2% do tempo total de inferência;
10. O passo seguinte foi a identificação de questões de projeto relacionadas à inferência sobre modelos ontológicos de informação contextual que devem ser consideradas por desenvolvedores de aplicações sensíveis a contexto, conforme relatado por Bulcão Neto & Pimentel [2006a;b]:
- Além de calcular o *tempo total de inferência* de cada ontologia do modelo *SeCoM* por meio da máquina de inferência *Pellet*, foram também medidos os tempos envolvidos no tempo total de inferência utilizando ontologias, que incluem, entre outros, os tempos de verificação de consistência, de classificação, e de associação de instâncias a classes das ontologias. Foram constatados, dentre outros, que: (a) quanto maior a expressividade de uma ontologia, maior é o seu tempo de verificação de consistência; (b) o tempo de classificação de ontologias sofre influência de outras etapas dentro do processo de inferência; e (c) quanto maior o número de instâncias em uma ontologia, maior será o tempo de associação de instâncias a classes;
 - Os tempos que compõem o processo de inferência sobre ontologias podem ser úteis para identificar em que etapas o processo como um todo tem demandado mais tempo. Isto permite também que desenvolvedores façam correções ou ajustes na ontologia de uma aplicação para adequar o tempo total de inferência ao requisito de tempo de resposta dessa aplicação;
 - As funcionalidades e otimizações oferecidas por cada máquina de inferência devem ser consideradas no momento de sua escolha para inferir sobre modelos ontológicos de informação contextual;
 - Máquinas de inferência com capacidade de expressão restrita podem ser as mais adequadas para atender aos requisitos de inferência de uma aplicação.
11. A partir da experiência com a extensão da aplicação *WebMemex* e da avaliação do serviço de inferência da infra-estrutura *SCK*, foi definido o processo *POCAp*⁴ (*Process for Ontological Context-aware Applications*) de apoio à construção de aplicações sensíveis a contexto baseadas em ontologias, relatado por Bulcão Neto et al. [2006a;b]. O processo *POCAp* prevê a existência de um modelo

⁴Ao longo de todo o texto o processo *Process for Ontological Context-aware Applications* será referenciado pela sigla *POCAp*.

ontológico de informação contextual, assim como o modelo *SeCoM*, e de uma infra-estrutura de serviços capaz de processar a semântica desse modelo, tal como a infra-estrutura *SCK*. De modo geral, o processo *POCAp* prevê as seguintes fases: análise e especificação, projeto, desenvolvimento, e verificação e validação. Cada fase descreve o conjunto de atividades que membros de um grupo de desenvolvimento de uma aplicação sensível a contexto deve realizar.

12. Em seguida, foi realizada uma instanciação do processo *POCAp* para construir aplicações sensíveis a contexto a partir do modelo de informação contextual *SeCoM* e da infra-estrutura de serviços *SCK*, conforme relatado por Bulcão Neto et al. [2006a;b]. Para isso, o processo *POCAp* foi instanciado para descrever o processo de extensão da aplicação *WebMemex* com informações de contexto baseadas no modelo *SeCoM*, e a integração dessa aplicação com os serviços da infra-estrutura *SCK*. Nesse interim, o modelo *SeCoM* facilita a manutenção, a evolução e a portabilidade de uma aplicação porque todo o conhecimento relacionado a essa aplicação está dissociado de sua lógica. O projeto em duas camadas do modelo *SeCoM* também viabiliza o desenvolvimento de aplicações, uma vez que este não gera sobrecarga quanto ao tempo que a ontologia da aplicação consome para importar ontologias do modelo *SeCoM*, conforme descrito em [Bulcão Neto & Pimentel, 2006a;b]. O desenvolvimento de aplicações sensíveis a contexto também é facilitado pela infra-estrutura *SCK*, pois não é preciso implementar na aplicação os serviços oferecidos pela infra-estrutura, e também devido ao fato de a infra-estrutura fornecer um formato padrão de intercâmbio de informações de contexto entre aplicações.

1.4 Contribuições

As contribuições deste trabalho são sumarizadas a seguir:

- O processo de software *POCAp* (*Process for Ontological Context-aware Applications*) de apoio à construção de aplicações cujas informações de contexto têm semântica proveniente de ontologias [Bulcão Neto et al., 2006a;b];
- O modelo de informações de contexto *Semantic Context Model* (*SeCoM*), caracterizado como independente de domínio, formal, modular, extensível e reusável sob a perspectiva de cada dimensão semântica que compõe seu domínio de projeto [Bulcão Neto & Pimentel, 2005]. Por ser baseado em ontologias e padrões da Web Semântica, e por ser utilizado como uma camada uniforme de acesso a informações de contexto, o modelo *SeCoM* oferece suporte à interoperabilidade semântica entre aplicações sensíveis a contexto [Bulcão Neto & Pimentel, 2003a; 2004; 2006a];

- A infra-estrutura *Semantic Context Kernel* (SCK), caracterizada como modular e com serviços configuráveis dedicados a armazenamento, consulta e inferência a partir da semântica de informações de contexto apoiadas por modelos ontológicos de informação contextual [Bulcão Neto et al., 2005b];
- A extensão de uma aplicação Web de captura, acesso e recomendação de páginas Web com informações de contexto instanciadas do modelo *SeCoM*, e a integração dessa aplicação com os serviços de armazenamento, consulta e inferência da infra-estrutura SCK [Bulcão Neto et al., 2005a];
- A identificação de questões de projeto relacionadas à inferência sobre a semântica de informação contextual baseada em ontologias, questões estas que devem ser consideradas por desenvolvedores de aplicações sensíveis a contexto [Bulcão Neto & Pimentel, 2006a;b].

1.5 Estrutura da tese

Os demais capítulos desta tese estão organizados como segue. O Capítulo 2 discorre sobre computação sensível a contexto. São apresentados conceitos básicos referentes a informações de contexto, o conjunto de dimensões semânticas utilizadas para a construção do modelo *SeCoM*, bem como requisitos para desenvolvimento de sistemas sensíveis a contexto. É apresentado também um conjunto de aplicações de computação ubíqua que utilizam informações de contexto.

O Capítulo 3 trata de Web Semântica. São discutidos os papéis de metadados e de ontologias para aplicações na Web Semântica. A seguir são apresentados padrões de metadados e ontologias que contribuíram para a construção do modelo *SeCoM*. Metodologias utilizadas para a construção do modelo *SeCoM* são também discutidas neste capítulo. Destaque é dado a especificações componentes da arquitetura da Web Semântica, como o padrão RDF e a linguagem OWL.

O Capítulo 4 descreve o modelo *SeCoM*. O modelo é descrito quanto as suas principais características, e para cada ontologia principal que compõe o modelo, é detalhada a sua respectiva semântica, bem como aspectos sobre seu processo de desenvolvimento. A avaliação do modelo *SeCoM* é discutida quanto à abordagem e aos diversos critérios utilizados.

O Capítulo 5 apresenta a infra-estrutura de serviços SCK. São discutidas inicialmente as diretrizes de projeto que conduziram o desenvolvimento da infra-estrutura. Em seguida, cada elemento componente da arquitetura da infra-estrutura SCK é detalhado. A implementação dos serviços da infra-estrutura é ilustrada em detalhes, assim como a maneira pela qual esses serviços utilizam um modelo ontológico de informação contextual, tal como o modelo *SeCoM*. Por fim, é descrita uma avaliação

de desempenho do serviço de inferência quanto à utilização das ontologias do modelo *SeCoM*, a partir da qual foram identificadas questões de projeto referentes a processos de inferência sobre modelos ontológicos de informação contextual que precisam ser consideradas por desenvolvedores.

O Capítulo 6 foca no processo de software *POCAp* de apoio ao desenvolvimento de aplicações sensíveis a contexto baseadas em ontologias. O processo *POCAp* é descrito como um conjunto de atividades organizadas e relacionadas de maneira iterativa com o intuito de analisar, especificar, projetar, implementar e testar aplicações dessa natureza. O processo *POCAp* assume que a construção de aplicações deve se basear em uma infra-estrutura de serviços capaz de interpretar a semântica de informações de contexto baseada em ontologias.

O Capítulo 7 discorre sobre uma instanciação do processo *POCAp* na extensão de uma aplicação sensível a contexto. Inicialmente, é detalhado como utilizar a infra-estrutura *SCK* e o modelo *SeCoM* como artefatos auxiliares no desenvolvimento de aplicações sensíveis a contexto baseadas em ontologias. Em seguida, é descrita, passo a passo, cada atividade do processo *POCAp*, desde a análise e especificação de requisitos até a fase de verificação e validação, aplicada a um estudo de caso em que uma aplicação é estendida com informações de contexto apoiadas pelo modelo *SeCoM* e com os serviços oferecidos pela infra-estrutura *SCK*. A partir desse estudo, foram identificadas questões adicionais de projeto referentes à inferência sobre informação contextual ontológica.

O Capítulo 8 discute sobre trabalhos da literatura de computação sensível a contexto que abordam, em geral, desafios do mesmo contexto do presente trabalho: modelagem de informação contextual, infra-estrutura de serviços e soluções de Engenharia de Software para apoiar o desenvolvimento de aplicações sensíveis a contexto. Em seguida, é realizada uma análise comparativa entre os trabalhos apresentados e o trabalho proposto na forma da infra-estrutura *SCK*, do modelo *SeCoM* e do processo *POCAp*. A partir dessa análise comparativa, são elencados aspectos positivos e limitações da abordagem proposta pelo autor.

O Capítulo 9 sumariza a proposta deste trabalho ao revisar os problemas tratados e as contribuições obtidas, ao apresentar as limitações da abordagem subjacente a este trabalho, e ao discutir linhas de investigação quanto a trabalhos futuros.

O Apêndice A descreve as ontologias de apoio à descrição de perfis de usuários do modelo *SeCoM*. Para cada ontologia, é discutida sua respectiva semântica, processo de desenvolvimento, e avaliação quanto à expressividade lógica e ao tempo de resposta de processos de inferência baseados em ontologias.

Computação Sensível a Contexto

A computação sensível a contexto investiga o emprego de informações que caracterizam a situação de uma interação usuário-computador no sentido de fornecer serviços adaptados a usuários e aplicações. Essas informações, conhecidas como informações de contexto, podem ser obtidas de duas formas: *explícita*, quando a informação obtida é expressa intencionalmente pelo usuário, como o reconhecimento de sua voz; ou *implícita*, quando a informação é obtida sem a comunicação intencional do usuário, como seu foco de atenção ou sua localização em um ambiente físico.

Pesquisas em computação sensível a contexto têm abordado várias questões para o projeto e o desenvolvimento de aplicações sensíveis a contexto, como o desenvolvimento de infra-estruturas de software especializadas no gerenciamento de informações de contexto [Grimm, 2004; Gu et al., 2005; Weal et al., 2006], a integração de infra-estruturas de software a sensores e a outros dispositivos de captura [Wu et al., 2002; Khedr & Karmouch, 2004; Bravo et al., 2006] e experiências práticas do uso de informações de contexto [Bardram, 2004; Borriello et al., 2005; Jardim et al., 2005a].

Inicialmente, este capítulo aborda os conceitos básicos de informação de contexto e de aplicação sensível a contexto. São discutidas dimensões semânticas clássicas para modelagem de informação contextual que foram utilizadas para a realização deste trabalho. É apresentado também um conjunto de requisitos propostos na literatura para apoiar o desenvolvimento de sistemas sensíveis a contexto em geral. Em seguida, são apresentados exemplos de aplicações de computação ubíqua que utilizam informações de contexto. Nas considerações finais, o autor relaciona os conceitos discutidos ao longo do capítulo com o objetivo proposto neste trabalho.

2.1 Conceitos básicos

A primeira definição de informação de contexto relatada na literatura foi dada por Schilit & Theimer [1994] na qual informações de contexto seriam informações de localização, de identidade de pessoas e de objetos próximos entre si e das mudanças nesses objetos. De forma similar, Brown et al. [1997] definem informações de 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. Já Dey et al. [1998] consideram informações de contexto como informações sobre o estado emocional de um usuário, seu foco de atenção, sua localização e orientação, data, hora e os objetos e pessoas que se encontram em um mesmo ambiente físico.

Todas essas definições foram propostas a partir de exemplos no sentido de identificar um conjunto de informações que poderiam ser tratadas como informações de contexto. Em virtude disso, projetistas de aplicações sentiam dificuldades ao modelar informações de contexto que não se enquadravam em nenhum desses tipos de informação definidos.

Uma definição amplamente aceita na comunidade de computação ciente de contexto é a de que informação de contexto *“é qualquer informação que possa ser utilizada para caracterizar uma entidade. Uma entidade é uma pessoa, lugar ou objeto considerado relevante para uma interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação em questão”* [Dey et al., 2001].

Ou seja, se uma informação pode ser utilizada para caracterizar o estado de um participante em uma interação usuário-computador, então esta é uma informação de contexto. Tal definição tem facilitado o papel dos projetistas na especificação do tipo de informação de contexto relevante para uma aplicação. O trabalho descrito nesta tese segue essa definição de informação de contexto de Dey et al. [2001].

Também conhecidas como dirigidas a respostas [Elrod et al., 1993], reativas [Cooperstock et al., 1995] ou adaptativas [Brown, 1996], aplicações sensíveis a contexto são o principal foco de estudo em computação sensível a contexto. Em meio a várias definições, este trabalho adota a definição em que uma aplicação é classificada como sensível a contexto *“quando utiliza informações de contexto para fornecer serviços e/ou outras informações relevantes a um usuário, onde a relevância está diretamente relacionada à tarefa que o usuário desempenha em um dado momento”* [Dey et al., 2001].

A próxima seção discute características encontradas em aplicações sensíveis a contexto.

2.2 Características de aplicações sensíveis a contexto

Para Schilit & Theimer [1994], as características de aplicações sensíveis a contexto são determinadas pelo seu comportamento em tempo de execução quanto ao fornecimento de serviços e informações. O trabalho de Schilit & Theimer [1994] deu origem à primeira taxonomia para aplicações sensíveis a contexto, organizadas em quatro grupos: as aplicações que adaptam, de forma manual ou automática, a apresentação de informações a usuários; e as aplicações que executam comandos, de forma manual ou automática, associados a informações de contexto. Ou seja, tanto a apresentação de informação quanto a execução de comandos podem ser realizadas com intervenção do usuário (manual), ou pela própria aplicação (automática).

Pascoe [1998] propõe outra taxonomia para as características de aplicações sensíveis a contexto com foco na identificação de características centrais de computação sensível a contexto. A taxonomia de Pascoe [1998] descreve as seguintes características, não necessariamente presentes em todas as aplicações:

1. Percepção contextual é a habilidade de detectar e apresentar informação contextual ao usuário sob uma forma conveniente. Uma aplicação para monitoramento de carros-forte, por exemplo, deve apresentar informações de localização no formato de latitude e longitude em uma interface gráfica com a metáfora de mapas;
2. Adaptação contextual é a habilidade de executar ou modificar um serviço automaticamente segundo as informações de contexto atuais. Por exemplo, uma aplicação sensível a localização que notifique o telefone celular de emitir um alarme quando seu usuário estiver em uma sala de reuniões;
3. Descoberta de recursos contextuais é a habilidade de localizar e explorar recursos e serviços considerados relevantes para o usuário. Por exemplo, pontos de ônibus de uma cidade poderiam consultar um serviço de transporte rodoviário sensível à localização de cada ônibus; este serviço deveria calcular e transmitir, em tempo-real, o horário de chegada atualizado a cada ponto de ônibus;
4. Expansão contextual é a habilidade de associar informações de contexto à situação em que se encontra o usuário. Por exemplo, uma aplicação que forneça a cada visitante de uma pinacoteca informações adicionais sobre uma obra de arte, ao monitorar o foco de atenção e a localização do visitante no ambiente. Esta característica não está presente na taxonomia de Schilit & Theimer [1994].

Com base nas taxonomias de Schilit & Theimer [1994] e Pascoe [1998], Dey [2000] propõe uma categorização que lista três características que uma aplicação

sensível a contexto pode atender. Com essa categorização, Dey [2000] defende que é possível saber que tipos de características devem ser consideradas no projeto e no desenvolvimento de aplicações sensíveis a contexto:

1. Apresentação de informações e serviços para o usuário é a habilidade que adiciona às taxonomias de Schilit & Theimer [1994] e Pascoe [1998] a distinção entre informações de contexto e os serviços que uma aplicação fornece, onde tanto informações quanto serviços são apresentados de forma manual;
2. Execução automática de um serviço é a habilidade que aplicações têm de executar um serviço de forma automática de acordo com a situação atual do usuário, situação essa baseada em regras do tipo SE-ENTÃO. Esta categoria é uma alusão à característica de adaptação contextual de Pascoe [1998] e de execução automática de comandos associados a informações de contexto de Schilit & Theimer [1994];
3. União de informações de contexto é a habilidade semelhante à expansão contextual definida por Pascoe [1998]. Informações de contexto assumem o papel de marcações que descrevem uma situação do usuário, marcações essas que podem servir como índices para acesso posterior.

A próxima seção discute sobre dimensões semânticas de informação de contexto propostas na literatura.

2.3 Dimensões semânticas de informação de contexto

A partir das definições apresentadas nas seções anteriores, percebe-se que existe uma grande diversidade de informações que podem ser utilizadas como informações de contexto, diversidade essa que depende do domínio da aplicação em questão. Muitas aplicações sensíveis a contexto têm explorado informações de identidade e de localização de pessoas e objetos para proverem algum serviço útil a usuários, como as aplicações pioneiras *Active Badge* [Want et al., 1992] e *ParcTab* [Schilit et al., 1993]. Ambos protótipos utilizavam mecanismos emissores de sinais que forneciam a localização de pessoas em um edifício, além de identificarem essas pessoas em mapas eletrônicos periodicamente atualizados. Com tais informações era possível, por exemplo, realizar transferências automáticas de chamadas telefônicas.

Aplicações sensíveis a contexto mais recentes passaram a utilizar as facilidades do sistema de localização *outdoor* GPS (*Global Positioning System*), bastante utilizado no monitoramento de automóveis em cidades e rodovias. Por exemplo, o sistema *CyberGuide* [Abowd et al., 1997] é utilizado como um guia turístico capaz de escolher conteúdos áudio-visuais para serem exibidos conforme as informações de localização

de pessoas. Com os avanços na área de comunicação por redes sem fio, novos sistemas sensíveis a contexto passaram a explorar informações de localização, como o sistema *Guide* [Davies et al., 2001], que utiliza sinais de redes 802.11 [IEEE, 2006] para identificar a localização de turistas ao longo de uma cidade e, a partir de sua localização, gerar roteiros personalizados.

No entanto, existem outras informações de contexto além de localização e identificação de pessoas e objetos, conforme mostrado na Seção 2.1. A maioria dos sistemas sensíveis a contexto não incorpora várias das informações disponíveis em um ambiente, como noções de tempo, histórico e dados de outros usuários. Em combinação com as características de aplicações sensíveis a contexto apresentadas na Seção 2.2, Abowd & Mynatt [2000] discutem a utilização de cinco dimensões semânticas de informações de contexto para auxiliar projetistas e desenvolvedores na especificação, na modelagem e na estruturação de informações de contexto de suas aplicações. Essas cinco dimensões semânticas são:

- Who (quem) — seres humanos realizam suas atividades e recordam de fatos passados com base na presença de pessoas e/ou objetos. Aplicações sensíveis a contexto devem, portanto, controlar a identificação de todas as entidades participantes de uma atividade no intuito de atender às necessidades de usuários. Informações de contexto de identificação podem incluir, entre outras, nome, email, senha, voz e impressão digital.
- Where (onde) — a mais explorada das dimensões de informações de contexto, a localização de entidades em ambientes físicos é normalmente associada a outras dimensões, como a dimensão temporal When (quando). Ao combinar essas duas dimensões, é possível explorar não apenas a mobilidade de usuários, mas também informações sobre sua orientação em um ambiente físico e, conseqüentemente, fornecer serviços e/ou informações adaptados ao comportamento desses usuários. Informações de contexto de localização incluem, entre outras, latitude, longitude, altitude, cidade e posição relativa a objetos e pessoas.
- When (quando) — informações de contexto temporais podem ser usadas para situar eventos em uma linha do tempo, ou auxiliar na interpretação de atividades humanas e no estabelecimento de padrões de comportamento. Por exemplo, uma visita breve a uma página Web pode indicar falta de interesse do usuário com relação ao conteúdo da página. Já no caso de uma aplicação de monitoramento de pessoas idosas [Hori & Nishida, 2005], essa aplicação verifica se os instantes ou intervalos de tempo das atividades do paciente são compatíveis com a rotina diária do mesmo. Nos casos em que há desvios de padrão, a aplicação deve notificar o médico de plantão. Informações de contexto temporais incluem, entre outras, data, hora, intervalos de tempo, dia da semana, mês e ano.

- What (o quê) — identificar o que um usuário está fazendo em um determinado momento pode ser uma tarefa complicada para uma aplicação em que atividades, não-previstas pelo projeto da aplicação, podem ser realizadas de forma concorrente. Configura-se, assim, como um dos principais desafios na computação sensível a contexto a obtenção de informações de contexto que possibilitem a interpretação correta da atividade de um usuário. Informações de contexto de atividades variam de aplicação para aplicação, por exemplo, escrever na lousa, anotar em um caderno, trabalhar em grupo e participar de uma reunião, palestra, ou operação cirúrgica.
- Why (por quê) — mais desafiador ainda que perceber e interpretar o que um usuário está fazendo, é entender o porquê de sua ação. Em geral, as informações de contexto de atividade (What) e de motivação (Why), por serem mais complexas, são obtidas por meio da combinação de informações de outras dimensões. O estado emocional de um usuário pode também ser indicativo de sua motivação para a realização de uma tarefa. Aplicações sensíveis a contexto podem obter, via sensores, informações que possam dar uma indicação do estado emocional de um usuário [Picard, 2000], por exemplo, o foco de atenção e a expressão facial [Kim et al., 2005], características de batimento cardíaco e níveis de pressão arterial [Wijnalda et al., 2005], entonação vocal [Takahashi et al., 2004] e ondas cerebrais do tipo alfa [Aizawa et al., 2004].

Essas cinco dimensões semânticas discutidas em Abowd & Mynatt [2000] não sugerem completeza, mas sim, um conjunto básico de diretrizes a ser seguido no processo de construção de uma aplicação sensível a contexto. Nesse interim, Truong et al. [2001] discutem uma dimensão semântica originada do domínio de aplicações de captura e acesso:

- How (como) — no contexto de aplicações de captura e acesso, esta dimensão fornece informações relativas a como recursos de um ambiente físico podem ser capturados e acessados. É importante que aplicações sensíveis a contexto tenham informações não apenas do número e do papel dos dispositivos disponíveis para captura e acesso em um ambiente, mas também que estejam informados acerca das características funcionais de cada dispositivo para captura e acesso. Essas informações podem ser utilizadas, por exemplo, para a personalização de acesso a informações capturadas via dispositivos — por exemplo, os *handhelds* — com características de acesso bastante restritas, como tamanho de tela, quantidade de energia em bateria e suporte à entrada e saída de dados.

A seguir são apresentados requisitos clássicos discutidos na literatura para a construção de software sensível a contexto.

2.4 Requisitos para construção de software sensível a contexto

Para construir sistemas sensíveis a contexto, deve-se definir um conjunto de requisitos ou características que sirvam de orientação durante seu processo de desenvolvimento. Esta seção apresenta o arcabouço conceitual definido por Dey [2000], que delimita um conjunto de requisitos que podem ser utilizados para a construção de sistemas sensíveis a contexto, a saber: especificação de informações de contexto, separação entre aquisição e utilização de informações de contexto, interpretação de informações de contexto, comunicação distribuída e transparente, disponibilidade contínua de componentes de captura de informações de contexto, armazenamento de informações de contexto e descoberta de recursos.

2.4.1 Especificação de informações de contexto

Desenvolvedores devem se beneficiar de um mecanismo que lhes permita especificar que tipos de informações de contexto uma aplicação pode requisitar. O mecanismo e a linguagem de especificação devem permitir que os desenvolvedores expressem as informações de contexto como simples ou múltiplos fragmentos, se há relacionamento entre essas informações e se estas são interpretadas ou não. Henriksen et al. [2002] complementam a visão de Dey [2000] com um mecanismo que expressa o grau de precisão das informações de contexto fornecidas por ambientes físicos instrumentados com sensores.

O mecanismo de especificação de informações de contexto deve também permitir múltiplas representações de uma mesma informação de contexto de acordo com a demanda das aplicações. Banavar & Bernstein [2002] acrescentam à proposta de Dey [2000] a necessidade de mecanismos e linguagens de especificação de informações de contexto que favoreçam o intercâmbio, o reúso, a extensão e a generalização dessas informações.

2.4.2 Separar aquisição de utilização de informações de contexto

Não há uma maneira padrão de adquirir e manipular informações de contexto. De maneira geral, informações de contexto são manipuladas de forma *ad hoc*, sem particular preocupação com generalização e reúso. Para a aquisição de informações de contexto, o gerenciamento de eventos é normalmente explorado, ou via consulta (técnica de *polling*), ou via notificação (técnica de *callback*). Embora ambos não sejam necessariamente implementados, é desejável que aplicações possam se beneficiar desses mecanismos por razões de flexibilidade.

Uma vez adquiridas as informações de contexto necessárias via consulta ou notificação, aplicações podem, por exemplo, determinar a ocorrência e a relevância de

modificações nessas informações. Separar os processos de aquisição e de utilização de informações de contexto permite que aplicações utilizem essas informações sem se preocupar com os detalhes de como foram adquiridas.

2.4.3 Interpretação de informações de contexto

Dado que podem existir várias camadas envolvidas no processo de aquisição de informações de contexto, sob a perspectiva dos desenvolvedores de aplicação, essas camadas devem ser transparentes. Para apoiar tal transparência, as informações de contexto devem ser interpretadas antes de serem utilizadas por aplicações. A interpretação de informações de contexto envolve a análise dessas informações como variáveis independentes, ou então a combinação dessas informações com outras registradas no passado ou presente. A interpretação pode ser, por exemplo, do nome de um usuário para o endereço de correio eletrônico correspondente. A combinação de mecanismos de interpretação e de especificação permite que aplicações possam requisitar as informações de contexto desejadas e adquiri-las em sua forma interpretada quando necessário.

2.4.4 Comunicação distribuída e transparente

À medida que ambientes tornam-se cada vez mais instrumentados, por exemplo com sensores, uma maior quantidade e diversidade de informações de contexto pode ser adquirida, distribuída e compartilhada. A comunicação entre sensores e aplicações sensíveis a contexto deve ser, em geral, transparente, uma vez que sem essa transparência o desenvolvedor deve especificar e implementar um protocolo de comunicação e um esquema de codificação e decodificação para a transmissão de informações de contexto, requisito este ratificado por Zimmer [2004].

2.4.5 Disponibilidade contínua de componentes de captura de informações de contexto

É necessário que os componentes que capturam informações de contexto sejam executados de maneira independente das aplicações que os executam. Além disso, esses componentes de captura devem também estar sempre disponíveis. Como não se sabe quando aplicações solicitarão informações de contexto, os componentes devem ser executados continuamente para permitir que aplicações os contatem quando necessário. Em combinação com o requisito de comunicação distribuída, apresentada na Seção 2.4.4, aplicações devem, portanto, assumir que informações de contexto estarão disponíveis a qualquer hora e lugar, independentemente da localização das entidades envolvidas em um cenário de computação sensível a contexto.

2.4.6 Armazenamento de informações de contexto

A necessidade de manter o histórico de informações de contexto é um requisito ligado à aquisição de informações de contexto bem como à disponibilidade contínua dos componentes de captura de informações de contexto, assuntos abordados nas Seções 2.4.2 e 2.4.5. Um histórico de contexto pode ser utilizado para estabelecer tendências e prever valores futuros de informações de contexto. Sem o armazenamento de informações de contexto, esse tipo de análise não é possível de ser realizado.

Spence et al. [2005] ressaltam, porém, que grandes quantidades de informações de contexto armazenadas podem dificultar o acesso via dispositivos móveis, comuns em sistemas de computação ubíqua, mas normalmente com restrições de memória principal. Nesse sentido, Spence et al. [2005] têm explorado subconjuntos de históricos de informação contextual.

2.4.7 Descoberta de recursos

Ao se comunicar com dispositivos de captura de informações de contexto, uma aplicação deve saber que tipos de informações o dispositivo pode fornecer, qual a localização desse dispositivo e que protocolos e linguagens devem ser utilizados para se comunicar com o mesmo. Assim que uma determinada aplicação é iniciada, deve-se especificar o tipo de informação de contexto requisitado. Com isso, o mecanismo de descoberta de recursos informa onde encontrar os componentes adequados, bem como descreve os mecanismos de acesso para eles. Complementando a visão de Dey [2000], Forstadius et al. [2005] apontam ainda o desafio de se localizar o componente mais apropriado para uma particular situação em que usuários se encontram.

O mecanismo de descoberta de recursos pode ser também utilizado por uma aplicação para determinar se a informação de contexto requisitada está disponível no ambiente. Ainda, o mecanismo pode ser combinado com o mecanismo de especificação (vide Seção 2.4.1) e com os componentes de captura (vide Seção 2.4.2) para determinar quais situações podem ser capturadas e se uma dada requisição de informação de contexto pode ser realizada pela infra-estrutura de serviços em execução. Caso essa infra-estrutura não forneça a informação de contexto necessária, o desenvolvedor deve saber que componentes precisam ser adicionados.

2.5 Exemplos de aplicações sensíveis a contexto

Esta seção apresenta aplicações de computação ubíqua que utilizam informações de contexto para fornecer serviços a usuários. São discutidos os tipos de serviços prestados e as informações de contexto utilizadas por cada aplicação.

2.5.1 Conference Assistant

Conferências são eventos que normalmente envolvem uma grande quantidade de pessoas e de atividades paralelas — apresentações de artigos, demonstrações e reuniões de grupos de trabalhos. Pesquisadores do Instituto de Tecnologia da Georgia nos EUA desenvolveram a *Conference Assistant* [Dey et al., 1999b], uma aplicação de captura e acesso que utiliza informações de contexto para auxiliar um participante de conferência a decidir de que atividade deve participar e a saber que atividades outros participantes realizam. Ao utilizar essa aplicação em seu PDA (*Personal Digital Assistant*), o usuário pode fazer anotações sobre apresentações durante ou a posteriori, bem como acessar as informações capturadas após a conclusão de cada atividade.

Ao chegar a uma conferência, o participante registra seus dados pessoais, os temas de pesquisa de seu interesse e uma lista de colegas que também participam da conferência. Ao longo de toda a conferência, cada participante carrega consigo um PDA com a aplicação *Conference Assistant*, que mostra a agenda da conferência destacando as apresentações cujos assuntos podem ser de interesse do participante. Como o local da conferência é munido de uma infra-estrutura computacional que monitora a localização de seus participantes, quando um usuário dirige-se a uma sala de apresentação, a aplicação lhe apresenta as informações capturadas da apresentação corrente até o momento de sua chegada. Assim, o participante pode fazer comentários sobre as apresentações em seu PDA.

A aplicação *Conference Assistant* exibe também a participação de seus colegas em outras atividades da conferência. Se um participante se mover para outra sala cuja apresentação tem despertado muito interesse de algum de seus colegas, o conteúdo da apresentação anterior continua sendo capturado, e o conteúdo da nova apresentação passa a ser transmitido também para o PDA desse participante. Ao término da conferência, as anotações de cada participante são integradas ao conteúdo das respectivas apresentações assistidas.

As dimensões semânticas de informação utilizadas pela aplicação *Conference Assistant* incluem a dimensão de tempo (início e término de cada atividade da conferência, instantes de tempo em que participantes adentram as salas de cada atividade), a dimensão de atividade (apresentações e demonstrações assistidas por cada participante), a dimensão identidade (de palestrantes e participantes) e a dimensão de localização (tanto de cada participante quanto de cada atividade da conferência).

2.5.2 CARE™

Como exemplo de aplicação de computação ubíqua em prol da qualidade de vida de pessoas, a empresa *Elite Care* construiu uma residência repleta de sensores com o objetivo de criar um ambiente personalizado satisfatório a pessoas da terceira idade em contraposição ao modelo tradicional de cuidados médicos em asilos. Através de uma rede de sensores, a aplicação *CARE™* registra as atividades e sinais vitais de cada residente para que uma equipe médica local identifique quais dentre os residentes que necessitem de cuidados imediatos [Stanford, 2002].

Figura 2.1: (a) Mapa da residência da *Elite Care* com destaque para a localização de residentes com temperatura corpórea acima do normal. (b) Gráficos relacionados às atividades de um residente que servem para diagnóstico de sua saúde física e mental. Adaptado de [Elite Care, 2006].

Cada residente possui um crachá que age como chave de seu apartamento. À medida que um residente se move, seu crachá emite sinais de rádio-freqüência aos sensores espalhados no ambiente, o que permite monitorar a localização desse residente, conforme mostra a Figura 2.1a.

A cama de cada residente possui sensores por meio dos quais médicos podem identificar perdas repentinas de peso e alterações de pressão sanguínea. Identidade, localização, peso, temperatura corpórea, pressão sanguínea e atividades sociais de cada residente são armazenadas em bancos de dados personalizados a partir dos quais são calculadas tendências sobre a saúde física e mental de cada paciente.

A Figura 2.1b mostra uma interface da aplicação *CARE™* que exibe gráficos personalizados de um residente que mostram o número e a duração de chamadas realizadas ao corpo médico, intensidade de atividades realizadas durante um dia,

quantidade de mudanças de localização na residência, grau de socialização, peso médio, e número de horas de sono semanal. Enfermeiros podem utilizar tais informações para chamar por atenção médica qualificada quando necessário.

2.5.3 Co-occupant Awareness

Existem situações do cotidiano humano em que pessoas precisam saber da presença de outras em um grande espaço físico fechado, tal como um prédio. Pesquisadores da Universidade de Berkeley nos EUA desenvolveram a aplicação *Co-occupant Awareness* [Heer et al., 2003a] que automaticamente notifica a presença de pessoas em uma mesma sala e disponibiliza informações a respeito dessas pessoas com o objetivo de promover a comunicação inter-pessoal.

Figura 2.2: (a) Mapa com a localização de pessoas em um prédio de um campus universitário. (b) Janela com as informações de contato das pessoas que se encontram na mesma sala que o usuário corrente, no caso *John Zee* [Heer et al., 2003a].

A aplicação inclui um modelo completo de um prédio de campus universitário que representa qualquer pessoa ou objeto que esteja nesse prédio. Sem a infra-estrutura de sensores e de rede de comunicação necessária para os testes do sistema em geral, a aplicação *Co-occupant Awareness* é apoiada por um sistema particular que simula a movimentação de pessoas e objetos por meio de uma interface Web, conforme apresentado na Figura 2.2a.

Quando uma pessoa adentra uma sala, a aplicação é notificada pelo simulador com a identidade dessa pessoa e fornece, em acréscimo, os endereços eletrônicos e

páginas Web de todos os ocupantes dessa sala (vide Figura 2.2b). A disponibilização de informação de contexto privada — localização e informações de contato — é realizada se as políticas de privacidade de cada usuário assim permitirem. À medida que pessoas entram e saem de uma sala, a interface da aplicação é atualizada.

2.5.4 IM4Sports

Em virtude do efeito positivo que músicas têm com respeito à motivação de pessoas para a prática de atividades físicas, pesquisadores da Philips e da Universidade de Vrije na Holanda desenvolveram o *IM4Sports* [Wijnalda et al., 2005]. Esse sistema personaliza o *playback* de músicas de um usuário durante a prática de uma atividade física, especialmente corridas. O sistema *IM4Sports* pode ser utilizado não apenas para motivar o usuário durante a prática de um exercício, mas também para o acompanhamento profissional e o ensino de educação física.

Figura 2.3: As diferentes fases de utilização da aplicação *IM4Sports* incluem um programa de treinamento ou preparação, a seleção de músicas e o *playback* personalizado de músicas. Adaptado de [Wijnalda et al., 2005].

O sistema é composto de um computador pessoal, um tocador de músicas portátil, um monitor cardíaco e um medidor de passos. Segundo apresentado na Figura 2.3, durante a fase de preparação do sistema são configurados os dados pessoais e preferências musicais do usuário, que incluem o seguinte conjunto de informações de contexto: nome, sexo, idade, peso, nível de experiência, níveis máximo e mínimo de batimento cardíaco, tipo de exercício, programa de treinamento, e títulos, artistas e gêneros musicais prediletos.

Durante a prática do exercício, são capturados os dados do monitor cardíaco, do medidor de passos e do tocador de músicas. Esses dados correspondem às seguintes informações de contexto: batimento cardíaco, frequência de passadas, velocidade, distância percorrida e os conjuntos de músicas tocadas, excluídas e preferidas por

um usuário. Juntas, essas informações de contexto servem de entrada para a fase de retroalimentação do sistema, em que o conjunto de músicas do usuário é reconfigurado (vide Figura 2.3). Para cada usuário, o sistema *IM4Sports* então automatiza os tipos de músicas que serão tocadas considerando as características físicas (sexo, idade e peso) e motoras (ritmo, intensidade e duração de treinamento).

2.5.5 ContextContacts

A plataforma computacional de telefonia móvel tem uma estreita relação com seus usuários, pois telefones celulares armazenam informação privada, movem-se com seus usuários e são personalizados quanto a sua aparência e tons de chamada. Assim, a telefonia móvel apresenta características de computação sensível a contexto. Numa tentativa de explorar o uso de informações de contexto em telefonia móvel celular, pesquisadores da Universidade de Helsinki na Finlândia construíram a aplicação *ContextContacts* [Raento et al., 2005].

Figura 2.4: (a) Lista de contatos, cada qual com informações de localização passada e atual, status de operação do telefone (ícone de mão à esquerda), pessoas na proximidade e perfil de alarme do telefone. (b) Ao clicar em um contato, o usuário obtém mais detalhes e uma explicação sobre ícones e atalhos do sistema [Raento et al., 2005].

A aplicação *ContextContacts* permite que usuários compartilhem informações de contexto que possam dar um indício do contexto social em que esses usuários se encontram. A aplicação é composta de dois elementos: (a) o publicador de presença, que coleta informações de contexto do telefone celular de cada usuário, como localização, status de operação, dispositivos próximos e perfil de chamada, tudo associado à validade temporal dessas informações; e (b) o receptor de presença, que recebe todas as informações de contexto do publicador de presença e as integra à interface de apresentação da aplicação *ContextContacts*. A Figura 2.4 apresenta essa interface na qual são exibidas dicas sobre a situação social atualizada de usuários,

dicas essas que podem ser úteis na decisão de se realizar chamadas telefônicas.

Raento et al. [2005] têm investigado como a aplicação *ContextContacts* afeta a socialização de usuários e as práticas de chamadas telefônicas, especialmente como usuários gerenciam a privacidade de seus dados. Uma abordagem seguida pelos pesquisadores é a de proverem metáforas na interface com o usuário que permitam avisar os usuários de que estão revelando informação pessoal.

2.5.6 Friend Locator

Freqüentadores típicos de grandes eventos, como festivais musicais, tendem a se mover constantemente na companhia de amigos, cada qual portando seu telefone celular. Quando se separam, o telefone é usado para saber a localização um do outro. Entretanto, o ambiente desses eventos é, em geral, bastante ruidoso para se falar ao telefone e extenso para localizar alguém facilmente. Além disso, não há precisão quanto à distância que separa as pessoas, o que as deixa normalmente inseguras. Para tratar esses problemas levantados, pesquisadores do Instituto de Tecnologia de Blekinge na Suécia desenvolveram a aplicação *Friend Locator* [Olofsson et al., 2006].

Figura 2.5: (a) Usuário consultando a aplicação *Friend Locator* para encontrar um amigo. (b) Interface da aplicação que mostra a localização corrente do usuário (*You* no mapa) e a que distância seu amigo (*David*) se encontra (250 metros) na direção de um local chamado *Pampas* [Olofsson et al., 2006].

Assumindo que eventos de tal porte possuem mecanismos — como o GPS — para localização em ambientes ao ar livre, usuários podem executar a aplicação *Friend Locator* em seus telefones celulares, bem como cadastrar cada um de seus amigos que possa vir a se perder. A Figura 2.5a mostra um usuário da aplicação *Friend Locator* tentando localizar um amigo que tenha eventualmente se separado.

Devido às pequenas dimensões da tela de um telefone celular, uma metáfora de mapas simplificada mostra apenas a informação necessária ao usuário, no caso, a localização atual do usuário e do(s) amigo(s) que este procura, a distância que os separa, e os nomes das áreas em que eles se encontram, por exemplo, os nomes de palcos de um show musical, como exibido na Figura 2.5b. Além disso, a aplicação *Friend Locator* pode também fazer com que o telefone celular de uma pessoa vibre quando alguém a estiver procurando.

2.6 Considerações finais

Este capítulo buscou fornecer o conhecimento de computação sensível a contexto necessário para o entendimento do objetivo deste trabalho, que inclui as dimensões semânticas que agrupam informações de contexto e os requisitos para construção de aplicações que interpretam essas informações.

As dimensões semânticas de identidade (Who), localização (Where), tempo (When), atividade (What) e modo de captura e acesso de dados (How), discutidas na Seção 2.3, alicerçam o modelo de informação contextual proposto neste trabalho, a ser apresentado no Capítulo 4.

Com respeito aos requisitos para construção de software sensível a contexto, este trabalho complementa os requisitos de especificação e de interpretação de informações de contexto apresentados, respectivamente, nas Seções 2.4.1 e 2.4.3. Quanto ao requisito de *especificação*, este trabalho identifica a necessidade de mecanismos de especificação que considerem a *semântica* de informações de contexto. Dessa forma, é possível melhor descrever a organização estrutural e semântica dos conceitos gerenciados por uma aplicação sensível a contexto. Ao atender a essa necessidade, desenvolvedores podem fornecer a usuários serviços que se beneficiem da semântica de informações de contexto, como inferência, busca e recuperação de informação. O modelo semântico de informação contextual proposto neste trabalho é apresentado no Capítulo 4.

Quanto ao requisito de *interpretação*, de posse de um modelo semântico de informação contextual, este trabalho identifica a necessidade de mecanismos que possam interpretar essa semântica, tais como mecanismos de apoio à inferência de informações de contexto. Embora esses mecanismos possam ser implementados em uma aplicação, a literatura relata que tais mecanismos devam ser implementados como serviços especializados de uma infra-estrutura de software para computação sensível a contexto, conforme já discutido no Capítulo 1, Seção 1.1. Neste trabalho, foi desenvolvida uma infra-estrutura de serviços capaz de interpretar a semântica do modelo de informação contextual proposto, infra-estrutura essa apresentada no Capítulo 5.

Web Semântica

Várias pesquisas sobre a Web, como Berners-Lee et al. [2001] e Fensel et al. [2005], são unânimes em afirmar que a expansão da Web deve-se em parte à simplicidade da linguagem HTML [Raggett et al., 1999]. Documentos HTML são compostos do conteúdo em si, informações de apresentação, um simples modelo de hipertexto, pouca utilização de metadados e quase nenhuma informação semântica. Para Lacy [2005], essa mesma simplicidade da linguagem HTML restringe a variedade de aplicações que podem explorar a quantidade e a diversidade de informações disponíveis na Web corrente.

Como solução para as limitações que a linguagem HTML impõe à Web, Berners-Lee et al. [2001] afirmam que é necessário associar a semântica correspondente às informações contidas nos documentos disponíveis na Web atual. Com significado associado, informações podem ser interpretadas de forma automática por um mais amplo leque de aplicações na Web. Para que esse cenário possa ser factível, é necessário, pois, que a Web seja estendida de três maneiras: informação estruturada, metadados e semântica explícita.

Segundo Lacy [2005], a estruturação de informação facilita a criação, a busca e a manutenção de informação na Web, em contraposição à abordagem centrada em documentos semi-estruturados da Web atual. Metadados, por sua vez, facilitam a procura por respostas às consultas realizadas por usuários, em contraste com as máquinas de busca atuais que demandam uma extensa filtragem manual de seus resultados. Semântica explícita funciona como informação adicional a respeito de como aplicações podem efetivamente processar a informação, que inclui a inferência de novos fatos e a integração de informações distribuídas na Web.

Nesse interim, pesquisadores têm trabalhado rumo à Web Semântica [Berners-Lee et al., 2001], uma extensão da Web atual em que documentos têm conteúdo estruturado, provido de metadados e com significado explícito associado. Para atingir seus objetivos, a Web Semântica tem se apoiado em três elementos básicos: padrões de metadados [Berners-Lee, 1997; Tannenbaum, 2001], ontologias [Gruber, 1993; Gómez-Pérez & Corcho, 2002] e especificações de representação sintática, estrutural, semântica e lógica de informações [W3C, 2001; Fensel et al., 2005].

Este capítulo apresenta os respectivos papéis de metadados e ontologias quanto à descrição de informações da Web Semântica, bem como exemplos de padrões de metadados e ontologias utilizados neste trabalho. São também abordados neste capítulo aspectos de engenharia de ontologias, como processos, metodologias e avaliação. São apresentadas a arquitetura em camadas da Web Semântica e as especificações de representação de informação referentes a cada camada. Por fim, são discutidos a relação da Web Semântica com o presente trabalho e os benefícios obtidos com o seu emprego em infra-estruturas de software para computação sensível a contexto.

3.1 Padrões de metadados

Metadados são elementos descritores que permitem localizar, caracterizar e relacionar recursos de uma maneira compreensível por máquinas [Berners-Lee, 1997]. Recursos são objetos encontrados na Web ou no mundo real, como documentos textuais, imagens, arquivos executáveis, eventos, conteúdo audiovisual e software.

Metadados são importantes na utilização de recursos Web ou do mundo real, por exemplo, para fornecer informações de propriedade autoral sobre recursos [Content-Guard, 2001], para definir políticas de privacidade e acesso [Cranor et al., 2002], na sumarização do significado e na estruturação de recursos [DCMI, 2004], para decidir como interpretar dados e que instâncias recuperar quando múltiplos formatos são fornecidos [Ayars et al., 2005] e na localização de recursos [IDF, 2006]. Metadados são, pois, um mecanismo de atribuição de semântica aos recursos que representam.

Entretanto, devido à diversidade de recursos, é improvável que um único conjunto de metadados atenda a todas as áreas de conhecimento humano. Faz-se surgir, assim, a demanda por padrões de metadados que sejam específicos para diferentes tipos de recursos. Padrões de metadados definem conjuntos de atributos para descrever recursos com relação ao tipo de informação armazenada, à obrigatoriedade, ao significado e à sintaxe do valor do atributo.

Os padrões de metadados utilizados neste trabalho são apresentados a seguir: *vCard* [Dawson & Howes, 1998], *iCalendar* [Dawson & Stenerson, 1998] e *Dublin Core* [DCMI, 2004]. Informações adicionais sobre padrões de metadados podem ser encontrados em Brand et al. [2003].

3.1.1 Padrão vCard

O padrão *vCard* [Dawson & Howes, 1998] tem como objetivo promover a interoperabilidade entre sistemas de software quanto ao intercâmbio de informações pessoais, tipicamente encontradas em cartões de visita. Várias aplicações utilizam seu conjunto de elementos descritores: softwares de correio eletrônico, vídeo-conferência, telefonia, gerenciamento de informações pessoais e navegadores Web.

O Exemplo 3.1 mostra um trecho de descrição *vCard* de um cartão de visita. Os elementos N e FN descrevem sobrenome, primeiro nome, nome adicional e nome completo da pessoa (linhas 1 e 2). O elemento TEL tem vários tipos, no caso, domiciliar (HOME), móvel (CELL) e para serviço de voz (VOICE) (linhas 3 e 4). O elemento ADR também tem vários tipos, como o domiciliar (linha 5). No elemento EMAIL pode ser especificado o endereço eletrônico preferencial (PREF) para correspondência (linha 6).

```
0  <!--                               Exemplo 3.1                               -->
1  N:Bulcão Neto;Renato;de Freitas
2  FN:Renato de Freitas Bulcão Neto
3  TEL;HOME;VOICE:(16) 3361-2201
4  TEL;CELL;VOICE:(16) 8133-5358
5  ADR;HOME;;;Rua Alvarenga Peixoto 331,Apto 22;São Carlos;São Paulo;13566-582;Brasil
6  EMAIL;PREF;INTERNET:rbulcao@icmc.usp.br
```

3.1.2 Padrão iCalendar

O padrão *iCalendar* [Dawson & Stenerson, 1998] tem como objetivo promover o intercâmbio de informações pessoais de agendamento de eventos — como compromissos e atividades a fazer — entre aplicações sob um formato independente de plataforma. Aplicações que utilizam o *iCalendar* incluem navegadores Web e gerenciadores de informação pessoal e de grupos para *desktops* e PDAs.

O Exemplo 3.2 descreve uma chamada para envio de artigos (SUMMARY) para participação em uma conferência (CONFERENCE) sobre Computação Ubíqua (DESCRIPTION), confirmada (STATUS) entre os dias 17 e 21 de setembro de 2006 nos Estados Unidos (LOCATION) (linhas 1 a 7). Dados do organizador (ORGANIZER na linha 8) da conferência são também representados, como nome (CN) e email (MAILTO). O prazo limite (DUE) para submissão de artigos é 31 de março de 2006 (linha 9).

```
0  <!--                               Exemplo 3.2                               -->
1  SUMMARY:Call for papers
2  CATEGORIES:CONFERENCE
3  DESCRIPTION:Call for papers - 8th International Conference on Ubiquitous Computing
4  STATUS:CONFIRMED
5  DTSTART:20060917
6  DTEND:20060921
7  LOCATION:Orange County, California, USA
8  ORGANIZER:CN:Gregory Abowd;MAILTO:abowd@cc.gatech.edu
9  DUE:20060331
```

3.1.3 Padrão Dublin Core

O padrão *Dublin Core* [DCMI, 2004] tem como objetivo promover a interoperabilidade entre coleções e sistemas de indexação por meio de 15 elementos descritores de recursos, que sugerem um vocabulário controlado e compreensível por sistemas de software. O padrão *Dublin Core* é bastante utilizado no domínio de bibliotecas digitais.

O Exemplo 3.3 descreve uma referência bibliográfica com o título do artigo (Title), autor (Creator) e co-autor (Contributor) (linhas 1 a 3). São também descritos o assunto (Subject) que trata o artigo, na forma de palavras-chave, e a organização responsável (Publisher) pela publicação do artigo (linhas 4 e 5).

```
0  <!--                               Exemplo 3.3                               -->
1  DC:Title = "Toward a domain-independent semantic model for context-aware computing"
2  DC:Creator = "Renato de Freitas Bulcão Neto"
3  DC:Contributor = "Maria da Graça Campos Pimentel"
4  DC:Subject = "Context-aware computing, Semantic Web, Metadata, Ontologies"
5  DC:Publisher = "IEEE CS Press"
```

Naturalmente, os descritores fornecidos pelos padrões de metadados *vCard*, *iCalendar* e *Dublin Core* não cobrem as necessidades de descrição de todos os tipos de recursos na Web. Entretanto, apesar de sua simplicidade, esses padrões são extensíveis, ou seja, fornecem um núcleo básico de descritores para a definição de novos vocabulários segundo as necessidades de autores de aplicações Web.

3.2 Ontologias

A comunidade de Inteligência Artificial adota como definição clássica de ontologia aquela proposta por Gruber [1993], em que “*ontologia é uma especificação formal de uma conceitualização*”. Outra definição amplamente aceita na comunidade, Fensel [2001] estende a definição de Gruber ao afirmar que “*ontologia é uma especificação explícita e formal de uma conceitualização compartilhada*”.

Conceitualização refere-se a um modelo abstrato de algum fenômeno que identifica os conceitos relevantes desse fenômeno. *Explícita* significa que os tipos de conceitos utilizados e as restrições no seu uso são explicitamente definidos. *Formal* refere-se ao fato de que a ontologia deve ser representada sob uma linguagem legível por sistemas de software. *Compartilhada* reflete a noção de que uma ontologia representa o conhecimento consensual de um domínio [Fensel, 2001].

Quando uma ontologia define a semântica de conceitos voltados para um domínio específico, esta é chamada ontologia de domínio ou vertical. Em contrapartida, quando a semântica dos conceitos de uma ontologia independe de um domínio particular, esta é denominada ontologia de nível superior, independente de domínio, ou horizontal [Gómez-Pérez et al., 2004].

Para descrever de forma explícita a semântica do vocabulário de uma ontologia são necessários primitivas de modelagem e relacionamentos semânticos [Lacy, 2005]. Primitivas de modelagem formam a base de qualquer ontologia, representadas por classes, propriedades e indivíduos (ou objetos). Os relacionamentos semânticos expressam os variados tipos de relacionamentos que podem existir entre as primitivas de modelagem de um domínio, tais como especialização, generalização, composição, equivalência, disjunção, simetria, negação, transitividade, entre outros.

Por exemplo, uma ontologia para o domínio de zoologia inclui o seguinte vocabulário: as classes *animal*, *vertebrado*, *invertebrado*, as propriedades *tem_notocorda*, *nome_científico*, *idade*, que são comuns às classes mencionadas, e o indivíduo *João*. Pode-se criar uma taxonomia, em que a classe *animal* é a super-classe do domínio de zoologia, e que suas subclasses diretas são *vertebrado* e *invertebrado*.

Ao associar valores booleanos, verdadeiro e falso, à propriedade *tem_notocorda*, pode-se distinguir entre indivíduos que sejam membros das classes *vertebrado* e *invertebrado*, respectivamente. Estas duas classes podem ser modeladas como disjuntas entre si, ou seja, indivíduos vertebrados não podem ser invertebrados ao mesmo tempo. Cada animal deve ter, no máximo, um *nome_científico* e uma *idade* que armazenam, respectivamente, seqüências de caracteres e números inteiros positivos.

Esta seção discute aspectos relacionados a ontologias em geral, que incluem os benefícios obtidos com sua utilização e a engenharia de ontologias.

3.2.1 Vantagens do uso de ontologias

Como descrito anteriormente, uma ontologia é um vocabulário consensual de termos que modela de maneira formal e abstrata um domínio de conhecimento.

Uma vantagem obtida com o uso de ontologias é a prevenção de diferentes interpretações a respeito da semântica dos termos de um domínio. Um sistema de software que utiliza a ontologia de zoologia, por exemplo, interpreta que um indivíduo da classe de invertebrados não pode ser vertebrado, pois tais conceitos são disjuntos.

A abordagem declarativa utilizada pelas ontologias permite descrever um domínio sem qualquer compromisso com a implementação de um sistema de software, ou seja, o conhecimento modelado é independente de implementação.

Outro benefício obtido com o uso de ontologias é a interoperabilidade entre sistemas de software. Sistemas que compartilham da ontologia de zoologia, por exemplo, possuem a mesma interpretação da semântica do vocabulário em questão, o que lhes permite integração transparente de serviços, como serviços de armazenamento, classificação, busca e correlação de informações distribuídas na Web.

Dado que a interoperabilidade requer concordância quanto a conceitos, ontologias são publicamente compartilhadas como arquivos distribuídos na Web. Cabe, pois, aos desenvolvedores saberem a localização de cada ontologia necessária para seus

sistemas. Por exemplo, o conteúdo da ontologia de zoologia poderia estar armazenado no arquivo *ontozoo* em um diretório compartilhado de um servidor Web específico.

Outra vantagem obtida com o uso de ontologias é a capacidade de reusar e estender definições de termos de outras ontologias distribuídas na Web. O mecanismo responsável por essa tarefa é o de importação de ontologias. No caso da ontologia de zoologia, esta poderia importar, sem modificações, as definições de uma ontologia para o domínio dos animais invertebrados no sentido de evitar a duplicação de esforços. Ao mesmo tempo, a ontologia de zoologia poderia estender o vocabulário de uma ontologia de mesmo domínio com termos não tratados por esta última.

Por fim, as características de formalidade e semântica explícita das ontologias permitem que sistemas de software sejam capazes de deduzir novos fatos a partir de fatos declarados acerca de um domínio. Um sistema que utiliza a ontologia de zoologia pode deduzir que o indivíduo *João*, declarado como instância da classe de animais vertebrados, deve ter também um *nome_científico* e uma *idade* associados.

3.2.2 Engenharia de ontologias

Existe uma grande variedade de ontologias: quanto à abrangência, podem ser horizontais ou verticais; quanto à expressividade, podem conter simples taxonomias ou complexas combinações de descrições formais; quanto ao formalismo de representação, podem utilizar *frames* [Minsky, 1974] ou lógica de descrição [Baader et al., 2003]. Dada essa diversidade, construir ontologias tem sido assunto de intensa discussão nas comunidades de ontologias [Gómez-Pérez et al., 2004] e da Web Semântica [Antoniou & van Harmelen, 2004].

A engenharia de ontologias inclui estudos sobre a caracterização do processo de construção de ontologias, bem como sobre metodologias para guiar esse processo [Gómez-Pérez et al., 2004]. Esses dois aspectos da engenharia de ontologias são discutidos a seguir.

a) Processo de construção de ontologias

O processo de construção de uma ontologia é dividido em cinco fases: especificação, conceitualização, formalização, implementação e manutenção. Segundo Pinto & Martins [2004], esse processo pode ser caracterizado como evolutivo, uma vez que é possível avançar ou retornar a qualquer fase do mesmo. As fases do processo de construção de ontologias são descritas a seguir.

1. Na fase de especificação, determinam-se o propósito e o escopo da ontologia. O propósito pode ser obtido por meio da pergunta “por que a ontologia está sendo construída?”. Já o escopo pode ser obtido pelas perguntas “qual uso terá a ontologia?” e “quem são os usuários da ontologia?” [Pinto & Martins, 2004].

2. Na fase de conceitualização, descreve-se a ontologia como um modelo conceitual que deve atender à especificação da ontologia. Um modelo conceitual consiste de conceitos do domínio, bem como de relacionamentos entre esses conceitos. Diferentes metodologias de desenvolvimento propõem o uso de diferentes modelos conceituais [Guarino & Welty, 2002; Cristani & Cuel, 2005].
3. Na fase de formalização, transforma-se o modelo conceitual da ontologia em um modelo formal. Conceitos são, em geral, definidos por meio de axiomas que restringem as possíveis interpretações do significado desses conceitos. A organização dos conceitos é geralmente feita na forma de hierarquias via relações de generalização, especialização e composição [Pinto & Martins, 2004].
4. Na fase de implementação, transforma-se o modelo formal da ontologia em uma linguagem de representação de conhecimento. A escolha da linguagem de representação é realizada com base na expressividade do modelo formal da ontologia [Gómez-Pérez et al., 2004].
5. Ao longo da fase de manutenção, atualiza-se e corrige-se a ontologia implementada em resposta a novas demandas, visando garantir a consistência da ontologia [Gargouri et al., 2005].

Segundo Gómez-Pérez et al. [2004], três atividades básicas devem ser realizadas paralelamente ao processo de construção de uma ontologia: aquisição de conhecimento e avaliação e documentação da ontologia.

Durante a atividade de aquisição de conhecimento, um engenheiro de ontologias obtém conhecimento do domínio da ontologia, quer seja por meio de especialistas do domínio, como observação direta, entrevistas e questionários, quer seja por meio de bibliografia relevante, como manuais, relatórios e artigos [Cimiano & Völker, 2005].

Durante a atividade de avaliação, é julgada a qualidade da ontologia. Esta atividade tem maior interação com a fase de implementação, pois testes de avaliação podem conduzir a mudanças na implementação da ontologia [Brank et al., 2005].

A atividade de documentação registra os termos que compõem a ontologia, as decisões tomadas durante o processo e as respectivas justificativas para cada tomada de decisão. A documentação da ontologia facilita não apenas a sua clareza, mas também sua utilização, manutenção e reúso [Grégoire, 2005].

b) Metodologias para construção de ontologias

A literatura descreve duas classes de metodologias para construção de ontologias: uma em que a ontologia é construída desde o início, e outra em que a ontologia é construída a partir do reúso de outras ontologias [Cristani & Cuel, 2005].

Dentre as metodologias para construção de ontologias desde o início, destacam-se a *TOVE* (*Toronto Virtual Enterprise*) [Gruninger & Fox, 1995], a *ENTERPRISE* [Uschold, 1996] e a *METHONTOLOGY* [López et al., 1999]. É apresentada a seguir uma análise comparativa realizada por Pinto & Martins [2004] dessas metodologias considerando o número, o tipo e a organização das atividades envolvidas para construir ontologias, bem como o perfil de usuário adequado para utilizar cada metodologia.

A metodologia *TOVE* possui o menor número de fases, pois agrupa as atividades de conceitualização, formalização e implementação de uma ontologia em uma única fase. As outras fases são a de conceitualização e a de avaliação de ontologias.

Pioneiras na engenharia de ontologias, *TOVE* e *ENTERPRISE* não apresentam atividades de manutenção de uma ontologia. Já a *METHONTOLOGY* propõe uma atividade de manutenção após a implementação da ontologia, daí ser a metodologia com maior número de atividades: especificação de requisitos, conceitualização de conhecimento do domínio, formalização de modelo conceitual, implementação de modelo formal, manutenção, aquisição de conhecimento, documentação e avaliação.

Metodologias diferentes propõem a realização de atividades em tempos distintos. *METHONTOLOGY* propõe que a aquisição de conhecimento deve ocorrer durante todo o ciclo de vida de uma ontologia, mas com esforço adicional na fase de conceitualização. Já a metodologia *ENTERPRISE* propõe que a aquisição de conhecimento seja realizada em apenas uma fase específica do ciclo de vida de uma ontologia, que em sua terminologia é a fase de aquisição de conhecimento e conceitualização.

A partir dos estudos de Pinto & Martins [2002], usuários novatos ou inexperientes na construção de ontologias preferem a metodologia *METHONTOLOGY*, por esta oferecer diretrizes concretas sobre o que fazer em cada uma de suas fases. Já a metodologia *TOVE* não fornece muita orientação a usuários, sendo mais adequada para especialistas em engenharia de ontologias. A metodologia *ENTERPRISE*, por sua vez, fornece um equilíbrio entre orientação e liberdade sobre o quê e como representar o conhecimento do domínio. Isto tem demonstrado que a *ENTERPRISE* pode ser utilizada tanto por novatos quanto por especialistas em engenharia de ontologias.

Além das metodologias que propõem a construção de ontologias desde o início, existem aquelas que defendem o desenvolvimento de ontologias por meio do reuso de outras ontologias [Cristani & Cuel, 2005]. Nesse sentido, são apontados dois processos de reuso implementados por essas metodologias: os processos de fusão [Noy & Munsen, 2000] e de composição [Pinto & Martins, 2002].

No processo de fusão, constrói-se uma ontologia sobre um assunto a partir da unificação do conhecimento de diferentes ontologias sobre o mesmo assunto. No processo de composição, a ontologia é construída por meio da agregação e da combinação de ontologias que modelam assuntos diferentes, relacionados ou não, após essas ontologias terem sido estendidas.

Pinto & Martins [2004] afirmam que poucas metodologias implementam o reúso de ontologias. A metodologia *ONIONS* (*Ontologic Integration Of Naive Sources*) [Gangemi et al., 1998], por exemplo, implementa o processo de fusão seguindo dos seguintes passos: a análise e seleção simultânea de termos relevantes das ontologias de origem; a procura por definições locais desses termos nas respectivas ontologias; a procura por teorias relacionadas às diferenças entre as definições locais; a procura por teorias para o projeto de categorias de mais alto nível da ontologia; a união de definições locais e categorias de alto nível para classificar termos; e a formalização e eventual implementação do modelo.

Já Pinto & Martins [2001] propõem uma metodologia para o processo de composição, que deve ocorrer ao longo de todo o desenvolvimento de uma ontologia, principalmente na fase de conceitualização. Esta metodologia consiste nas seguintes atividades: a identificação e caracterização do conhecimento dos módulos de ontologias; a identificação de ontologias candidatas a reúso; a representação apropriada do conhecimento de ontologias candidatas; a análise de ontologias candidatas por especialistas do domínio e engenheiros de ontologias; a escolha de ontologias para reúso considerando compatibilidade e completude; a aplicação de operações de integração para gerar ontologia final; e a análise da qualidade da ontologia resultante.

Para finalizar as discussões sobre metodologias para construção de ontologias, vale ressaltar o trabalho de Noy & McGuinness [2001], em que são sugeridas decisões de modelagem que um engenheiro de ontologias deve considerar, bem como as vantagens, desvantagens e implicações de cada uma dessas decisões. Essas decisões de modelagem são independentes do processo de construção de ontologias em questão, e são instanciadas para cada fase desse processo como segue:

1. Decidir a que domínio a ontologia deve atender, sua finalidade, que questões a ontologia deve responder e quem deve usá-la e mantê-la.
2. Considerar o reúso de ontologias criadas e validadas por outro projetista. Mais relevante ainda se essas ontologias consideram o mesmo domínio da ontologia que se deseja construir, e se o sistema que deve utilizá-la precisa se comunicar com outras aplicações que já utilizam ontologias específicas.
3. Enumerar termos importantes da ontologia sobre os quais esta deve manipular declarações. Não deve haver preocupação com sobreposições dos conceitos que os termos representam, com relacionamentos entre termos, com atributos que os conceitos podem ter, ou se os conceitos são classes ou atributos.
4. Não confundir nomes de um conceito com o conceito em si durante a definição de classes. Para isso, há sistemas que permitem a inclusão de sinônimos e termos associados a conceitos de uma ontologia. Na definição da hierarquia de

classes, considerar a quantidade de subclasses de uma classe, por exemplo, ao considerar se pode haver outras classes entre uma superclasse e uma subclasse.

5. Definir atributos de classes, pois tomadas isoladamente estas não fornecem informação suficiente para responder às questões definidas na delimitação do domínio e do escopo da ontologia. Atributos definem uma classe e podem ser obtidos da lista de termos da ontologia criada anteriormente. Para cada atributo na lista, é necessário determinar que classe este descreve e vinculá-lo à mesma. Um atributo deve ser associado à classe mais genérica que pode ter esse atributo. Todas as subclasses de uma classe herdam, por conseguinte, seus atributos.
6. Definir restrições sobre atributos, como o tipo de dado do valor armazenado — por exemplo, números inteiros — os valores possíveis desses atributos, as cardinalidades máxima e mínima e outras características de um atributo.
7. Criar instâncias, os conceitos mais específicos de uma ontologia. Para isso, deve-se escolher uma classe, criar uma instância individual dessa classe e associar valores aos atributos dessa instância.

Dado o importante papel que desempenha na realização deste trabalho e para a construção de ontologias, a atividade de avaliação de ontologias é detalhada a seguir.

c) Avaliação de ontologias

Várias abordagens para avaliação de ontologias têm sido propostas na literatura [Vrandecic et al., 2006] que, segundo Brank et al. [2005], podem ser agrupadas em quatro categorias, a saber:

1. Comparação da ontologia com algum padrão, que pode ser uma ontologia;
2. Utilização da ontologia por uma aplicação com avaliação dos resultados;
3. Comparação da ontologia com uma coleção de dados (por exemplo, uma coleção de documentos) que tratam o domínio a ser coberto pela ontologia;
4. Avaliação da ontologia com base em critérios pré-definidos, padrões e requisitos.

Mesmo considerando estas quatro categorias de abordagens de avaliação, Gómez-Pérez [2004] sugere avaliar diferentes níveis de uma ontologia em vez de avaliá-la como um todo, dada a estrutura complexa que uma ontologia pode representar. Dentre os diferentes níveis relacionados a uma ontologia que podem ser avaliados, incluem-se: as camadas léxica [Maedche & Staab, 2002; Velardi et al., 2005] e sintática [Gómez-Peréz, 1996; Gómez-Pérez, 2001], as relações semânticas [Brewster et al., 2004; Sleeman & Reul, 2006], o contexto de aplicação [Porzel & Malaka,

2004; Sirin et al., 2006] e a estrutura, arquitetura e projeto da ontologia [Tello & Gómez-Pérez, 2004; Burton-Jones et al., 2005].

Para orientar o processo de avaliação de uma ontologia, Brank et al. [2005] relacionam os níveis de uma ontologia sujeitas a avaliação com as quatro categorias de avaliação apresentadas. Com base nesse estudo, esta seção apresenta brevemente a categoria de avaliação por meio de uma aplicação (categoria 2), que avalia o contexto de aplicação de uma ontologia. De forma similar, esta seção também aborda a categoria de avaliação via critérios, padrões e requisitos (categoria 4), que avaliam a estrutura, a arquitetura e o projeto de uma ontologia.

Como uma ontologia é normalmente desenvolvida para uso por uma aplicação, aspectos não-funcionais dessa aplicação, como seu desempenho na realização de uma tarefa, podem ser afetados conforme a ontologia utilizada. Baader et al. [2003] mostram que o desempenho de uma aplicação pode ser influenciado pelo número de classes ou instâncias, ou mesmo pela complexidade dos axiomas da ontologia.

Como exemplo, Porzel & Malaka [2004] descrevem um cenário em que uma ontologia é utilizada por uma aplicação para determinar a proximidade semântica de dois conceitos. A tarefa em questão é um problema de reconhecimento de fala, onde a avaliação da saída da aplicação consiste em comparar interpretações das sentenças de entrada da aplicação com os conceitos definidos na ontologia em uso pela aplicação. Informações adicionais sobre esta categoria de avaliação de ontologias podem ser encontradas em Gómez-Pérez [2004].

Uma ontologia pode também ser avaliada por meio de princípios de projeto ou critérios pré-definidos que têm estreita relação com a organização estrutural e funcional dessa ontologia. Esta categoria de avaliação é normalmente realizada de forma manual, o que a difere das demais categorias de avaliação, que quase sempre consistem em processos automatizados.

Os trabalhos de Tello & Gómez-Pérez [2004] e Burton-Jones et al. [2005] são bastante citados na literatura de avaliação de ontologias. Ambos trabalhos propõem critérios e respectivas métricas para avaliação da qualidade de ontologias. Dentre os critérios propostos estão incluídos: coerência (se as inferências são corretas e consistentes com a especificação da ontologia), extensibilidade (se a ontologia permite extensões sem prejudicar sua manutenção), modularidade (se há baixo acoplamento entre os conceitos da ontologia), autoridade (quantas ontologias utilizam os conceitos da ontologia corrente), complexidade lógica (obtida pela combinação dos axiomas da ontologia) e mínimo compromisso ontológico (se apenas o conhecimento essencial de cada conceito é utilizado para permitir a criação de novos conceitos, mais especializados ou estendidos).

3.3 Arquitetura da Web Semântica

Além de metadados e ontologias, existe um terceiro pilar sobre o qual se apóia a Web Semântica: um conjunto de especificações para a identificação de recursos na Web, bem como para a representação sintática, estrutural, semântica e lógica de informações referentes a esses recursos.

Figura 3.1: Arquitetura da Web Semântica. Adaptado de Berners-Lee [2000].

Esse conjunto de especificações está distribuído ao longo das diversas camadas em que se divide a arquitetura da Web Semântica, conforme ilustra a Figura 3.1: a camada básica de dados, a camada de descrição sintática, a camada de descrição semântica e estrutural, a camada de descrição semântica e lógica, a camada de descrição lógica, e a camada de prova e confiança.

3.3.1 Camada básica de dados

O *Unicode* é um padrão de codificação cuja meta é fornecer uma representação numérica universal e não-ambígua para cada caracter — letra, pontuação ou símbolo técnico — independentemente da plataforma de software ou idioma utilizados [Durst & Freytag, 2003]. O *Unicode* tem sido adotado por vários padrões e incorporado aos principais sistemas operacionais e navegadores Web do mercado.

O padrão URI (*Uniform Resource Identifier*) [Berners-Lee et al., 1998] é um mecanismo que localiza e identifica recursos físicos ou abstratos de forma unívoca. Uma URL (*Uniform Resource Locator*) é um caso específico de URI, em que são declarados o protocolo de acesso ao recurso, uma seqüência de caracteres que identifica a máquina onde se encontra o recurso e o próprio recurso em questão.

Assim, a camada básica de dados fornece interoperabilidade em relação à codificação de caracteres e ao endereçamento e nomeação de recursos da Web Semântica.

3.3.2 Camada de descrição sintática

A linguagem XML (*Extensible Markup Language*) é voltada para a representação sintática de recursos de maneira independente de plataforma [Bray et al., 2006]. Essa linguagem tem sido bastante utilizada como formato padrão para intercâmbio de dados estruturados, como é o caso da especificação XMI (*XML Metadata Interchange*) [OMG, 2005], que utiliza a linguagem XML para definir um padrão para o intercâmbio de metadados entre sistemas de modelagem de dados.

Documentos cujo conteúdo e estrutura são representados na linguagem XML são chamados documentos XML. Em geral, documentos XML utilizados no intercâmbio entre aplicações devem ser submetidos a um processo de validação, em que se verifica se seu conteúdo e estrutura seguem as regras de formação de sua gramática.

Gramáticas no padrão esquema XML [Biron & Malhotra, 2004] têm o propósito de definir e descrever uma classe de documentos XML por meio de construtores que restringem e documentam tanto o significado quanto as relações entre as partes constituintes do documento: tipos de dados, elementos e seu conteúdo, e atributos e respectivos valores. Documentos que seguem sua respectiva gramática esquema XML são chamados documentos auto-descritivos válidos, como ilustra a Figura 3.1.

Esquemas XML têm suporte a espaços de nomes XML [Bray et al., 2004], um método para referenciar e distinguir, por meio de URIs, aqueles elementos definidos com mesmo nome, porém representados em esquemas XML diferentes. Dessa forma, um documento XML pode herdar elementos previamente definidos, e associá-los a sua estrutura por meio de referências aos esquemas que definem esses elementos.

Assim, por meio da linguagem XML, esta camada provê interoperabilidade quanto à sintaxe de descrição de recursos da Web Semântica, associados a gramáticas em esquema XML com apoio à herança permitida pelos espaços de nomes XML.

3.3.3 Camada de descrição estrutural e semântica

O padrão RDF (*Resource Description Framework*) [Klyne & Carroll, 2004] é utilizado para descrever e representar metadados associados a recursos. Para atingir seu propósito, o padrão RDF é dividido em três componentes: modelo de dados, sintaxe para intercâmbio de metadados e linguagem de descrição de esquemas.

O *modelo de dados* RDF constitui-se de declarações sobre recursos, onde cada *declaração* é uma tripla (*sujeito, predicado, objeto*) [Manola & Miller, 2004]. Sujeito é um recurso identificado por uma URI. Predicado pode ser uma propriedade do recurso, ou um relacionamento entre um recurso e um objeto. Objeto pode ser um recurso relacionado, ou o valor da propriedade do recurso. O objeto é um literal quando representa o valor de uma propriedade, por exemplo, uma cadeia de caracteres.

Por exemplo, na declaração “a data de revisão da página Web `http://w3.org` é 14/07/2006”, é possível identificar `http://w3.org` como *sujeito*, “data-revisão” como *predicado*, e 14/07/2006 como *objeto*. A URI `http://w3.org`, na verdade, uma URL, descreve a localização do recurso “página Web” da declaração.

O ponto forte do modelo de dados RDF está em sua generalidade: estruturas de dados para hipertextos são facilmente mapeadas para grafos rotulados dirigidos, idéia central do modelo em questão. Sob a representação de grafo, um recurso é representado por uma elipse identificada por uma URI; o valor da propriedade, por retângulo; e a propriedade; por um arco que conecta o recurso ao valor da propriedade. Como propriedades são também recursos, elas são rotuladas com uma URI que identifica o espaço de nomes no qual foram definidas. A Figura 3.2 ilustra a declaração RDF dada como exemplo nesta seção. O prefixo `ex:` é utilizado como sinônimo para a URI do espaço de nomes XML no qual a propriedade “data-revisão” fora definida.

Figura 3.2: Declaração RDF sob a representação de grafo.

Existem várias formas de serialização do modelo de dados RDF, porém a mais comum é a utilização da sintaxe XML [Beckett, 2004]. O Exemplo 3.4 ilustra a sintaxe RDF/XML correspondente à declaração da Figura 3.2. O prefixo `rdf:` (linha 1) referencia o espaço de nomes XML no qual o modelo do RDF fora definido.

```

0  <!--                               Exemplo 3.4                               -->
1  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2      xmlns:ex="http://description.org/schema/"
3  <rdf:Description rdf:about="http://w3.org">
4      <ex:data-revisão>14/07/2006</ex:data-revisão>
5  </rdf:Description>
6  </rdf:RDF>

```

Além de seu modelo de dados e sintaxe, o padrão RDF apresenta um terceiro componente: o esquema RDF (ou RDFS) [Brickley & Guha, 2004]. Esquema RDF é uma linguagem de ontologia que fornece a semântica de interpretação das declarações de um modelo de dados RDF por meio de taxonomias, de relacionamentos entre classes e propriedades e de restrições desses relacionamentos. Em esquemas RDF, classes, propriedades e relacionamentos são subclasses de recursos. A linguagem de descrição de esquemas RDF também utiliza o formato XML como sintaxe básica.

Um esquema RDF para a declaração representada na Figura 3.2 poderia descrever o recurso “página Web” como subclasse de `rdfs:Resource`, ou como subclasse de uma

classe de documentos. O prefixo `rdfs:` corresponde ao espaço de nomes XML no qual a linguagem de descrição de esquemas RDF fora definida.

Quanto à propriedade “data-revisão”, esta poderia ser descrita como pertencente a recursos da classe *Página Web*. Quanto ao tipo de dado armazenado pela propriedade “data-revisão”, este poderia ser descrito como do mesmo tipo de dado da propriedade `dc:date`, definida no espaço de nomes `dc:` do padrão de metadados *Dublin Core*.

Assim, ao combinar os padrões RDF e XML, têm-se um modelo de dados genérico para descrever recursos e relacionamentos entre recursos da Web Semântica, e uma linguagem independente de plataforma e amplamente utilizada para representar esse modelo. A combinação de RDF e XML facilita, pois, a interoperabilidade sintática e estrutural entre aplicações que fazem intercâmbio de metadados.

3.3.4 Camada de descrição semântica e lógica

Ao utilizar esquemas RDF, é possível construir apenas ontologias com expressividade e capacidade de inferência limitadas. Isto se deve ao fato de esquemas RDF fornecerem um conjunto básico de construtores para modelagem de domínios, além de poucos desses construtores poderem ser utilizados para deduzir novos fatos.

Nesse interim, surge a necessidade de uma linguagem de ontologia que apresente não apenas maior expressividade e capacidade de inferência, mas também que seja baseada nos padrões da Web Semântica para representação de informação.

O W3C (*World Wide Web Consortium*) criou, assim, uma linguagem de ontologia padrão com as características acima discutidas. A linguagem OWL (*Web Ontology Language*) [Bechhofer et al., 2004] estende o vocabulário de esquemas RDF ao incluir construtores mais ricos em relação à expressividade e à inferência, utiliza o modelo de dados do padrão RDF, apresenta sintaxe na linguagem XML e também faz uso das especificações esquema XML, para tipagem de valores de propriedades, e espaços de nomes XML, para referenciar ontologias reusadas, estendidas e a ontologia corrente.

Para comportar aplicações com diferentes requisitos de expressividade e inferência, a linguagem OWL apresenta três módulos (ou dialetos) [McGuinness & van Harmelen, 2004]: *OWL Lite*, *OWL DL* e *OWL Full*. DL é um acrônimo para *Description Logics*, ou lógica de descrição, uma técnica de modelagem de conhecimento que provê a sustentação formal da linguagem OWL [Baader et al., 2003]. Cada um dos módulos OWL são apresentados a seguir. Informações adicionais sobre a descrição dos construtores de cada módulo podem ser encontradas em Smith et al. [2004].

OWL Lite: herda do esquema RDF os mecanismos de definição e de hierarquização de classes e propriedades. Além disso, o módulo *OWL Lite* fornece semântica formal para: (a) versionamento e importação de ontologias, (b) equivalência e desigualdade entre classes, propriedades e indivíduos, (c) definição de classes

via intersecção de outras classes, (d) descrição de propriedades transitivas, simétricas, funcionais e inversas, (e) restrição universal e existencial de tipos de propriedades, e (f) restrição da cardinalidade de propriedades (0 ou 1).

OWL DL: estende o módulo *OWL Lite* com construtores para: (a) definição de classes via união, enumeração e complemento de outras classes, (b) relação de disjunção entre classes, e (c) restrição e enumeração de valores de propriedades.

OWL Full: estende o módulo *OWL DL* ao fornecer semântica formal para: (a) eliminar restrições do módulo *OWL DL* com respeito a classes, propriedades, indivíduos e valores de propriedades, (b) tratar classes e valores de propriedades como indivíduos, e (c) dar suporte completo às cardinalidades máxima e mínima de propriedades.

Desenvolvedores devem escolher o módulo OWL com base nos requisitos de suas aplicações. Lacy [2005] sugere que desenvolvedores escolham o módulo OWL adequado com base na relação de custo-benefício entre expressividade e complexidade computacional. Quanto maior a expressividade, baseada nos construtores utilizados em uma ontologia, maior é a complexidade de processamento desta. Este fato tem influência direta na disponibilidade de software¹ para processamento de ontologias OWL, por exemplo, sistemas validadores, APIs e máquinas de inferência.

Apesar de *OWL Lite* fornecer um conjunto de construtores com expressividade limitada, o processamento de ontologias que utilizam este dialeto é realizado em uma quantidade finita de tempo. Já *OWL DL* fornece maior expressividade para a construção de ontologias que, embora apresente algumas restrições, ainda consegue-se garantir decidibilidade. Livre de todas as restrições dos módulos anteriores, *OWL Full* tem a maior expressividade que a linguagem OWL pode oferecer, porém não garante que ontologias construídas neste dialeto sejam processadas em tempo finito.

A Tabela 3.1 sintetiza os principais fatores que desenvolvedores devem considerar na escolha do dialeto OWL a ser utilizado nas ontologias de suas aplicações. É também apresentada a complexidade computacional, no pior caso, relacionada a cada dialeto OWL, conforme descrito em Horrocks et al. [2003].

Tabela 3.1: Critérios para escolha de dialeto OWL e sua respectiva complexidade computacional no pior caso. Adaptado de Lacy [2005].

Critério/Requisito	Dialeto OWL recomendado	Complexidade computacional
Simples taxonomias	<i>OWL Lite</i>	Exponencial determinístico
Decidibilidade	<i>OWL Lite</i> ou <i>OWL DL</i>	Exponencial não-determinístico
Máxima expressividade	<i>OWL Full</i>	Indecidível

¹O endereço <http://www.w3.org/2004/OWL/#projects> apresenta uma relação de ferramentas comerciais e de livre acesso compatíveis com os dialetos da linguagem OWL.

Uma vez escolhido o dialeto OWL, o desenvolvedor passa a construir sua ontologia com o auxílio de ferramentas de edição especializadas, como a Protégé [Gennari et al., 2003] e a SWOOP [Kalyanpur et al., 2005]. Ambas ferramentas fornecem uma interface gráfica e um conjunto de *plugins* que facilitam a tarefa do desenvolvedor durante o processo de construção de uma ontologia.

Em geral, ontologias OWL são representadas em arquivos com sintaxe RDF/XML, disponibilizados na Web e referenciados por uma URI definida pelo autor da ontologia. Para compor uma base de conhecimento OWL, arquivos de ontologias têm associados a si arquivos de instâncias. Arquivos de ontologias contêm descrições dos conceitos do domínio — a literatura de Inteligência Artificial denomina-os *TBox* (*Terminological Box*), ou componentes terminológicos. Arquivos de instâncias, por sua vez, contêm fatos acerca dos componentes terminológicos — em Inteligência Artificial são conhecidos como *ABox* (*Assertional Box*), ou componentes declarativos [Donini et al., 1996].

É apresentado a seguir trechos de uma ontologia OWL para vinhos [McGuinness, 2000] utilizada por um agente, cuja meta é recomendar vinhos de acordo com a refeição servida. Esta ontologia tem sido utilizada para descrever os construtores oferecidos pela linguagem OWL em Smith et al. [2004].

Nos arquivos de ontologias OWL, é necessário indicar que vocabulários específicos são reusados e estendidos por meio de um conjunto de espaços de nomes XML declarados no início da definição da ontologia. O Exemplo 3.5 declara que vinhos (linha 1) são líquidos potáveis (linha 2), e que uvas de vinho (linha 4) são tipos de uvas (linha 5). As classes de líquidos potáveis e uvas estão definidas no espaço de nomes XML identificado pela entidade `&food;` (linhas 2 e 5).

```

0  <!--                                Exemplo 3.5                                -->
1  <owl:Class rdf:ID="Wine">
2    <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
3  </owl:Class>
4  <owl:Class rdf:ID="WineGrape">
5    <rdfs:subClassOf rdf:resource="&food;Grape" />
6  </owl:Class>

```

A ontologia de vinhos pode ser enriquecida com propriedades e restrições. O Exemplo 3.6 declara que vinhos (linha 2) possuem uma propriedade (linha 1) que indica que estes são feitos de uva de vinho (linha 3). Além de serem líquidos potáveis, vinhos são feitos (linha 8) a partir de, no mínimo, um tipo de uva (linha 9).

```

0  <!--                                Exemplo 3.6                                -->
1  <owl:ObjectProperty rdf:ID="madeFromGrape">
2    <rdfs:domain rdf:resource="#Wine"/>
3    <rdfs:range rdf:resource="#WineGrape"/>
4  </owl:ObjectProperty>
5  <owl:Class rdf:ID="Wine">
6    <rdfs:subClassOf>
7      <owl:Restriction>

```

```

8      <owl:onProperty rdf:resource="#madeFromGrape"/>
9      <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">1</owl:minCardinality>
10     </owl:Restriction>
11   </rdfs:subClassOf>
12 </owl:Class>

```

O Exemplo 3.7 apresenta trechos da ontologia em que são definidas propriedades transitivas, simétricas, inversas, funcionais e funcionais inversas. Transitividade é utilizada para indicar que se uma região vinícola está localizada em uma região Y, e Y está localizada em uma região Z, então X está localizada na região Z (linhas 1 e 2). Simetria é utilizada para expressar que regiões vinícolas X e Y estão próximas uma da outra (linhas 6 e 7). Inversão é utilizada para indicar que se uma entidade X produz o vinho Y (linha 14), há uma relação inversa que indica que o vinho Y é produzido por essa entidade X (linha 16). Propriedades funcionais são aquelas que podem ter no máximo um valor para cada instância, como a propriedade que se refere à entidade produtora de vinho (linhas 11 e 12). Propriedades funcionais inversas são aquelas cujo valor identificam uma instância univocamente, como uma chave primária de banco de dados relacional. Neste caso, o vinho Y é produzido por uma única entidade que possui identificação também única (linha 15).

```

0  <!--                                Exemplo 3.7                                -->
1  <owl:ObjectProperty rdf:ID="locatedIn">
2    <rdf:type rdf:resource="&owl;TransitiveProperty" />
3    <rdfs:domain rdf:resource="&owl;Thing" />
4    <rdfs:range rdf:resource="#Region" />
5  </owl:ObjectProperty>
6  <owl:ObjectProperty rdf:ID="adjacentRegion">
7    <rdf:type rdf:resource="&owl;SymmetricProperty" />
8    <rdfs:domain rdf:resource="#Region" />
9    <rdfs:range rdf:resource="#Region" />
10 </owl:ObjectProperty>
11 <owl:ObjectProperty rdf:ID="hasMaker" />
12   <rdf:type rdf:resource="&owl;FunctionalProperty" />
13 </owl:ObjectProperty>
14 <owl:ObjectProperty rdf:ID="producesWine">
15   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
16   <owl:inverseOf rdf:resource="#hasMaker" />
17 </owl:ObjectProperty>

```

Se um arquivo de instâncias contendo a declaração do Exemplo 3.8 for submetido a uma máquina de inferência para a linguagem OWL, é inferido que o indivíduo *LindemansBin65Chardonnay* (linha 1) é um tipo de líquido potável classificado como vinho, pois este possui uma propriedade (linha 2) que é do domínio da classe de vinhos. Além disso, o vinho *LindemansBin65Chardonnay* tem propriedades que indicam sua entidade produtora, a região na qual esta se encontra e regiões adjacentes.


```
0  <!--                               Exemplo 3.8                               -->
1  <owl:Thing rdf:ID="LindemansBin65Chardonnay">
2    <madeFromGrape rdf:resource="#ChardonnayGrape" />
3  </owl:Thing>
```

Embora os Exemplos 3.5 a 3.8 descrevam apenas construtores do módulo *OWL Lite*, tais exemplos servem de ilustração da sintaxe da linguagem OWL. Informações adicionais a respeito da linguagem OWL podem ser obtidas em Bechhofer et al. [2004].

Ao utilizar a linguagem de ontologia OWL, recursos da Web Semântica são, pois, definidos segundo uma semântica formal que utiliza o modelo de dados genérico do padrão RDF e a sintaxe interoperável no formato RDF/XML.

3.3.5 Camada de descrição lógica

A camada de descrição lógica da arquitetura da Web Semântica permite a especificação de regras que atuam sobre recursos da Web. Regras são necessárias quando não é possível, apenas por meio dos construtores da linguagem OWL, expressar mapeamentos sobre conceitos de uma ontologia, como relações parte-todo (ou composição). A comunidade de pesquisa em Web Semântica tem trabalhado rumo à padronização de uma linguagem para definição de regras que permita o seu armazenamento, intercâmbio e recuperação, a ativação de aplicações e a dedução de novos fatos.

Nesse sentido, destacam-se duas linguagens para descrição de regras para a Web Semântica: as linguagens RuleML (*Rule Markup Language*) [Hirtle et al., 2005] e SWRL (*Semantic Web Rule Language*) [Horrocks et al., 2004].

Baseada no padrão XML, RuleML utiliza um subconjunto da linguagem Prolog (*Programming Logic*) [Colmerauer & Roussel, 1992] para a representação de fatos e regras. A Iniciativa RuleML [RMI, 2006], comunidade que lhe dá sustentação, tem como meta principal fornecer, por meio da RuleML, um vocabulário compartilhado para a definição de regras para inferência na Web Semântica. Devido a isso, RuleML foi utilizada como base para a definição da linguagem SWRL.

SWRL é uma linguagem para especificação de fatos e regras baseada na combinação dos dialetos *OWL Lite* e *OWL DL* com a linguagem RuleML. Assim, SWRL utiliza todos os padrões das camadas inferiores da arquitetura da Web Semântica, que incluem URI, XML, esquema XML, espaço de nomes XML, RDF e OWL.

As regras em SWRL possuem a forma de uma implicação lógica com um antecedente (corpo) e um conseqüente (cabeça), e devem ser interpretadas da seguinte maneira: se as condições especificadas no antecedente são verdadeiras, então as condições especificadas no conseqüente também o são. Tanto o antecedente quanto o conseqüente podem ser conjunções de átomos (classes, propriedades ou relações OWL) associados a variáveis sempre precedidas de um ponto de interrogação.

A seguir é apresentada uma regra que define a relação familiar de tio extraída de Horrocks et al. [2004]: se o irmão do pai de X é do sexo masculino, então ele é tio de X. Essa regra é descrita no Exemplo 3.9 utilizando a sintaxe RDF/XML da linguagem SWRL. Perceba que inicialmente são definidas as variáveis que compõem a regra (linhas 1 a 3). A partir da linha 4 é construída a regra em si, reusando o vocabulário para definição de regras da linguagem RuleML. Os átomos do antecedente e do conseqüente são relações e propriedades definidas em uma eventual ontologia OWL cujo espaço de nomes XML é identificado pela entidade ⪚ (linhas 7, 12, 17 e 24). As variáveis SWRL definidas nas linhas 1 a 3 são os argumentos referentes tanto aos átomos do antecedente (linhas 8, 9, 13, 14 e 18) quanto do conseqüente (linha 25 e 26). A linha 19 exibe o valor que a variável *x3* deve conter para a relação *hasSex*; neste caso, o valor para sexo masculino é definido como um indivíduo OWL.

$$hasParent(?x1, ?x2) \wedge hasSibling(?x2, ?x3) \wedge hasSex(?x3, male) \Rightarrow hasUncle(?x1, ?x3)$$

```

0  <!--                               Exemplo 3.9                               -->
1  <swrl:Variable rdf:ID="x1"/>
2  <swrl:Variable rdf:ID="x2"/>
3  <swrl:Variable rdf:ID="x3"/>
4  <ruleml:Imp>
5    <ruleml:body rdf:parseType="Collection">
6      <swrl:IndividualPropertyAtom>
7        <swrl:propertyPredicate rdf:resource="&eg;hasParent"/>
8        <swrl:argument1 rdf:resource="#x1" />
9        <swrl:argument2 rdf:resource="#x2" />
10     </swrl:IndividualPropertyAtom>
11     <swrl:IndividualPropertyAtom>
12       <swrl:propertyPredicate rdf:resource="&eg;hasSibling"/>
13       <swrl:argument1 rdf:resource="#x2" />
14       <swrl:argument2 rdf:resource="#x3" />
15     </swrl:IndividualPropertyAtom>
16     <swrl:IndividualPropertyAtom>
17       <swrl:propertyPredicate rdf:resource="&eg;hasSex"/>
18       <swrl:argument1 rdf:resource="#x3" />
19       <swrl:argument2 rdf:resource="#male" />
20     </swrl:IndividualPropertyAtom>
21   </ruleml:body>
22   <ruleml:head rdf:parseType="Collection">
23     <swrl:IndividualPropertyAtom>
24       <swrl:propertyPredicate rdf:resource="&eg;hasUncle"/>
25       <swrl:argument1 rdf:resource="#x1" />
26       <swrl:argument2 rdf:resource="#x3" />
27     </swrl:IndividualPropertyAtom>
28   </ruleml:head>
29 </ruleml:Imp>

```

A linguagem SWRL tem sido modularizada no sentido de acomodar futuras extensões e fornecer maior flexibilidade a desenvolvedores no suporte a essa linguagem. O módulo básico da SWRL contém os construtores ilustrados no Exemplo 3.9. Os

demais módulos enriquecem a linguagem SWRL quanto à expressividade na definição de regras, como os módulos com construtores para operações matemáticas e para a manipulação de cadeias de caracteres, listas, URIs e dados temporais. Com esse construtores, uma empresa pode definir, por exemplo, uma regra que descreve que se um vendedor vende mais de 100 produtos dessa companhia, então este torna-se membro do clube de maiores vendedores, como apresentado a seguir.

$$\text{Employee}(?x1) \wedge \text{productsSold}(?x1, ?x2) \wedge \text{swrlb : greaterThanOrEqual}(?x2, 100) \Rightarrow \text{memberOfSuperSalesmanClub}(?x1)$$

Um sistema que manipula informações de uma ontologia instanciadas na regra acima pode deduzir que se “Roberto tem vendido 110 produtos, então ele passa a se tornar membro do clube de maiores vendedores da companhia para qual trabalha”.

Atualmente, a especificação da linguagem SWRL está tramitando no W3C rumo a sua padronização como linguagem de definição de regras da Web Semântica. Informações adicionais podem ser encontradas em Horrocks et al. [2004].

3.3.6 Camada de prova e confiança

Para Swartz [2006], nem todas as fontes de informação na Web Semântica deverão ser igualmente confiáveis. Em vez de apenas responder a uma consulta de uma aplicação, um sistema habilitado para a Web Semântica poderia também associar uma prova de como essa resposta foi obtida, o que daria à aplicação que realizou a consulta a possibilidade de avaliar quão confiáveis são os fatos envolvidos nas regras.

No exemplo do vendedor apresentado na Seção 3.3.5, se uma aplicação necessita obter as razões pelas quais o vendedor Roberto é membro do clube de maiores vendedores de uma companhia, então deverão ser retornados a essa aplicação as condições do antecedente da regra em questão, no caso, $\text{Employee}(?x1) \wedge \text{productsSold}(?x1, ?x2) \wedge \text{swrlb : greaterThanOrEqual}(?x2, 100)$, e os fatos que dão sustentação a essa regra, ou seja, $\text{Employee}(\text{Roberto}) \wedge \text{productsSold}(\text{Roberto}, 110)$.

Assim, a camada de prova e confiança da arquitetura da Web Semântica executa as regras definidas na camada de descrição lógica e também avalia a corretude e a confiabilidade da execução dessas regras. A confiabilidade dos fatos envolvidos em regras depende do contexto de execução das aplicações e de assinaturas digitais.

Assinaturas digitais fornecem a prova de confiança a respeito de documentos ou declarações RDF. Uma vez assinados digitalmente, documentos ou declarações RDF têm sua autenticidade garantida, e basta ao usuário informar a uma aplicação que assinaturas esta deve confiar ou não.

Para que esta camada da arquitetura da Web Semântica se desenvolva, as camadas inferiores devem estar sedimentadas, o que ainda não é verdade em

virtude do trâmite da padronização da linguagem SWRL. Segundo Horrocks et al. [2005], mecanismos de prova e confiança ainda merecem profunda investigação pela comunidade de pesquisa em Web Semântica.

3.4 Ontologias da Web Semântica

Esta seção descreve de forma resumida o conjunto de ontologias da Web Semântica que serviram de apoio à construção do modelo de informação contextual proposto neste trabalho: SUMO [Pease & Niles, 2002], *OpenCyc* [OpenCyc.org, 2005], FOAF [Brickley & Miller, 2005], SWEET [NASA, 2006], CC/PP [Klyne et al., 2004] e *OWL-Time* [Hobbs & Pan, 2004].

3.4.1 SUMO

A ontologia SUMO (*Suggested Upper Merged Ontology*) tem sido projetada para servir de base a vários sistemas de processamento de informação, como busca, lingüística e inferência [Pease & Niles, 2002]. O grupo de trabalho 1600.1 do IEEE tem trabalhado em sua padronização como ontologia de nível superior.

A ontologia SUMO trata de meta-conceitos, conceitos gerais que não pertencem a um domínio particular de problemas, que são estruturados na forma de taxonomias, relacionamentos e axiomas. Atualmente, a ontologia conta com 20.000 classes e 60.000 axiomas que têm sido utilizados como base para a construção de ontologias para vários domínios, tais como Comunicações, Transportes e Forças Armadas.

Para maximizar a compatibilidade, projetistas de esquemas podem tentar garantir que suas convenções de nomes usem os mesmos significados para aqueles conceitos definidos na ontologia SUMO que têm as mesmas palavras, como *agent* e *process*. A ontologia SUMO pode ser utilizada livremente para fins acadêmicos e comerciais.

3.4.2 OpenCyc

A ontologia *OpenCyc* é a maior e mais completa base de conhecimento geral disponível para uso acadêmico e comercial [OpenCyc.org, 2005]. *OpenCyc* pode ser utilizada como base para o desenvolvimento de ontologias de domínio e de sistemas especialistas, para a integração de bancos de dados, dentre outros.

Como ontologia de nível superior, a *OpenCyc* inclui 47.000 conceitos cujo domínio é tudo aquilo considerado de consenso humano. Existem ainda mais de 306.000 fatos sobre tais conceitos definindo-os, interrelacionando-os e restringindo-os.

OpenCyc está escrita na linguagem *CycL*, mas existem tradutores para várias linguagens, por exemplo, para a linguagem Lisp. Há também uma API e uma máquina de inferência para construir aplicações que utilizam termos da ontologia *OpenCyc*.

Matuszek et al. [2005] descrevem um método baseado em aprendizado de máquina para introduzir novos conceitos nessa ontologia a partir de páginas Web.

3.4.3 FOAF

A ontologia FOAF (*Friend of a Friend*) é composta por um vocabulário de 12 classes e 49 propriedades e relações que pode ser utilizado para descrever o conteúdo de páginas de pessoas, grupos e organizações, bem como para descrever informações de redes de relacionamentos *online* [Brickley & Miller, 2005].

O foco inicial de FOAF tem sido a descrição de pessoas, uma vez que pessoas relaciona-se a muitos tipos de coisas que são descritas na Web: pessoas criam documentos, participam de reuniões, são descritas em fotografias, relacionam-se com outras pessoas, etc. Dessa forma, documentos escritos em FOAF descrevem as características e relacionamentos de pessoas.

Para facilitar seu processamento e disseminação, documentos FOAF são escritos nas linguagens OWL e RDF. Diferente de uma página Web tradicional, um documento FOAF pode ser combinado com outros documentos de mesma natureza para criar uma base de dados unificada de informação. Por exemplo, Kelly & Dodds [2004] descrevem sistemas para criar, compartilhar e visualizar descrições FOAF de pessoas para apoiar o gerenciamento de comunidades *online* de relacionamentos.

3.4.4 SWEET

Baseadas nas linguagens OWL e RDF, as ontologias SWEET (*Semantic Web for Earth and Environmental Terminology*) são voltadas para projetos de busca e de análise de dados científicos a respeito do planeta Terra [NASA, 2006].

Com milhares de conceitos organizados e interrelacionados, as ontologias SWEET formam uma ontologia de nível superior sobre ciência que inclui, entre outros, fenômenos naturais (terremotos e furacões), atividades humanas (agricultura e comércio), substâncias vivas (biosfera) e não-vivas (radiação e compostos químicos), espaço (divisões político-territoriais) e tempo (durações e unidades temporais).

O projeto GEON (*Geosciences Network*) consiste em uma infra-estrutura computacional com mecanismos de busca, integração semântica e visualização de dados científicos, estruturados a partir das ontologias SWEET, para disponibilizar os avanços de pesquisas em geociência, como geologia e petrologia [GEONgrid.org, 2005].

3.4.5 CC/PP

Com o objetivo de adaptar conteúdo apresentado por meio de diferentes dispositivos conectados à Internet, o vocabulário CC/PP (*Composite Capabilities/Preference*

Profiles) descreve as características físicas e funcionais desses dispositivos, bem como as preferências de usuários na utilização dos mesmos [Klyne et al., 2004].

Um perfil de descrição CC/PP contém pares de atributos e respectivos valores que são utilizados para determinar qual a forma mais apropriada de um recurso que deve ser entregue a um cliente. Assim, um perfil CC/PP funciona como um formato de intercâmbio para que clientes possam descrever suas características. Para padronizar perfis CC/PP, estes são identificados por URIs, modelados segundo o modelo de dados RDF e descritos com sintaxe XML.

Indulska et al. [2003] apresentam um sistema de gerenciamento de informação contextual baseada em perfis CC/PP, utilizados como mecanismo padrão de intercâmbio entre dispositivos. A partir dessa experiência, Indulska et al. [2003] destacam os prós e contras da utilização de CC/PP em computação sensível a contexto.

3.4.6 OWL-Time

A ontologia *OWL-Time* é utilizada para descrever conteúdo temporal de páginas Web e propriedades temporais de serviços Web [Hobbs & Pan, 2004]. Essa ontologia é apoiada por modelos de representação temporal baseados na Álgebra Temporal de Allen, conforme descritos em Allen [1984] e Allen & Ferguson [1994].

OWL-Time fornece um vocabulário com conceitos e relações temporais básicos que a grande maioria das aplicações necessita para indexar fatos em uma linha do tempo, como instantes, intervalos, eventos, durações, múltiplas representações de datas e horas, e relações temporais (antes, depois, durante, etc.).

Hobbs & Pan [2006] têm proposto a ontologia *OWL-Time* ao Consórcio W3C como padrão para representação de informação temporal na Web Semântica. Nesse trabalho são apresentados casos de uso da ontologia na descrição de aspectos temporais de documentos na Web e de serviços Web.

3.5 Aplicações da Web Semântica

Pesquisas em Web Semântica têm explorado um variado nicho de aplicações, que incluem edição de ontologias, anotação semântica de documentos Web via metadados, descoberta, composição e invocação de serviços Web semânticos, personalização, busca e sistemas de recomendação [Gil et al., 2005; Ellis & Hagino, 2005].

Esta seção apresenta aplicações da Web Semântica que fornecem serviços baseados na semântica dos recursos que manipulam, a saber: *Swoogle* [Ding et al., 2004], *SWOOP* [Kalyanpur et al., 2005], *Photocopain* [Tuffield et al., 2006] e *Semantic Wikipedia* [Völkel et al., 2006].

3.5.1 Swoogle

Dado que as máquinas de busca atuais não usufruem a estrutura e a semântica de documentos codificados nos padrões RDF e OWL, existe a demanda por máquinas de busca customizadas para documentos da Web Semântica, especialmente ontologias. Para atender a essa demanda, pesquisadores da Universidade de Maryland nos EUA desenvolveram um sistema de recuperação e indexação de informação para a Web Semântica denominado *Swoogle* [Ding et al., 2004].

Figura 3.3: (a) Interface do sistema *Swoogle* para a consulta de ontologias. (b) Resultado da consulta com os metadados do documento da ontologia encontrada.

A arquitetura do *Swoogle* consiste de agentes que descobrem documentos RDF ou OWL na Web (ontologias ou arquivos de instâncias), um gerador de metadados, uma base de dados que armazena metadados dos documentos encontrados, um extrator de relacionamentos semânticos, uma máquina de busca e indexação desses metadados, e uma interface com o usuário para consultas,² como ilustra a Figura 3.3.

Ao permitir que usuários procurem por ontologias que contenham classes, propriedades e outros termos, *Swoogle* pode tornar mais fácil a tarefa de desenvolvimento de ontologias. Quanto a metadados, ao coletar especialmente aqueles que inter-relacionam documentos, *Swoogle* revela a organização estrutural da Web Semântica, o que permite que usuários façam consultas do tipo “como a Web Semântica está conectada?”, ou “como ontologias são referenciadas?” [Ding et al., 2005].

²Disponível para acesso em <http://swoogle.umbc.edu/>. Visitado em 08/06/2006.

3.5.2 SWOOP

Como uma alternativa para os problemas de usabilidade e de extensão dos ambientes integrados para desenvolvimento de ontologias, pesquisadores da Universidade de Maryland nos EUA desenvolveram SWOOP [Kalyanpur et al., 2005], um editor e navegador para ontologias OWL, cuja interface com o usuário explora a mesma metáfora de navegação via hipertexto de navegadores Web tradicionais.

Figura 3.4: (a) Interface do ambiente SWOOP em que elos de hipertexto permitem edição e navegação entre conceitos de uma ontologia. (b) Mecanismo de validação de tipo de dialeto OWL utilizado na ontologia corrente, classificada como *OWL Full*.

A Figura 3.4 ilustra os mecanismos de edição e navegação entre conceitos da ontologia de vinhos, descrita na Seção 3.3.4, bem como o mecanismo de validação de ontologias, que informa não apenas o tipo de dialeto OWL da ontologia, mas também os construtores que a caracterizam com o tipo de dialeto OWL em questão.³

O principal destaque do SWOOP é que a metáfora de hipertexto está em todas as fases de construção de ontologias: navegação entre conceitos OWL, edição de conceitos OWL relacionados, busca, referência cruzada, documentação via anotações hipermídia, controle de versões, depuração e verificação. Para Kalyanpur et al. [2005], essa variedade de tarefas integradas via hipertexto permite que SWOOP seja utilizado tanto por usuários novatos quanto por especialistas no desenvolvimento de ontologias.

³Disponível para download em <http://www.mindswap.org/2004/SWOOP/>. Visitado em 09/06/2006.

3.5.3 Photocopain

Para auxiliar usuários na árdua tarefa de descrever e organizar sua crescente coleção de fotos e imagens, pesquisadores britânicos desenvolveram um sistema para anotação de imagens chamado *Photocopain* [Tuffield et al., 2006]. Este sistema é capaz de registrar e integrar a uma imagem o máximo de informação possível acerca do ambiente em que foi obtida, utilizando essa informação como anotação da imagem.

Figura 3.5: Interface de anotação do sistema *Photocopain* para permitir que usuários insiram suas próprias anotações [Tuffield et al., 2006].

Câmeras fotográficas fornecem ao *Photocopain* imagens com metadados, que incluem hora de registro, distância focal e orientação da câmera, no formato EXIF (*Exchangeable Image File Format*) [JEITA, 2002]. Ao consultar o diário do usuário que registrou as imagens, o sistema obtém e anexa a essas imagens metadados, no formato *iCalendar* [Dawson & Stenerson, 1998], sobre atividades planejadas que coincidem com o instante em que as imagens foram obtidas, por exemplo, um metadado que descreve uma atividade. Nos casos onde imagens são registradas ao ar livre, metadados de posicionamento geográfico obtidos via GPS são também anexados a essas imagens. Estes metadados também são representados no formato EXIF.

Todos os metadados obtidos são então convertidos para o formato RDF e armazenados em uma base de dados, a partir da qual usuários podem consultar as anotações das imagens capturadas via linguagem SPARQL [Prud'hommeaux & Seaborne, 2006]. Uma interface para anotação de imagens pelo usuário, descrita na Figura 3.5, permite também que usuários estendam a quantidade de informação a respeito das imagens registradas.

3.5.4 Semantic Wikipedia

Wikipedia é uma enciclopédia *online*⁴ que permite a edição e extensão colaborativa de seu conteúdo. Entretanto, o conteúdo da *Wikipedia* carece de estrutura e semântica para inter-relacionar os conceitos que descreve, além de apresentar problemas de redundância e de inconsistência de informações. Como solução para tais problemas, pesquisadores da Universidade de Karlsruhe na Alemanha desenvolveram a *Semantic Wikipedia* [Völkel et al., 2006],⁵ uma extensão da enciclopédia original habilitada com tecnologias da Web Semântica, apresentada na Figura 3.6 (a).

Figura 3.6: (a) Interface da enciclopédia *Semantic Wikipedia* com uma descrição semântica da cidade de Londres. (b) Código-fonte da respectiva descrição na versão original da *Wikipedia*. (c) Código-fonte da mesma descrição na *Semantic Wikipedia*. Adaptado de [Völkel et al., 2006].

As Figuras 3.6 (b) e (c) ilustram, respectivamente, as diferenças entre a *Wikipedia* original e a *Semantic Wikipedia*. No primeiro caso, as ligações são compostas de texto puro e conectam conteúdo sem qualquer informação semântica. No segundo caso, as ligações hipertexto são tipadas com semântica associada, como *capital of* e sua relação inversa *is capital of*, além do fato de que as entidades descritas no conteúdo apresentarem atributos, como *area* e *population*.

Ligações com informação semântica e atributos de entidades são modelados segundo ontologias OWL. A tipagem dos valores de atributos é baseada no padrão esquema XML. Todas as informações gerenciadas pela *Semantic Wikipedia* são armazenadas em bases de dados no formato de triplas RDF que podem ser consultadas

⁴Disponível para acesso em <http://www.wikipedia.org/>. Visitado em 09/06/2006.

⁵Disponível para acesso em <http://wiki.ontoworld.org/>. Visitado em 09/06/2006.

por meio da linguagem SPARQL [Prud'hommeaux & Seaborne, 2006]. Segundo Völkel et al. [2006], aplicações podem utilizar a *Semantic Wikipedia* como base de conhecimento subjacente, como tradutores, dicionários, ferramentas de busca, etc.

3.6 Considerações finais

Este capítulo apresentou os três pilares de sustentação da Web Semântica e, por conseguinte, do modelo de informação contextual *SeCoM* proposto neste trabalho: metadados, ontologias e especificações de descrição de múltiplos níveis de recursos.

Seguindo o processo geral de construção de ontologias, descrito na Seção 3.2.2, o modelo *SeCoM* foi construído a partir dos padrões de metadados *Dublin Core*, *vCard* e *iCalendar*, apresentados na Seção 3.1, e das ontologias SUMO, *OpenCyc*, FOAF, SWEET, CC/PP e *OWL-Time*, apresentadas na Seção 3.4. Tanto dos padrões de metadados quanto das ontologias foram aproveitados seus respectivos termos, a estrutura organizacional e a semântica relacionada a esses termos.

Com exceção das camadas de descrição lógica e de prova e confiança da arquitetura da Web Semântica, as especificações das demais camadas foram utilizadas na construção do modelo *SeCoM*, notadamente os padrões URI, XML, esquema XML, espaço de nomes XML, RDF, esquema RDF e OWL.

Ao utilizar a linguagem OWL, o modelo *SeCoM* é caracterizado como ontológico e, conseqüentemente, semântico e formal quanto à representação de informação contextual. Ao ser descrito por meio de especificações padronizadas da Web Semântica, o modelo *SeCoM* pode facilitar o intercâmbio e o reúso de informações de contexto entre aplicações sensíveis a contexto.

Utilizar ontologias e padrões da Web Semântica também permite que infra-estruturas de software para computação sensível a contexto contenham serviços que processem a semântica de informações de contexto instanciadas do modelo *SeCoM* de maneira interoperável. A interoperabilidade advém do fato de que o modelo ontológico *SeCoM* fornece a aplicações uma visão homogênea de suas informações de contexto instanciadas. Isto permite a integração transparente de serviços, como armazenamento, consulta, descoberta, inferência e outros.

Com respeito a sua avaliação, o modelo *SeCoM* foi avaliado por meio de critérios pré-definidos, de uma infra-estrutura de software, e de uma aplicação Web, tal como descrito nos Capítulos 4, 5 e 7.

Modelo Semântico de Informações de Contexto

Durante os primeiros esforços relativos à definição deste trabalho, no início de 2003, o presente autor identificara que pesquisas em computação sensível a contexto utilizavam a infra-estrutura da Web como meio de apresentação de informações e de fornecimento de serviços [Burrell et al., 2002; Fleck et al., 2002], sem qualquer integração com esforços relacionados à Web Semântica.

Em virtude desse fato, o autor propôs investigar a utilização de tecnologias e padrões de Web Semântica no desenvolvimento de software sensível a contexto, proposta esta publicada em Bulcão Neto & Pimentel [2003b]. A idéia original é que informações de contexto devem ser representadas com semântica explícita de forma a facilitar o seu compartilhamento, reúso e processamento por sistemas de software.

Uma alternativa para atingir esse propósito é introduzir em modelos de informação contextual tecnologias de Web Semântica, tais como as ontologias. Como apresentado no Capítulo 3, Seção 3.2.1, as características ontológicas de formalidade, semântica explícita e abstração de implementação habilitam sistemas de software não apenas a inferir novas informações a partir de informações modeladas por ontologias, mas também a compartilhar entre si essas informações de maneira a integrar de forma transparente os serviços que as manipulam.

Originalmente, este trabalho propunha o desenvolvimento de um modelo ontológico voltado para o domínio de ensino universitário, como publicado em [Bulcão Neto & Pimentel, 2003a; 2004]. Após discussões com seu grupo de pesquisa, o

autor deste trabalho foi incentivado a construir o modelo de informações de contexto independente de domínio, visando atender a vários domínios de aplicações sensíveis a contexto, como ensino, reuniões e outros.

A decisão por um modelo ontológico independente de domínio influenciou as características que este deveria apresentar [Bulcão Neto & Pimentel, 2005]. Inicialmente, o modelo proposto deveria ser formal, com semântica explícita e padronizado, ou seja, para sua representação deveriam ser utilizadas especificações-padrão de Web Semântica. Após a proposta de construção de um modelo independente de domínio, foram incluídas as características de modularidade e extensibilidade, no sentido de estruturar informações de contexto de maneira a facilitar sua extensão para atender aos requisitos de diferentes domínios de aplicação.

Este capítulo apresenta o modelo semântico de informações de contexto desenvolvido neste trabalho, denominado modelo *SeCoM* (*Semantic Context Model*) [Bulcão Neto & Pimentel, 2005]. Para cada ontologia principal do modelo são descritas sua respectiva semântica, aspectos de desenvolvimento e o processo de avaliação. Por fim, este capítulo relata considerações sobre o desenvolvimento do modelo *SeCoM*.

4.1 Visão geral do modelo *SeCoM*

Esta seção descreve o modelo *SeCoM* em linhas gerais quanto as suas principais características e respectivas medidas para atendê-las.

O dimensionamento de informações de contexto descreve as classes de informações envolvidas em uma interação usuário-computador. Mesmo quando o modelo *SeCoM*, em seu estágio inicial, era voltado para o domínio de aplicações de ensino universitário [Bulcão Neto & Pimentel, 2003a], as dimensões semânticas para modelagem de informação contextual eram aquelas discutidas por Abowd & Mynatt [2000] e Truong et al. [2001], apresentadas no Capítulo 2, Seção 2.3: identificação (Who), localização (Where), tempo (When), atividade (What) e modo de captura e acesso (How).

A principal diferença entre a versão mais atual do modelo *SeCoM* [Bulcão Neto & Pimentel, 2006a] e aquela para ensino é que a segunda instancia os conceitos de identificação, localização, tempo, atividade e modo de captura e acesso para o domínio em questão, enquanto que a primeira descreve esses conceitos de forma genérica, com o intuito de atender a vários domínios de aplicação sensível a contexto.

A característica de formalidade do modelo *SeCoM* advém da utilização de ontologias como mecanismo subjacente de modelagem de informação contextual. A característica de padronização do modelo é apoiada por especificações da Web Semântica padronizadas pelo W3C para representação sintática, estrutural, semântica e lógica de informações de contexto, como descrito na Seção 3.3. O modelo *SeCoM* se

apóia sobre a expressividade e a formalidade fornecidas pela linguagem de ontologia OWL [Bechhofer et al., 2004]. O maior grau de expressividade encontrado no modelo *SeCoM* é aquele fornecido pelo dialeto *OWL DL*, o que lhe garante computabilidade em tempo finito, como ilustrado no Capítulo 3, Tabela 3.1.

O processo de construção do modelo *SeCoM* pode ser generalizado como aquele proposto em Noy & McGuinness [2001], onde ontologias podem ser construídas desde o início, ou pelo reuso de outras ontologias. As ontologias que compõem o modelo *SeCoM* foram construídas segundo diferentes metodologias, discutidas no Capítulo 3, Seção 3.2.2, uma vez que o autor deste trabalho não é especialista, nem teve auxílio de especialistas quanto às dimensões de informação contextual que servem de base ao modelo *SeCoM*. Prevalece, em geral, o reuso de definições de metadados e de ontologias existentes na Web, ambos apresentados no Capítulo 3, Seções 3.1 e 3.4.

Com vistas a facilitar a sua extensibilidade, o modelo *SeCoM* é composto de um conjunto modular de ontologias inter-relacionadas baseadas nas dimensões semânticas de identidade, localização, tempo, atividade e modo de captura e acesso.

A disposição das ontologias que compõem o modelo *SeCoM* segue uma abordagem em duas camadas: a camada superior de ontologias, apresentada na Figura 4.1, representa o modelo em si, enquanto que a camada inferior de ontologias é construída por um projetista de uma aplicação sensível a contexto. Nesse caso, o modelo pode ser reusado e/ou mesmo estendido com o conhecimento que é particular dessa aplicação.

Figura 4.1: Visão geral do modelo *SeCoM*. Setas representam o inter-relacionamento entre ontologias via mecanismo de importação. Ovais escuras representam as ontologias principais do modelo, enquanto que as ovais claras auxiliam na descrição de perfis de atores. Adaptado de Bulcão Neto & Pimentel [2006a].

Conforme mostra a Figura 4.1, as ontologias de apoio Knowledge, Relationship, Role, Contact, Document e Project modelam diversos aspectos relacionados a atores, isto é, entidades que executam alguma ação em uma interação usuário-computador. Nas primeiras versões do modelo *SeCoM* [Bulcão Neto & Pimentel, 2004], o conteúdo dessas ontologias estava todo embutido na ontologia Actor. Com a modularização do modelo *SeCoM* [Bulcão Neto & Pimentel, 2005; 2006a], foram criadas essas ontologias de apoio para facilitar o reúso de informações contextuais sobre atores, pois pode-se utilizar apenas o conhecimento necessário a respeito destes. Ontologias de apoio são apresentadas no Apêndice A.

Ainda segundo a Figura 4.1, as ontologias Spatial, Time, Activity e Device modelam, respectivamente, informações de contexto de localização, tempo, atividade e dispositivos computacionais de captura e acesso. As ontologias Spatial Event e Temporal Event são extensões das ontologias Spatial e Time para representar eventos que contenham componentes espaciais e temporais, respectivamente. Cada uma das ontologias que compõem o modelo *SeCoM* são apresentadas a seguir.

4.2 Ontologia Time

A ontologia Time foi a primeira ontologia a ser construída por representar um tipo de informação que envolve conhecimento de senso comum e, na visão do autor deste trabalho, não necessita de especialista para aquisição de conhecimento [Bulcão Neto & Pimentel, 2003a]. A ontologia Time apresenta as seguintes diretrizes de projeto:

1. Utilização de intervalos de tempo como primitivas do modelo, pois na prática, trabalha-se mais com informação temporal contínua que discreta;
2. Representação de relações temporais entre entidades que possuam um componente temporal, como as relações de passado, passado imediato, presente, futuro imediato e futuro;
3. Múltiplas granulosidades de representação de informação temporal, como horas, dias da semana, meses e anos;
4. Associação de informação temporal a conceitos do modelo *SeCoM*, o que permite não apenas inferências sobre os inter-relacionamentos temporais entre entidades de um ambiente de computação sensível a contexto, mas também o registro do histórico de mudanças em informações de contexto relacionadas a uma entidade em particular.

A Figura 4.2 ilustra a ontologia Time e suas principais classes, atributos e relações. A primeira versão desta ontologia foi publicada no início do desenvolvimento do presente trabalho em Bulcão Neto & Pimentel [2003a].

Figura 4.2: Ilustração da ontologia Time.

4.2.1 Descrição semântica

A principal classe da ontologia apresentada na Figura 4.2 é a classe *TemporalThing*, que descreve qualquer tipo de entidade que contenha uma extensão temporal, quer seja uma extensão por instante de tempo (classe *InstantThing*), quer seja uma extensão por intervalo de tempo (classe *IntervalThing*). Daí, a classe *TemporalThing* ser formada pela união (owl:unionOf) das classes *InstantThing* e *IntervalThing*. Dessa forma, é possível representar a partir do modelo *SeCoM* quaisquer entidades que possuam uma extensão temporal, quer seja um instante de tempo, quer seja um intervalo de tempo [Bulcão Neto & Pimentel, 2004; 2005].

Duas relações temporais básicas são possíveis entre indivíduos da classe *TemporalThing*: a relação temporal *before* e sua relação inversa (owl:inverseOf) *after*. Por meio dessas relações é possível determinar, em sua forma mais básica, a ordem temporal de entidades que contenham uma extensão temporal.

Indivíduos da classe *TemporalThing* contêm atributos que delimitam seu início e término temporais: os atributos *beginPointOf* e *endPointOf*, respectivamente. Os valores que esses atributos assumem são indivíduos da classe *InstantThing*. Indivíduos da classe *IntervalThing* possuem valores diferentes para os atributos *beginPointOf* e *endPointOf*; para indivíduos da classe *InstantThing*, tais valores são idênticos.

Quanto à classe *InstantThing*, a relação temporal *coOccurs* descreve a ocorrência simultânea de indivíduos dessa classe. A relação temporal *insideOf* descreve que

indivíduo(s) da classe `InstantThing` pode(m) ser parte de um indivíduo da classe `IntervalThing`. De forma similar, a relação temporal `temporallyContains`, inversa de `insideOf`, descreve que um indivíduo da classe `IntervalThing` engloba um ou mais indivíduos da classe `InstantThing`. Dessa forma, é possível descrever relações temporais entre entidades cuja extensão temporal são representadas por instantes e intervalos de tempo.

Utilizando como embasamento as relações temporais básicas definidas na teoria de Allen [1983], são definidas 13 (treze) relações temporais entre indivíduos da classe `IntervalThing`, conforme apresenta a Tabela 4.1 a seguir.

Tabela 4.1: Relações temporais entre indivíduos A e B da classe `IntervalThing`.

Relação temporal	Ilustração	Relação temporal inversa	Ilustração
A intCoOccurs B	AAAA BBBB		
A intPrecedes B	AAAA BBBB	A intFollows B	BBBB AAAA
A intMeets B	AAAABBBB	A intMetBy B	BBBBAAAA
A intContains B	AAAA BB	A intDuring B	BBBB AA
A intStarts B	AAAA BB	A intStartedBy B	BBBB AA
A intFinishes B	AAAA BB	A intFinishedBy B	BBBB AA
A intOverlaps B	AAAA BBBB	A intOverlappedBy B	BBBB AAAA

O conjunto das 13 relações temporais apresentadas atende à diretriz 2 do projeto da ontologia `Time`. A relação `intPrecedes`, por exemplo, descreve que um intervalo de tempo A ocorreu antes, porém não imediatamente antes, em relação a um intervalo de tempo B (passado); `intMeets` descreve que um intervalo de tempo B ocorreu imediatamente após um intervalo de tempo A (passado imediato); `intCoOccurs` descreve a simultaneidade de dois intervalos de tempo (presente), assim como a relação `intContains`, embora seus respectivos limites temporais (`beginPointOf` e `endPointOf`) sejam diferentes; `intMetBy` descreve a relação inversa de `intMeets`, ou seja, futuro imediato com respeito a dois intervalos de tempo; por fim, `intFollows` descreve a relação inversa de `intPrecedes`, ou seja, futuro não imediato entre dois intervalos de tempo.

Existem ainda 02 relações temporais entre indivíduos da classe `IntervalThing`. A relação `intNonOverlap` descreve indivíduos que nunca se sobrepõem no tempo; esta relação tem como sub-propriedades (`rdfs:subPropertyOf`) as relações `intPrecedes` e `intMeets`. A relação `intStartsOrDuring` descreve indivíduos que não apresentam intersecção temporal nos seus respectivos tempos de término (`endPointOf`). Ambas relações temporais são importantes para verificar se entidades de extensão temporal intervalar, como uma reunião e uma palestra, podem se sobrepor no tempo.

A ontologia Time representa instantes de tempo por meio da classe `TimeInstant`, que é subclasse (`rdfs:subClassOf`) da classe `InstantThing`. Um instante de tempo é definido neste trabalho como um ponto inextensível na linha do tempo, ou seja, não faz sentido descrever a duração de um instante de tempo. Por outro lado, intervalos de tempo (diretriz 1 de projeto) são representados na ontologia Time por meio da classe `TimeInterval` que, por sua vez, é disjunta (`owl:disjointWith`) da classe `TimeInstant`. Ou seja, não há indivíduos que representem instantes e intervalos de tempo simultaneamente.

Por serem respectivamente subclasses de `InstantThing` e `IntervalThing`, as classes `TimeInstant` e `TimeInterval` herdam todos os atributos e relações definidos para essas classes. A separação entre elementos que possuem extensão temporal e os próprios instantes e intervalos de tempo permite que outros elementos possam ser definidos a partir da extensão temporal que apresentam, tais como a descrição de eventos temporais realizada na ontologia Temporal Event, descrita na seção seguinte, o que permite assim atender à diretriz 4 de projeto da ontologia Time.

Atendendo a sua diretriz 3 de projeto, a ontologia Time fornece mecanismos que permitem representar instantes e intervalos de tempo sob múltiplas formas. Para descrever um instante de tempo específico, é possível utilizar o atributo `instantCalendarClockDataType`, que reusa a notação do tipo de dado data e hora (`xsd:dateTime`) do padrão XML Schema [Biron & Malhotra, 2004]. É possível também utilizar a relação `instantCalendarClock`, que descreve data e hora de forma particionada por meio de atributos da classe `CalendarClockDescription`. Esses atributos permitem descrever ano, semestre, mês do ano, mês, dia da semana, semana, dia, hora, minuto e segundo, todos conforme as respectivas notações de tipos de dados do padrão XML Schema. As classes `MonthOfYear` e `DayOfWeek` são enumerações (`owl:oneOf`) com indivíduos que representam, respectivamente, cada mês do ano e dia da semana.

Já para descrever a duração de intervalos de tempo, é possível utilizar a notação do tipo de dado duração temporal (`xsd:duration`) do padrão XML Schema, notação essa aceita pelo atributo `intervalDurationDescriptionDataType`. É possível também descrever a duração de intervalos temporais de maneira particionada, segundo a unidade temporal que se deseja através da relação `intervalDurationDescriptionOf`. Esta relação conecta-se à classe `DurationDescription` que contém atributos para descrever duração temporal em termos de anos, semestres, meses, dias, semanas, horas, minutos e segundos, todos em conformidade com as respectivas notações de tipos de dados do XML Schema.

4.2.2 Metodologia de desenvolvimento

Para o desenvolvimento da ontologia Time foi utilizada a metodologia *ONIONS* de reúso de ontologias, apresentada na Seção 3.2.2. Seguindo o processo de desenvolvimento proposto por essa metodologia, primeiro foram definidos e coletados

os termos relevantes para a construção da ontologia Time. Essa primeira fase foi realizada com base no reúso de definições de duas ontologias da Web Semântica: a SUMO e a *OWL-Time*, ambas apresentadas na Seção 3.4.

Para fornecer a fundamentação teórica e formal necessária para a construção da ontologia Time, foi utilizada a teoria da Álgebra Temporal de Allen descrita em Allen [1983; 1984]; Allen & Ferguson [1994]. A teoria de Allen é a mais referenciada com respeito a modelos de informação temporal e auxiliou bastante na procura por definições locais dos termos levantados, bem como no projeto de classes de mais alto nível da ontologia em questão.

4.3 Ontologia Temporal Event

A Figura 4.3 descreve a ontologia Temporal Event e suas principais classes, atributos e relações. Apesar de não ter sido considerada no início do desenvolvimento deste trabalho [Bulcão Neto & Pimentel, 2003a; 2004] como uma das ontologias que formariam o modelo *SeCoM*, a ontologia Temporal Event tem o papel de complementação da modelagem de informação temporal realizada pela ontologia Time, descrita na seção anterior.

Figura 4.3: Ilustração da ontologia Temporal Event.

4.3.1 Descrição semântica

Em suma, a ontologia Time modela elementos que apresentam extensão temporal na forma de instantes ou intervalos de tempo, bem como as possíveis relações temporais que podem existir entre esses elementos. A ontologia Temporal Event é uma extensão da ontologia Time ao definir que sua principal classe TemporalEvent, que modela eventos com extensão temporal, é subclasse de time:TemporalThing, onde time: representa o espaço de nomes associado à ontologia Time.

A classe TemporalEvent é composta pela união de dois tipos de eventos temporais: aqueles descritos por instantes de tempo, caso da classe InstantEvent, e os eventos descritos por intervalos de tempo, que são representados pela classe IntervalEvent.

Além de todos os atributos e relações temporais herdados de time:TemporalThing, a classe TemporalEvent apresenta os seguintes atributos: startDateTime e endDateTime, que representam, respectivamente, os instantes temporais de início e término de um indivíduo da classe de eventos temporais; eventDuration que modela a duração de um evento temporal com valor oriundo da classe time:DurationDescription; isPeriodic cujo valor booleano serve para indicar se um evento temporal segue uma periodicidade ou não; e eventFrequency que representa a frequência que um evento temporal segue.

A frequência temporal de eventos é expressa por meio da classe FrequencyDescription. O atributo numberOfTimes expressa o número de vezes que um evento é repetido a cada intervalo de tempo. Seu respectivo valor segue a notação de números inteiros não-negativos (xsd:nonNegativeInteger) do XML Schema. Já o atributo repeatingDuration descreve a duração do intervalo de tempo com que um evento é repetido. Por exemplo, combinando esses dois atributos, é possível representar a periodicidade de eventos que se repetem duas vezes por semana ou uma vez por mês.

As classes InstantEvent e IntervalEvent herdam todos os atributos e relações temporais das classes time:InstantThing e time:IntervalThing, respectivamente, uma vez que são subclasses destas. Ou seja, é possível relacionar indivíduos da classe InstantEvent a um indivíduo de IntervalEvent por meio da relação temporal time:temporallyContains. O mesmo se aplica a indivíduos da classe IntervalEvent que podem ser ordenados temporalmente entre si por meio das relações apresentadas na Tabela 4.1.

4.3.2 Metodologia de desenvolvimento

A ontologia Temporal Event também foi construída segundo as atividades propostas pela metodologia ONIONS de reúso de ontologias.

As ontologias SUMO e *OWL-Time* também serviram de base para a definição e coleta dos termos da ontologia Temporal Event. Sua base teórica e formal também é oriunda da Álgebra Temporal de Allen, que auxiliou principalmente no projeto de classes de mais alto nível da ontologia em questão.

4.4 Ontologia Spatial

Como a maioria das ontologias do modelo *SeCoM*, a ontologia *Spatial* foi inicialmente construída para representar informações, neste caso de localização, relacionadas ao domínio de ensino universitário. Em suas primeiras versões, esta ontologia continha conceitos como salas de aula e de estudos, laboratório de ensino, etc [Bulcão Neto & Pimentel, 2004].

Ao decidir por um modelo independente de domínio para computação sensível a contexto, o autor deste trabalho passou a construir a ontologia *Spatial* para representar informações de localização que abrangessem uma maior variedade de aplicações sensíveis a contexto. A ontologia *Spatial* apresenta as seguintes diretrizes de projeto:

1. Representação da localização de entidades em um espaço físico ou virtual;
2. Representação de relações mereológicas entre espaços físicos ou virtuais;
3. Representação de relações espaciais entre espaços físicos ou virtuais;
4. Múltiplas representações de informação de localização para espaços físicos;
5. Associação de informação de localização a conceitos do modelo *SeCoM*, no sentido de permitir inferências sobre relacionamentos espaciais entre entidades de um ambiente de computação sensível a contexto.

A Figura 4.4 ilustra a ontologia *Spatial* e suas principais classes, atributos e relações [Bulcão Neto & Pimentel, 2005; 2006a].

4.4.1 Descrição semântica

A principal classe da ontologia apresentada na Figura 4.4 é a classe *SpatialThing*, que descreve qualquer tipo de entidade que contenha uma extensão espacial, quer seja uma extensão de espaço físico (classe *PhysicalLocation*), quer seja uma extensão de espaço virtual (classe *VirtualLocation*). Daí, a classe *SpatialThing* ser formada pela união das classes *PhysicalLocation* e *VirtualLocation*. Dessa forma, é possível representar a partir do modelo *SeCoM* quaisquer entidades que possuam uma extensão espacial, seja um espaço físico (ex: sala de reunião) ou virtual (ex: sala de *chat*) [Bulcão Neto et al., 2005b]. Assim, atende-se à diretriz 1 de projeto da ontologia *Spatial*. A separação entre elementos que possuem extensão espacial e as próprias localidades físicas e virtuais permite que outros elementos possam ser definidos a partir desta ontologia, tais como a descrição de eventos espaciais realizada na ontologia *Spatial Event*, descrita na seção seguinte, o que permite assim atender à diretriz 5 de projeto da ontologia *Spatial*.

Figura 4.4: Ilustração da ontologia Spatial.

Duas relações mereológicas são definidas entre indivíduos da classe *SpatialThing*: a relação *isPartOf* e sua relação inversa *contains*. A relação *isPartOf* é caracterizada como transitiva (*owl:TransitiveProperty*), ou seja, se um indivíduo A é parte de um indivíduo B que, por sua vez, é parte de um indivíduo C, todos indivíduos da classe *SpatialThing*, então pode-se afirmar que A é parte de C. O mesmo se aplica com respeito à relação *contains*. Ou seja, por meio dessas duas relações é possível determinar relações parte-todo entre entidades com extensão espacial, atendendo à diretriz 2 de projeto.

Indivíduos da classe *SpatialThing* possuem o atributo *hasName* que lhes associam um nome descritivo. O valor que esse atributo assume segue a notação do tipo de dados cadeia de caracteres (*xsd:string*) do padrão XML Schema. Além de herdarem o atributo *hasName* e as relações mereológicas supracitadas, indivíduos da classe *PhysicalLocation* possuem o atributo *hasLocationCoordinates*, que descreve que cada localização física pode apresentar, no máximo, três coordenadas de localização geográfica obtidas a partir da classe *GeographicCoordinates* e suas subclasses *Latitude*, *Longitude* e *Altitude*. O atributo *hasGeoCoordinateValue* armazena o valor dessas três coordenadas geográficas segundo a notação do tipo de dados número real (*xsd:float*) do XML Schema. Coordenadas geográficas baseadas em latitude e longitude necessitam do atributo *hasGeoOrientation* para associar a direção geográfica correspondente (ex: norte, oeste, etc.). A classe *GeographicOrientation* fornece essa informação no formato de uma enumeração de indivíduos que representam pontos cardeais e colaterais.

A classe *PhysicalLocation* possui duas subclasses: *GeographicLocation* e *AdministrativeRegion*. A classe *GeographicLocation* representa espaços físicos delimitados ou não por um prédio, caso dos indivíduos das classes *IndoorLocation* e *OutdoorLocation*, respectivamente. A classe *GeographicLocation* fornece as relações espaciais *isConnectedTo*, *overlapsSpatially* e *meetsSpatially* para descrever espaços físicos quanto as suas adjacências e sobreposições. Todas essas relações espaciais são modeladas como simétricas, ou seja, se um espaço físico A é adjacente a um espaço físico B, então este é também adjacente ao espaço físico A. Dessa forma, atende-se à diretriz 3 de projeto da ontologia em questão.

A classe *OutdoorLocation* possui, basicamente, as subclasses *Street* e *Park*, cujos indivíduos representam ruas e praças, respectivamente. A classe *IndoorLocation* possui quatro subclasses: a que modela estruturas físicas como prédios (classe *Building*), a que representa andares de um prédio (classe *Floor*), a que modela salas (classe *Room*) e a que representa corredores (classe *Corridor*). Existem entre essas quatro classes mencionadas restrições quanto à relação mereológica *contains* e à relação espacial *isConnectedTo*: em alguns casos (*owl:someValuesFrom*), indivíduos da classe *Building* contêm indivíduos da classe *Room* que, por sua vez, também podem conter e estar conectados a indivíduos da mesma classe. Em alguns casos, indivíduos da classe *Corridor* estão conectados a indivíduos da classe *Room*, enquanto que indivíduos da classe *Floor* eventualmente contêm indivíduos das classes *Room* e *Corridor*.

A classe *AdministrativeRegion*, subclasse de *PhysicalLocation*, representa espaços físicos sob a visão de divisões administrativas, como cidade, estado ou província e país. Estes últimos são representados, respectivamente, pelas subclasses *City*, *StateOrProvince* e *Country*. É possível relacionar regiões administrativas entre si por meio dos atributos *isGeographicSubRegion* e *hasGeographicSubRegion*. Indivíduos da classe *AdministrativeRegion* podem relacionar-se a indivíduos da classe *GeographicLocation* por meio da relação *hasLocationAddress* e sua inversa *isLocationAddressOf*. Dessa forma, é possível associar espaços físicos, como prédios e ruas, a suas respectivas regiões administrativas, como cidade, estado e país. A decisão pela criação da classe *AdministrativeRegion* dá-se em virtude da diretriz 4 de projeto da ontologia *Spatial*.

4.4.2 Metodologia de desenvolvimento

A metodologia *ONIONS* de reúso de ontologias foi utilizada para o desenvolvimento da ontologia *Spatial*. Duas ontologias da Web Semântica contribuíram fortemente com suas respectivas definições de termos que compuseram a ontologia *Spatial*: a *OpenCyc* e a *SWEET*, ambas apresentadas na Seção 3.4.

Na busca por teorias que fornecessem uma fundamentação teórica à ontologia em questão, foi feito um estudo a respeito de relações mereológicas [Casati & Varzi, 1999],

por exemplo. A *WordNet* também contribuiu bastante com sua base de informações léxicas com respeito à definição dos termos levantados, bem como em relação ao projeto das classes de nível superior da ontologia em questão [Miller, 1998]. *WordNet* é uma base de dados léxicos que modela relacionamentos entre conceitos, como sinônimos, antônimos, etc. e é muito utilizada para aplicações de recuperação de informação e tradutores automáticos.

4.5 Ontologia Spatial Event

A Figura 4.5 descreve a ontologia Spatial Event e suas principais classes, atributos e relações. Embora não tenha sido idealizada no início do desenvolvimento deste trabalho como uma das ontologias que formariam o modelo *SeCoM* [Bulcão Neto & Pimentel, 2003a; 2004], a ontologia Spatial Event tem o papel de complementação da modelagem de informação de localização realizada pela ontologia Spatial, descrita na seção anterior.

Figura 4.5: Ilustração da ontologia Spatial Event.

4.5.1 Descrição semântica

A ontologia Spatial modela elementos que apresentam uma extensão espacial na forma de espaços físicos ou virtuais, bem como representa as possíveis relações espaciais e mereológicas que possam existir entre esses elementos. A ontologia

Spatial Event é uma extensão da ontologia Spatial ao definir que como sua principal classe SpatialEvent, que modela eventos com extensão espacial e é subclasse de loc:SpatialThing, onde loc: representa o espaço de nomes associado à ontologia Spatial. Além de todos os atributos e relações espaciais e mereológicas herdados de loc:SpatialThing, a classe SpatialEvent também contém o atributo isLocatedIn, que representa a localização de um evento em relação a um espaço físico ou virtual.

A classe SpatialEvent é composta pela união de dois tipos de eventos espaciais: aqueles cuja extensão está na forma de espaços físicos, caso da classe PhysicalEvent, e os eventos de extensão espacial virtual, representados pela classe VirtualEvent. Ambas as classes herdam os atributos e relações das classes loc:PhysicalLocation e loc:VirtualLocation, respectivamente, uma vez que são subclasses destas. Com as classes PhysicalEvent e VirtualEvent é possível representar eventos que ocorrem em espaços físicos, como reuniões e aulas, bem como eventos que ocorrem em espaços virtuais, como o envio de mensagens em uma sala de *chat*.

4.5.2 Metodologia de desenvolvimento

Assim como a ontologia Spatial, a ontologia SpatialEvent também foi construída através da metodologia *ONIONS* de reúso de ontologias. No entanto, apenas a ontologia *OpenCyc* e o projeto *WordNet* contribuíram com definições para os termos que formam a ontologia SpatialEvent.

4.6 Ontologia Actor

A ontologia Actor modela o conjunto de entidades, chamadas atores, que podem realizar ações em um ambiente de computação sensível a contexto. Como essa ontologia deveria representar inicialmente atores relacionados ao domínio de ensino universitário, em sua primeira versão descrita em Bulcão Neto & Pimentel [2004], Actor era composta de conceitos que descreviam o perfil de atores sob a visão de: relacionamento social, papel social, grau de experiência sobre um assunto específico, informações de contato, artefatos produzidos e participação em projetos.

Após a mudança de foco de uma ontologia para o domínio de ensino universitário para uma ontologia de atores independente de domínio, todas as informações que descreviam o perfil de atores foram modularizadas de forma a representar as ontologias de apoio do modelo *SeCoM*, conforme ilustra a Figura 4.1. Dessa forma, a ontologia Actor, descrita na Figura 4.6, modela apenas classes, atributos e relações básicas com respeito à descrição de atores, conforme publicado em Bulcão Neto & Pimentel [2005]. O conjunto de ontologias de apoio à descrição do perfil de atores do modelo *SeCoM* é descrito no Apêndice A.

Figura 4.6: Ilustração da ontologia Actor.

4.6.1 Descrição semântica

Segundo a Figura 4.6, a principal classe da ontologia é a classe Actor, que busca representar de maneira genérica os variados tipos de atores de um ambiente de computação sensível a contexto. A classe Actor especifica três tipos de atores, podendo ser estendido segundo as necessidades de uma aplicação: pessoas (classe Person, grupo (classe Group) e organização (classe Organization).

Indivíduos da classe Actor possuem um nome que os designa por meio do atributo hasName, de valor xsd:string do padrão XML Schema, bem como podem ou não participar de algum grupo por meio da relação isMemberOf. Essa relação tem hasGroupMember como sua relação inversa, de maneira que a cardinalidade mínima (owl:minCardinality = 1) de ambas as relações descreve que um grupo pode conter um ou mais atores e um ator pode ser membro de um ou mais grupos.

A classe Group tem PersonGroup como subclasse, que descreve um grupo cujos membros são todos (owl:allValuesFrom) indivíduos da classe Person. A classe Person representa os atores do tipo pessoa que possuem, basicamente, atributos que descrevem dados pessoais (hasSurname, hasFirstName e hasBirthday) e relações com indivíduos da classe Group que descrevem o criador de grupos (maker) e sua relação inversa made.

Indivíduos da classe Organization são aqueles atores que representam organizações ou instituições. É possível descrever a hierarquia de organizações entre si por meio da relação transitiva isSubOrganizationOf. Sem muitos detalhes, são representados quatro tipos de organizações: comerciais (classe CommercialOrganization), governamentais

(classe *GovernmentOrganization*), sem fins lucrativos (classe *NonProfitOrganization*) e educacionais (classe *EducationalOrganization*). Definir atributos e/ou relações para esses tipos de organizações, bem como a definição de novos tipos cabe ao desenvolvedor para atender aos seus requisitos específicos de sua aplicação.

4.6.2 Metodologia de desenvolvimento

Para o desenvolvimento da ontologia Actor foi utilizada a metodologia *ONIONS* de reúso de ontologias. Para auxiliar na coleta dos termos relevantes para a construção dessa ontologia, foram reusados termos das ontologias *OpenCyc* e *FOAF*, ambas apresentadas no Capítulo 3, Seção 3.4, e também do padrão de metadados *vCard*, apresentado na Seção 3.1. Tanto as ontologias quanto o padrão de metadados citados contribuíram com o projeto de hierarquização das classes da ontologia Actor. A base de dados *WordNet* também teve papel importante quanto à definição dos termos coletados.

4.7 Ontologia Device

Por ter sido construída após a decisão de desenvolver um modelo ontológico de informações contextuais independente de domínio, o processo de desenvolvimento da ontologia Device não sofreu grandes modificações. Em suma, essa ontologia deveria descrever dispositivos computacionais utilizados para captura e acesso de informações de um ambiente de computação sensível a contexto com o objetivo de promover a adaptação de interfaces com o usuário segundo essas descrições. Para atender a esse objetivo, a ontologia Device apresenta as seguintes diretrizes de projeto:

1. Descrição de dispositivos computacionais quanto aos seus componentes de hardware e software;
2. Representação de relações mereológicas entre componentes de dispositivos computacionais;
3. Representação de dispositivos e tecnologias de computação móvel, comuns em cenários de computação sensível a contexto.

A Figura 4.7 ilustra a ontologia Device e suas principais classes, atributos e relações [Bulcão Neto & Pimentel, 2005]. Por razões de espaço, alguns trechos específicos da ontologia Device, que não constam na Figura 4.7, são apresentadas à medida em que sua semântica é descrita.

Figura 4.7: Ilustração da ontologia Device.

4.7.1 Descrição semântica

A classe Device possui um conjunto de atributos que descreve dispositivos computacionais em geral: deviceModel, que armazena uma cadeia de caracteres (xsd:string) referente ao modelo do dispositivo em questão; deviceStatusOn, um atributo de valor booleano (xsd:boolean) que descreve se um dispositivo está ligado ou não; e os atributos isTextInputCapable, isVoiceInputCapable e isSoundOutputCapable, que descrevem, respectivamente, se um dispositivo tem suporte à entrada de texto e de voz, bem como o suporte à saída de áudio.

Atendendo à diretriz 2 de projeto, são modeladas relações parte-todo entre um dispositivo (classe Device) e seus componentes (classe DeviceComponent): a relação isPartOfDevice interliga um componente ao dispositivo do qual faz parte, enquanto que hasDeviceComponent faz a relação inversa. Indivíduos da classe de componente de dispositivo podem ser de dois tipos disjuntos: componente de hardware, identificado pela classe HardwareComponent, ou componente de software, indicado pela classe SoftwareComponent. Dessa forma, a diretriz 1 de projeto é atendida.

Todo indivíduo classificado como componente de hardware possui o atributo componentModelName, que descreve o nome do modelo do componente em questão, bem como a relação operatingSystemCompatible, que indica com qual sistema operacional (classe OperatingSystem) esse componente é compatível. Na ontologia Device, sistemas operacionais são subclasses de componentes de software.

Com respeito a componentes de hardware, existem 13 subclasses, a saber:

1. a classe CPU descreve o processador principal de um dispositivo quanto ao seu tipo e frequência de operação (`cpuType` e `cpuFrequency`, respectivamente) e quantidade de memória *cache* interna (`internalCacheSize`);
2. a classe `AudioInterfaceCard` descreve as placas de áudio de um dispositivo computacional. São representadas suas características de suporte a áudio tridimensional (`has3DAudioSupport`), som estéreo (`isStereoEnabled`), taxas de amostragem (`samplingRateSupported`), codificadores de entrada de áudio (`audioInputEncoderSupported`) e o número de *bits* por amostragem de áudio (`bitsPerSamplingSupported`);
3. a classe `Display` descreve as telas ou monitores de dispositivos. São considerados os tamanhos de tela em pixels e em caracteres (`screenSize` e `screenSizeChar`, respectivamente), a razão de aspecto em pixels (`pixelAspectRatio`) e a capacidade de exibição de cores (`colorCapability`);
4. a classe `Keyboard` descreve os tipos de teclados que dispositivos computacionais podem oferecer. São considerados o tipo do teclado, como o de telefone ou de microcomputador convencional (`keyboardType`), o suporte a teclas com função especial (`isSoftKeysCapable`), o número de teclas com função especial (`numberOfSoftKeys`) e as páginas de código de caracteres de entrada e saída suportadas pelo dispositivo (`inputCharSetSupported` e `outputCharSetSupported`, respectivamente);
5. a classe `VideoCaptureCard` descreve indivíduos do tipo placa de captura de vídeo com base nas características de (número de bits por) taxa de amostragem suportada (`bitsPerSamplingSupported` e `samplingRateSupported`, respectivamente), número de quadros por segundo (`framesPerSecondSupported`), número de *pixels* por linha de resolução (`pixelsPerLineSupported`), tipos de entrada de vídeo analógico e de decodificação de TV analógica aceitos (`analogVideoInputSupported` e `TVDecoderSupported`, respectivamente), suporte a mecanismo automático de detecção de sinal de TV analógico (`automaticTVStdDetection`) e tipos de codificadores de saída de vídeo aceitos (`videoOutputEncoderSupported`);
6. a classe `Modem` descreve componentes de hardware do tipo modulador-demodulador existentes em certos dispositivos computacionais. Em suma, são descritas a existência de suporte a fax (`isFaxEnabled`) e as velocidades máximas de transmissão suportadas (`maxSpeedSupported`);

7. a classe `mainMemory` descreve a memória principal de um dispositivo computacional quanto à quantidade de memória total (`mainMemorySize`) e de memória livre (`mainMemoryFreeSize`);
8. para os casos de dispositivos computacionais móveis, a classe `Battery` representa informações de baterias quanto a sua duração de consumo médio (`meanConsumptionDuration`), capacidade total de energia (`isFullyCharged`) e quantidade instantânea de energia (`powerAvailable`);
9. a classe `GraphicsCard` descreve componentes de hardware do tipo placa de vídeo quanto à quantidade de memória de vídeo fornecida (`graphicsMemorySize`) e à quantidade de cores suportada pela placa com respeito ao número de *bits* por *pixel* (`bitsPerPixel`);
10. a classe `Pointer` descreve o mecanismo apontador aceito pelo dispositivo computacional, como *mouse*, caneta eletrônica ou dedo da mão. Basicamente, existem três tipos de resolução aceitos via atributo `pointingResolution`: resolução por caracter, por linha e por *pixel*.
11. a classe `Printing` descreve as características de dispositivos de hardware do tipo impressoras. Essas características incluem a capacidade de impressão a cores (`colorCapability`), tamanho e orientação de papel aceitos, como A4 e retrato (`paperOrientation` e `paperSize`, respectivamente), e resolução e leiaute de impressão aceitos, como 600dpi e “2 pages to 1” (`printingResolution` e `printingLayout`, respectivamente);
12. a classe `SecondaryStorage` representa indivíduos de armazenamento secundário de um dispositivo. Segundo a Figura 4.8, indivíduos dessa classe têm como atributos `storageCapacity` e `storageFreeCapacity`, que medem, respectivamente, a quantidade total e livre de espaço de armazenamento secundário. Dependendo do tipo de mídia utilizada para armazenamento (classe `StorageMedia`), podem existir dois tipos de armazenamento secundário disjuntos entre si: armazenamento óptico (classe `OpticalMediaDrive`), quando a relação `storageMedia` assume o valor de `OpticalMedia` para a classe `StorageMedia`; e armazenamento magnético (classe `MagneticMediaDrive`), quando a relação `storageMedia` assume o valor de `MagneticMedia` para a classe `StorageMedia`.

A classe `OpticalMedia` é uma enumeração dos indivíduos CD e DVD. Quando o atributo `storageMedia` de dispositivos de armazenamento secundário óptico (`OpticalMediaDrive`) tem o valor CD ou DVD, esse dispositivo pode ser um *drive* de CD ou um *drive* de DVD. O atributo booleano `canReadAndWrite` descreve a capacidade de leitura e gravação desse tipo de dispositivo. Sempre que seu valor

`isWirelessEnabled` assume o valor verdadeiro, caracterizam-se as interfaces de rede sem fio *WirelessCard*. Tais interfaces podem ser de três tipos: interfaces *Wi-Fi* (*WiFiCard*), de comunicação via infra-vermelho (*IrDACard*) e com suporte a *Bluetooth* (*BluetoothCard*).

Interfaces do tipo *WiFiCard* têm a relação `MACProtocolSupport` com os padrões IEEE da série 802.11 como valores. Quanto às interfaces *IrDACard*, o atributo `IrDASpeedRate` descreve sua taxa de velocidade de transmissão. Interfaces do tipo *BluetoothCard* são descritas segundo a potência de saída do rádio transmissor (`radioOutputPower`), a faixa de frequência de operação (`frequencyBand`), a versão de compatibilidade (`bluetoothVersionCompatibility`) e o máximo valor de alcance do sinal de transmissão (`maxRadiusRange`).

Figura 4.9: Ilustração das classes para interface de rede de um dispositivo computacional da ontologia Device.

Por fim, quanto aos componentes de software, estes podem ser descritos por uma série de atributos: `softwareName` e `softwareVersion` descrevem, respectivamente, o nome e a versão do software em questão; `isJavaEnabled` indica se o dispositivo tem suporte à linguagem Java; `JVMInstalled` indica as versões de máquina virtual Java instaladas no dispositivo; `JavaPlatformsSupported` indica as plataformas Java aceitas pelo dispositivo, como J2SE e J2EE; `acceptDownloadableSoftware` indica se o usuário do dispositivo permite o *download* de software; `downloadableSoftwareSupport` indica os tipos de software que o usuário aceita para *download*; `MIMEtypesSupported` descreve a lista de tipos MIME [Freed & Borenstein, 1996] que o usuário do dispositivo aceita, como *text/plain* e *text/html*; `documentLanguageSupported` descreve a lista de idiomas preferidos do usuário para os documentos contidos no dispositivo; `charsetSupported` descreve a lista de páginas de código que o dispositivo aceita, como *ISO 8859* e

Unicode; e *transferEncodingSupported* descreve a lista de codificações de transferência suportadas por um dispositivo segundo a RFC 2045 [Freed & Borenstein, 1996].

Figura 4.10: Ilustração das classes para componente de software de um dispositivo computacional da ontologia Device.

Além da classe *OperatingSystem*, existe outra subclasse de componentes de software: a classe *UserAgent*. Essa classe descreve principalmente indivíduos do tipo navegador Web e possui a seguinte lista de atributos: *isTablesCapable*, *isImageCapable* e *isFramesCapable*, que indicam se o navegador é capaz de exibir tabelas, imagens e quadros HTML, respectivamente; *preferenceForFrames*, que indica se o usuário do dispositivo tem preferência por exibir quadros HTML; *isJavaScriptEnabled* e *isJavaAppletEnabled*, que descrevem se o navegador é habilitado para rodar miniaplicativos e *scripts* Java, respectivamente; *javascriptVersionSupported*, que indica as versões de *JavaScript* que o navegador suporta; *HTMLVersionSupported* e *XHTMLVersionSupported*, que indicam as versões das linguagens HTML e XHTML suportadas pelo navegador, respectivamente; e *XHTMLModuleSupported*, que informa que módulos da linguagem XHTML são suportadas pelo navegador do dispositivo em questão.

A diretriz 3 de projeto da ontologia Device é atendida por meio de conceitos que descrevem perfis de dispositivos móveis, por exemplo, as classes *Battery*, *Pointer* e *BluetoothCard* e seus respectivos atributos e relações.

4.7.2 Metodologia de desenvolvimento

A ontologia Device foi construída segundo as atividades previstas na metodologia ONIONS de reúso de ontologias. A coleta dos termos relevantes para a construção dessa ontologia baseou-se em termos da ontologia CC/PP, apresentada na Seção 3.4, bem como na especificação *UAProf* (*User Agent Profile*) [OMA, 2001]. Embora *UAProf* também descreva perfis de dispositivos computacionais como a ontologia CC/PP, essa especificação não é um padrão do W3C, mas sim do consórcio *Open Mobile Alliance*.

Para auxiliar tanto no projeto de hierarquização das classes da ontologia Device quanto na definição das classes de mais alto nível foi realizado um estudo sobre relações mereológicas. Relações parte-todo não são tratadas pelas especificações que servem de base para a construção da ontologia Device e, ao mesmo tempo, são importantes para a descrição de composição de dispositivos computacionais.

4.8 Ontologia Activity

Desde o início do desenvolvimento deste trabalho, o autor considerava que, em virtude da dependência de informações provenientes das ontologias supracitadas, a ontologia Activity deveria ser a última ontologia a ser desenvolvida [Bulcão Neto & Pimentel, 2003a]. Mesmo tendo sido construída após a decisão de desenvolver um modelo ontológico de informações contextuais independente de domínio [Bulcão Neto & Pimentel, 2005], o processo de desenvolvimento da ontologia Activity sofreu várias modificações, principalmente em função da heterogeneidade de informações que esta envolve, bem como da dificuldade de relacionar essas informações.

O objetivo da ontologia Activity é descrever ações que atores realizam ou fazem acontecer em um ambiente de computação sensível a contexto. Para atender a esse objetivo, a ontologia Activity apresenta as seguintes diretrizes de projeto:

1. Descrição de atividades quanto ao tempo em que ocorrem;
2. Descrição de atividades quanto ao espaço em que acontecem;
3. Descrição de atividades em função dos atores que as realizam;
4. Descrição de atividades que envolvam atores e dispositivos computacionais;
5. Diferenciação entre atividades que ocorrem espontaneamente daquelas que ocorrem de maneira prevista.

A Figura 4.11 ilustra a ontologia Activity e suas principais classes, atributos e relações [Bulcão Neto & Pimentel, 2005]. Assim como pela Figura 4.1, é possível verificar pela Figura 4.11 que a ontologia Activity importa as principais ontologias do modelo *SeCoM*: Actor, Device, TemporalEvent e SpatialEvent¹.

4.8.1 Descrição semântica

Segundo a Figura 4.11, a principal classe da ontologia é a classe Activity, que descreve atividades como eventos espaço-temporais que, por sua vez, são descritos por meio da classe SpatioTemporalEvent. Esta classe representa a união dos indivíduos

¹Conseqüentemente, Activity importa também as ontologias Time e Spatial.

Figura 4.11: Ilustração da ontologia Activity.

pertencentes às classes `tEve:TemporalEvent` e `sEve:SpatialEvent`, onde `tEve:` e `sEve:` são, respectivamente, os espaços de nomes das ontologias `TemporalEvent` e `SpatialEvent` importadas pela ontologia `Activity`. A classe `SpatioTemporalEvent` é subclasse de `SpatioTemporalThing`, que descreve quaisquer entidades que contenham extensões espaciais (`loc:SpatialThing`) e temporais (`time:TemporalThing`) simultaneamente.

Como atividades são descritas como eventos espaço-temporais, estas podem valer-se dos mesmos atributos e relações válidos para eventos espaciais e para eventos temporais. Dessa maneira, é possível, dentre outros, situar atividades quanto a relações espaciais e mereológicas entre espaços físicos virtuais ou reais, ou mesmo situar atividades quanto a instantes e intervalos temporais que descrevam passado, presente e futuro. Assim, as diretrizes 1 e 2 de projeto da ontologia são atendidas.

Todo indivíduo da classe `Activity` possui uma lista geral de atores (relação `hasParticipant`) que participam da atividade em questão. Como na maioria dos casos os atores de uma atividade são pessoas, foi criada uma restrição de valor (`owl:someValuesFrom`) à relação `hasParticipant` para aceitar como válidos indivíduos da classe `act:Person`. A relação `isEngagedIn`, inversa de `hasParticipant`, permite descrever que atores estão engajados na realização de uma determinada atividade. Esses conceitos permitem que se atenda à diretriz 3 de projeto da ontologia `Activity`.

Atividades podem ser descritas textualmente por meio dos atributos `hasLongDescription`, para longas descrições, e `hasSummary`, para descrições breves. Para situações

em que atividades tenham um prazo de validade associado (ex: uma conferência), foi criado o atributo booleano `validationStatus`. A existência de periodicidade de uma atividade é dada pelo atributo `isPeriodicActivity`, que quando assume valor booleano verdadeiro faz sentido descrever a frequência (classe `tEve:FrequencyDescription`) com que essa atividade ocorre por meio da relação `activityFrequency`.

Para descrever atividades que incluam ações específicas de uma pessoa foi criada a classe `ColocatedAction`, que possui os seguintes atributos e relações: `actionStartDateTime`, `actionEndDateTime` e `actionDuration`, para descrever os instantes de tempo em que a ação inicia, termina e sua respectiva duração temporal; e `subjectIsLocatedIn` que descreve a localização da pessoa (`subjectOfAction`) que realiza a ação em questão. Com esses conceitos é possível descrever, por exemplo, que “Carlos tem estado na sala D-5 por duas horas no contexto de uma reunião”.

De maneira complementar à classe `ColocatedAction`, foi criada sua subclasse `ColocatedActionOnObject` para descrever ações específicas de uma pessoa utilizando um determinado dispositivo computacional. A relação `objectOfAction` é responsável por descrever esse relacionamento entre uma ação e um objeto dessa ação. Dessa maneira é possível descrever, por exemplo, que “Carlos começou a utilizar a lousa eletrônica da sala F-3 às 19:00 de 31/08/2006 no contexto de uma aula”. Com esses conceitos, a diretriz 4 de projeto da ontologia Activity é atendida.

No intuito de atender à diretriz 5 de projeto da ontologia Activity, foi elaborada o atributo `schedulingStatus`, cujo valor booleano permite classificar uma atividade como espontânea (classe `ImpromptuActivity`) ou agendada (classe `ScheduledActivity`). Essas classes disjuntas permitem descrever atividades que ocorrem de maneira informal, como reuniões em *cocktails*, bem como aquelas atividades planejadas quanto ao espaço e tempo em que devem ocorrer, como palestras de uma conferência.

4.8.2 Metodologia de desenvolvimento

Diferentemente das demais ontologias do modelo *SeCoM*, a ontologia Activity não foi desenvolvida por meio do reúso de ontologias da Web Semântica. Nesse contexto, foi utilizada a metodologia *METHONTOLOGY* para a construção da ontologia Activity, metodologia apresentada no Capítulo 3, Seção 3.2.2.

A escolha pela metodologia *METHONTOLOGY* advém da dificuldade de se encontrar ontologias que abordem de maneira coerente os conceitos de atividades como eventos espaço-temporais, bem como do fato do autor deste trabalho ser novato na tarefa de construir ontologias desde o início.

Para a fase de conceitualização do conhecimento da ontologia Activity, foram utilizados conceitos provenientes do padrão de metadados *iCalendar*, bem como da base de dados léxicos *WordNet*.

4.9 Avaliação do modelo SeCoM

Com relação à avaliação do modelo *SeCoM*, foram inicialmente utilizados critérios comumente utilizados para descrever o nível estrutural e a complexidade lógica de uma ontologia [Brank et al., 2005; Burton-Jones et al., 2005]. Dentre os critérios utilizados para avaliação, incluem-se o tipo (ou dialeto) OWL, a expressividade lógica e os números de triplas, classes, propriedades e instâncias (ou indivíduos). Os critérios tipo OWL e expressividade lógica de uma ontologia são obtidos pela combinação dos axiomas da mesma.

Embora esse tipo de avaliação seja, na maioria das vezes, realizado de forma manual, foi utilizado o programa *Benchmarking.java* distribuído com a máquina de inferência *Pellet 1.3beta2* [Sirin et al., 2006] para coletar as medidas relacionadas aos critérios, como ilustra a Tabela 4.2. O ambiente de hardware e software utilizado para as medições inclui CPU Intel^(R) Pentium^(R)4 2.66 GHz, 1 GiB RAM, sistema operacional Linux Red Hat 7.3, plataforma Java J2SE 1.5 e Eclipse 3.0.

Tabela 4.2: Complexidade lógica e caracterização estrutural das ontologias principais do modelo *SeCoM*. Adaptado de Bulcão Neto & Pimentel [2006a].

Ontologia	OWL	Expressividade	Triplas	Classes	Propriedades	Instâncias
Time	DL	SHOIF(D)	325	10	44	19
Temporal Event	DL	SHOIF(D)	383	14	52	19
Spatial	DL	SHOIN(D)	186	22	14	8
Spatial Event	DL	SHOIN(D)	208	25	15	8
Actor	Lite	ALR+IF(D)	72	9	9	0
Device	DL	ALCHOIF(D)	891	35	92	10
Activity	DL	SHOIN(D)	1681	90	186	37

Segundo a Tabela 4.2, a maioria das ontologias *SeCoM* utiliza construtores da linguagem OWL DL, o que sugere que para processar informações dessas ontologias é necessária uma máquina de inferência que reconheça tais construtores.

Como declarado na Seção 3.3.4, quanto maior a expressividade de uma ontologia OWL, maior é a complexidade computacional desta. A Tabela 4.3 seguinte apresenta uma notação padrão para descrever a expressividade lógica (ou DL) de uma ontologia baseada em Lógica de Descrições, assim como são as ontologias do modelo *SeCoM*.

Por meio do endereço Web <http://www.cs.man.ac.uk/~ezolin/logic/complexity.html> intitulado “Complexity of reasoning in description logics”, foi possível combinar os possíveis construtores DL de maneira a obter a complexidade computacional para processos de inferência sobre ontologias.

É possível verificar que, por utilizar apenas construtores do dialeto OWL Lite, a ontologia Actor é a que possui menor complexidade computacional, no caso, exponencial determinístico, o que corrobora a informação apresentada na Tabela 3.1.

Tabela 4.3: Notação para expressividade lógica de ontologias baseadas em Lógica de Descrições [Baader et al., 2003].

Símbolo	Construtores presentes
AL	Lógica de atributos: conjunção, restrição de valor universal e quantificação existencial limitada
C	Complemento: AL acrescida de disjunção e quantificação existencial completa
R+	Propriedade transitiva
S	Abreviação para ALCR+
H	Hierarquia de propriedades
I	Propriedade inversa
F	Propriedade funcional ou restrição de cardinalidade = 1
O	Nominais, como enumerações de indivíduos ou de valores de dados
N	Restrições de número não-qualificadas, como restrições de cardinalidade > 1
D	Tipos de dados

Entretanto, para o caso de ontologias que usam o dialeto *OWL DL*, a complexidade de tempo para o pior caso é maior — exponencial não-determinístico — quando são usados simultaneamente em uma mesma ontologia os seguintes construtores:

1. declaração de nominais (O), de propriedades inversas (I) e de propriedades funcionais (F), como `owl:oneOf`, `owl:inverseOf` e `owl:FunctionalProperty`;
2. declaração de nominais (O), de propriedades inversas (I) e de restrições de cardinalidade não-qualificadas (N), como `owl:oneOf`, `owl:inverseOf` e `owl:minCardinality` e `owl:maxCardinality` quando seus valores são maiores que 1.

Assim, é possível verificar, a partir da Tabela 4.2, que a complexidade de tempo no pior caso será alta nos casos das ontologias *Time*, *Temporal Event* e *Device*, cujas complexidades são, respectivamente $\text{SHOIF}(D)^2$, $\text{SHOIF}(D)$ e $\text{ALCHOIF}(D)$. Entretanto, a complexidade de tempo será ainda maior, segundo a literatura [Baader et al., 2003], nos casos das ontologias *OWL DL* que possuem expressividade DL com valor $\text{SHOIN}(D)$, caso das ontologias *Spatial*, *Spatial Event* e *Activity*.

Observe que se uma ontologia importa direta (ou indiretamente) outra(s) ontologia(s), o grau de expressividade DL da primeira será o maior dentre todos. No caso da ontologia *Activity*, as ontologias importadas têm expressividade $\text{ALR+IF}(D)$, $\text{ALCHOIF}(D)$, $\text{SHOIF}(D)$ e $\text{SHOIN}(D)$; daí a expressividade DL da ontologia *Activity* ser $\text{SHOIN}(D)$, a mesma da ontologia *Spatial Event* cuja expressividade DL é a maior dentre todas. O critério de expressividade DL é importante para escolher uma máquina de inferência para ontologias. Atualmente, apenas a máquina de inferência *Pellet* é capaz de manipular a máxima expressividade dos construtores do dialeto *OWL DL*.

²Ou seja, a ontologia em questão contém os construtores OWL de lógica de atributos, complemento, propriedades funcional, transitiva e inversa, hierarquia de propriedades, nominais e tipos de dados.

A Tabela 4.4 ilustra as ontologias do modelo *SeCoM* em ordem crescente de complexidade computacional no pior caso, conforme já discutido.

Tabela 4.4: Ordem crescente de complexidade computacional das ontologias principais do modelo *SeCoM*.

Ontologia	OWL	Expressividade	Complexidade
Actor	Lite	ALR+IF(D)	Exponencial determinístico
Device	DL	ALCHOIF(D)	Exponencial não-determinístico
Time	DL	SHOIF(D)	Exponencial não-determinístico
Temporal Event	DL	SHOIF(D)	Exponencial não-determinístico
Spatial	DL	SHOIN(D)	Exponencial não-determinístico
Spatial Event	DL	SHOIN(D)	Exponencial não-determinístico
Activity	DL	SHOIN(D)	Exponencial não-determinístico

Além de ilustrar o tipo OWL e a expressividade lógica, a Tabela 4.2 mostra as medidas coletadas dos números de triplas, classes, propriedades e instâncias das ontologias do modelo *SeCoM*. Essas medidas são interessantes para situações em que se deseja avaliar o desempenho de sistemas de software que utilizarão tais ontologias. Na maioria das situações, quanto maiores os valores dessas medidas, maior será o tempo de resposta total relacionado a um processo de inferência sobre ontologias.

No caso das ontologias *SeCoM*, sistemas que requeiram representar atividades associadas a atores, dispositivos, localização e tempo deverão ter maior tempo total de inferência. Para sistemas que demandem apenas a representação de informações a partir da ontologia Actor o tempo total de inferência deverá ser menor.

A avaliação de desempenho quanto ao uso das ontologias do modelo *SeCoM* por sistemas de software é abordada nos Capítulos 5 e 7, onde são calculados não apenas tempos de resposta totais, mas também tempos intermediários de processos de inferência sobre cada ontologia do modelo. A diferença com respeito à avaliação do modelo *SeCoM* nos Capítulos 5 e 7 é que nestes o modelo é avaliado, respectivamente, por meio de uma infra-estrutura de software e de uma aplicação Web.

4.10 Considerações finais

Este capítulo apresentou o processo de desenvolvimento do modelo ontológico *SeCoM* para informações de contexto. Cada uma das ontologias principais do modelo foi descrita com atenção especial à metodologia de desenvolvimento e às especificações, teorias, ontologias e metadados da Web Semântica utilizados como fundamentação. Por fim, foi mostrado como se deu a atividade de avaliação do modelo *SeCoM* e os critérios utilizados para tal.

Dado que a maioria das ontologias do modelo *SeCoM* foi construída por meio de uma metodologia que explora o reuso de outras ontologias, não foi explicitado por

quais motivos essas ontologias reusadas não foram utilizadas em sua forma natural. Em geral, a maioria dessas ontologias carece de modularidade quanto à utilização de sua informação, por exemplo, as ontologias *OpenCyc*, SUMO e SWEET. Importá-las significaria uma considerável sobrecarga no processo de inferência das ontologias do modelo *SeCoM*.

Outras ontologias não são adequadas para representar a dimensão semântica de informação contextual que modelam. Por exemplo, a ontologia CC/PP não representa relações mereológicas entre componentes de dispositivos computacionais, além de carecer de organização quanto à hierarquia dos conceitos descritos [Eisinger et al., 2005].

A ontologia *OWL-Time*, por sua vez, poderia ser reusada da forma como está publicada em Hobbs & Pan [2006]. O motivo pelo qual o autor construiu a ontologia *Time* deve-se ao fato de que à época a ontologia *OWL-Time* ainda não existia. Daí a maior parte do trabalho na ontologia *Time* ter tido como linha mestra a Álgebra Temporal de James Allen e a ontologia SUMO.

Construído com base na semântica e lógica fornecidas pela linguagem OWL, o modelo *SeCoM* tem complexidade computacional no pior caso previsível mediante o processo de inferência sobre ontologias do modelo. Tal informação pode ser confirmada mediante a verificação da expressividade DL das ontologias que o compõem.

Uma vez que as informações de contexto de um ambiente tenham um modelo semântico, uniforme e padronizado de representação, é necessária uma infra-estrutura de serviços básicos que sejam capazes de processar a semântica e lógica desse modelo. O assunto abordado no capítulo seguinte inclui a construção dessa infra-estrutura de serviços e sua utilização para validação e avaliação do modelo *SeCoM*.

Serviços para Semântica de Informações de Contexto

Na fase de definição deste trabalho, o presente autor visava desenvolver um modelo de informação contextual para apoiar o desenvolvimento de software sensível a contexto no domínio de ensino universitário Bulcão Neto & Pimentel [2003b]. Entretanto, o autor estava ciente de que apenas desenvolver um modelo não lhe dava garantias de que este seria útil, nem de que se saberia como utilizá-lo.

Em virtude disso, o autor propôs desenvolver algoritmos para processamento da semântica de informação contextual associada ao modelo construído. Originalmente, esses algoritmos deveriam ser incluídos em uma infra-estrutura de serviços existente à época da definição deste trabalho chamada *Context Kernel* [Arruda Jr. et al., 2003]. Essa infra-estrutura é composta de serviços Web [W3C, 2006] para o gerenciamento de informações de contexto. Seus mecanismos de armazenamento, consulta e recuperação de informações apóiam-se em um modelo cuja sintaxe XML descreve combinações de informações de contexto na forma de regras com premissas e conclusões, embora nenhum processo de inferência seja realmente utilizado.

Após um longo estudo da infra-estrutura *Context Kernel*, o autor concluiu que muito pouco de seus serviços poderia ser reusado, principalmente em virtude da utilização de um modelo sintático de informação contextual [Bulcão Neto et al., 2004a;b]. Além disso, esse mesmo modelo não é uniforme para as aplicações, uma vez que uma aplicação pode armazenar informações de localização como *sala* e uma outra aplicação pode consultá-la utilizando *room* como parâmetro. Dado que o mecanismo

de consulta da infra-estrutura *Context Kernel* é baseado em combinações léxicas, essa consulta falha, embora do ponto de vista semântico não devesse.

Posteriormente, foram realizados experimentos com a infra-estrutura *Context Kernel* no sentido de integrar aplicações voltadas para ensino colaborativo [Macedo et al., 2005; Jardim et al., 2005a;b]. Todos esses experimentos corroboraram a carência do modelo de representação de informações de contexto da infra-estrutura *Context Kernel* que, conseqüentemente, reflete-se em seus serviços de armazenamento, consulta e recuperação subjacentes. Portanto, cenários que demandam a interpretação da semântica de informações de contexto não podem ser tratados pela versão corrente da infra-estrutura *Context Kernel*.

Com o desenvolvimento do modelo *SeCoM*, discutido no Capítulo 4, deu-se o primeiro passo no sentido de tratar a carência de informação semântica da infra-estrutura *Context Kernel* original. O autor constatou que substituir o modelo XML dessa infra-estrutura pelo modelo semântico *SeCoM* não seria uma solução viável, uma vez que todos os serviços estavam fortemente atrelados ao modelo original. Ao invés de reusar a infra-estrutura *Context Kernel*, o autor optou por desenvolver uma nova infra-estrutura de serviços chamada *Semantic Context Kernel SCK* [Bulcão Neto et al., 2005b], que manipula a semântica explícita de informações de contexto instanciadas a partir de um modelo ontológico, tal como o modelo *SeCoM*. Por meio da infra-estrutura *SCK*, o autor consegue mostrar como o modelo *SeCoM* pode ser utilizado no apoio ao desenvolvimento de sistemas sensíveis a contexto.

Este capítulo apresenta a infra-estrutura *SCK* ao discutir inicialmente as diretrizes de projeto que conduziram seu desenvolvimento. Em seguida, são detalhadas a arquitetura de serviços da infra-estrutura *SCK* e sua relação com as diretrizes de projeto citadas. A partir dos serviços da infra-estrutura *SCK*, são descritos como utilizar o modelo *SeCoM*, bem como a avaliação de desempenho do serviço de inferência utilizando o modelo *SeCoM* como base de testes. Por fim, este capítulo descreve considerações sobre a avaliação do serviço de inferência da infra-estrutura *SCK*, que incluem um conjunto de questões de projeto relacionadas a processos de inferência sobre a semântica ontológica de informação contextual.

5.1 Diretrizes de projeto

Esta seção discute as diretrizes de projeto da infra-estrutura de serviços *SCK* utilizadas para orientar seu processo de desenvolvimento. As diretrizes apresentadas a seguir baseiam-se no arcabouço conceitual definido por Dey [2000] para a construção de software sensível a contexto, conforme apresentado no Capítulo 2, Seção 2.4.

1. A infra-estrutura deve oferecer um conjunto de serviços que seja de utilização para qualquer aplicação sensível a contexto;

2. Os serviços oferecidos pela infra-estrutura devem ser configuráveis no sentido de atender a uma ampla diversidade de requisitos de aplicações sensíveis a contexto;
3. Os serviços oferecidos pela infra-estrutura devem funcionar independentemente do modelo ontológico de informação contextual utilizado;
4. A infra-estrutura deve promover a interoperabilidade semântica entre aplicações sensíveis a contexto ao fornecer-lhes um modelo compartilhado e uniforme de acesso a informações de contexto;
5. Em virtude da heterogeneidade de ambientes de computação sensível a contexto, a infra-estrutura de serviços deve funcionar em múltiplas plataformas de hardware e software.

As seções seguintes descrevem de que maneira as diretrizes supracitadas têm sido atendidas pela arquitetura e implementação da infra-estrutura SCK.

5.2 Arquitetura da infra-estrutura SCK

A Figura 5.1 ilustra a arquitetura da infra-estrutura SCK cujos componentes podem pertencer a um dos seguintes níveis ou camadas: o nível de aplicação e o nível de infra-estrutura.

Figura 5.1: Arquitetura da infra-estrutura Semantic Context Kernel. Adaptado de Bulcão Neto et al. [2005b].

O nível de aplicação representa os componentes do ponto de vista do desenvolvedor ou do usuário de uma aplicação sensível a contexto. Esses componentes comunicam-se com o nível de infra-estrutura ao fornecer-lhe ou solicitar-lhe informações de contexto instanciadas a partir do modelo ontológico subjacente na forma de triplas RDF. Esse mecanismo de comunicação entre componentes dos níveis de aplicação e infra-estrutura atende à diretriz 4 de projeto da infra-estrutura SCK.

Informações de contexto são fornecidas ao nível de infra-estrutura por meio dos componentes *fontes de contexto*, que podem incluir não apenas aplicações, mas também serviços Web [W3C, 2006] e sensores físicos. Serviços Web podem, por exemplo, fornecer informações de contexto referentes a condições climáticas ou a atividades agendadas de um usuário, como descrito por Sheshagiri et al. [2004]. Sensores físicos são amplamente utilizados em computação sensível a contexto, por exemplo, no sistema *CARETM*, apresentado no Capítulo 2, Seção 2.5.

Há situações em que fontes de contexto não representam informações segundo o modelo ontológico subjacente no formato de triplas RDF. Tais situações incluem fontes de contexto não projetadas inicialmente para serem integradas aos serviços da infra-estrutura *SCK*, ou de sensores físicos que, em geral, têm sua própria representação para os tipos de dados que manipulam. Nesses casos, fontes de contexto necessitam de componentes *tradutores de contexto* cuja principal tarefa consiste em converter (ou traduzir) a informação capturada das fontes de contexto para a representação semântica comum entre os componentes dos níveis de aplicação e de infra-estrutura: o modelo de triplas RDF. Vale ressaltar que é possível implementar um tradutor de contexto como parte integrante de uma fonte de contexto, principalmente, quando esta é uma aplicação. No caso de sensores físicos, tradutores de contexto são módulos independentes das fontes de informação contextual a ser convertida para triplas RDF.

Finalizando a apresentação do nível de aplicação, os componentes *consumidores de contexto* fazem uso da informação contextual fornecida pelas fontes de contexto ao nível de infra-estrutura. Consumidores de contexto são representados por aplicações sensíveis a contexto, que utilizam informação contextual para adaptarem seus serviços segundo a situação corrente do usuário. A maneira pela qual consumidores de contexto recuperam informações do nível de infra-estrutura também se dá segundo o formato de triplas RDF.

Com respeito aos componentes do nível de infra-estrutura, apresentados na Figura 5.1, estes fornecem um conjunto de serviços básicos que aplicações sensíveis a contexto podem usufruir para estender suas capacidades, bem como para compartilhar informações de contexto entre si. Os quatro serviços que compõem o nível de infra-estrutura são uma menção à diretriz 1 de projeto supracitada.

O *serviço de descoberta* fornece um mecanismo de anúncio que permite identificar os componentes que fornecem informações de contexto ao nível de infra-estrutura, bem como os próprios serviços oferecidos pela infra-estrutura *SCK*. Vale lembrar que se a informação contextual necessita ser convertida para triplas RDF do modelo subjacente, tradutores de contexto são objeto de identificação pelo serviço de descoberta, caso contrário, fontes de contexto o são. Por meio do mecanismo de anúncio oferecido pelo serviço de descoberta, consumidores de contexto (aplicações)

são capazes de localizar os serviços do nível de infra-estrutura para consulta ou inferência de informações de contexto.

O *serviço de armazenamento de contexto* permite que componentes do nível de aplicação, fontes ou tradutores de contexto, armazenem informação contextual. Como descrito anteriormente, essa informação armazenada é recuperada pelos componentes consumidores de contexto via serviços de consulta ou inferência de contexto. Atendendo à diretriz 2 de projeto da infra-estrutura SCK, o serviço de armazenamento de contexto pode ser configurado quanto ao tipo de armazenamento persistente para informações contextuais: em bases de dados relacionais ou em arquivos.

O *serviço de consulta de contexto* permite que componentes consumidores de contexto consultem informações por meio de uma linguagem declarativa para modelos RDF. Esse serviço oferece a opção de configurar a linguagem de consulta de modelos RDF a ser utilizada com diferentes capacidades de expressão, uma referência à diretriz 2 de projeto da infra-estrutura SCK. Em geral, as expressões de consulta oferecidas por tais linguagens são representadas por meio da comparação de um grafo RDF de entrada, definido pelo usuário, em relação aos grafos de triplas RDF que representam as informações de contexto armazenadas.

Concluindo a lista de componentes do nível de infra-estrutura, o *serviço de inferência de contexto* fornece a consumidores de contexto (ou aplicações) um mecanismo de inferência sobre informações de contexto que, em geral, pode ser configurado para realizar dois tipos de raciocínio computacional: o raciocínio baseado em ontologias e o raciocínio baseado em regras. Essa característica do serviço de inferência de contexto permite que este atenda à diretriz 2 de projeto da infra-estrutura SCK.

Maiores detalhes sobre os serviços de armazenamento, consulta e inferência de informações de contexto são fornecidos na seção a seguir.

5.3 Implementação dos serviços da infra-estrutura SCK

Como descrito na seção anterior, a infra-estrutura SCK oferece quatro serviços básicos para o desenvolvimento de aplicações sensíveis a contexto: descoberta, armazenamento, consulta e inferência. Antes de descrever a implementação de cada serviço, é importante ressaltar o ambiente de hardware e software utilizado para o desenvolvimento da infra-estrutura em questão.

A plataforma de hardware e software utilizada para o desenvolvimento da infra-estrutura SCK inclui CPU Intel^(R) Pentium^(R)4 2.66 GHz, 1 GiB RAM, sistema operacional Linux Red Hat 7.3, plataforma Java J2SE 1.5, ambiente de desenvolvimento Eclipse 3.0 e o arcabouço *Jena* 2.3 [Carroll et al., 2004]. Desenvolvido pela *HP Labs*, o arcabouço *Jena* para aplicações da Web Semântica oferece:

- mecanismos para manipulação de modelos de dados RDF em memória, bases de dados relacionais e arquivos;
- suporte às mais difundidas linguagens de consulta para modelos de dados RDF;
- um conjunto de APIs para manipulação de ontologias codificadas em esquema RDF ou OWL;
- e máquinas de inferência baseadas em ontologias e em regras, além de oferecer um mecanismo que permite integrar máquinas de inferência de terceiros.

Outros fatores que levaram o autor deste trabalho a optar pelo uso do arcabouço *Jena* incluem: o fato de ser desenvolvido na linguagem Java, o que permite atender à diretriz 5 de projeto da infra-estrutura *SCK*, a sua atualizada e extensa documentação, com tutoriais e diversos exemplos, e a sua alta adesão por parte de desenvolvedores de aplicações, que se reflete na quantidade de discussões conceituais e técnicas em sua lista de discussão. Bizer & Westphal [2006] fazem um levantamento das principais ferramentas para desenvolvimento de aplicações para Web Semântica levando em consideração suas principais características, o grau de esforço para seu desenvolvimento e o nível de adesão da comunidade de usuários.

As seções seguintes apresentam como foram implementados os serviços de armazenamento, consulta e inferência de contexto a partir do arcabouço *Jena*.

5.3.1 Serviço de Armazenamento de Contexto

Conforme apresentado na Seção 5.2, o *serviço de armazenamento de contexto* gerencia o armazenamento de informações de contexto em bases de dados relacionais ou em arquivos.

A abordagem baseada em bases de dados utiliza um esquema relacional especialmente adaptado para o armazenamento de dados no formato RDF [Guha, 2001; Wilkinson et al., 2003], daí ser comum utilizar o termo base de dados RDF. O mapeamento de bases relacionais com o modelo de dados RDF dá-se como segue: um registro é um recurso RDF, uma coluna na tabela é uma propriedade RDF e uma célula do registro é o valor da propriedade RDF. Nesta abordagem, informações de contexto são armazenadas em um base de dados RDF, enquanto que as ontologias que modelam essas informações e que compõem o modelo ontológico em questão são armazenadas cada qual em arquivos. Isto permite que as ontologias sejam lidas apenas quando necessário, por exemplo, pelo serviço de inferência de contexto.

A abordagem baseada em arquivos, por sua vez, é uma alternativa para aplicações sensíveis a contexto que não necessitam de funcionalidades de bases de dados relacionais, como consistência de dados ou transações. Nesta abordagem, informações

contextuais de cada fonte de contexto são armazenadas em seu próprio arquivo de contexto. Vale ressaltar que nesta abordagem as ontologias que compõem o modelo subjacente são também armazenadas em arquivos separados.

Considerando as abordagens utilizadas para armazenamento de informações de contexto, dados de configuração de bases de dados e de arquivos estão codificadas no formato RDF em um arquivo de configuração da infra-estrutura SCK. Esse arquivo é único para todas as aplicações que utilizam a infra-estrutura, portanto sempre que uma aplicação requisitar o armazenamento de informações de contexto, o serviço em questão checa os devidos parâmetros desse arquivo de configuração.

O Exemplo 5.1 descreve o formato desse arquivo de configuração na sintaxe RDF do tipo N-Triple [Grant & Beckett, 2004]. Neste caso, uma aplicação chamada *WebMemex* (APPNAME, linha 2) deseja armazenar informações de contexto em uma base de dados (STORAGE TYPE, linha 4) do tipo PostgreSQL (DBTYPE, linha 6), passando os seguintes parâmetros para conexão a base de dados (linhas 6 a 10): tipo de gerenciador de banco de dados a ser utilizado (DBTYPE), endereço de acesso (DBURL), usuário (DBUSER), senha (DBPWD) e *driver* JDBC (*Java Database Connectivity*), representado por DBDRIVER. Atualmente, a implementação do serviço de armazenamento de contexto acomoda MySQL e PostgreSQL.

```
0 <!--                               Exemplo 5.1                               -->
1 @prefix : <http://sck/config/app#>.
2 :appURI :appName "WebMemex".
3 :appURI :appStorage :appStorageURI.
4 :appStorageURI :storageType "DB".
5 :appStorageURI :storageParams :paramsURI.
6 :paramsURI :dbType "PostgreSQL".
7 :paramsURI :dbURL "jdbc:postgresql://localhost/sck".
8 :paramsURI :dbUser "postgres".
9 :paramsURI :dbPwd "wsxcvfr".
10 :paramsURI :dbDriver "org.postgresql.Driver".
```

Para o caso de configuração de armazenamento em arquivo, o único parâmetro passível de configuração é o nome do arquivo que servirá como repositório de informações de contexto. O Exemplo 5.2 ilustra o caso da mesma aplicação do Exemplo 5.1 indicando que suas informações de contexto devem ser armazenadas em um arquivo (STORAGE TYPE, linha 4) chamado *repository.rdf* (FILENAME, linha 5).

```
0 <!--                               Exemplo 5.2                               -->
1 @prefix : <http://sck/config/app#>.
2 :appURI :appName "WebMemex".
3 :appURI :appStorage :appStorageURI.
4 :appStorageURI :storageType "File".
5 :appStorageURI :fileName "repository.rdf".
```

Para que uma aplicação armazene informações de contexto, são necessários dois parâmetros de entrada para o serviço de armazenamento: o nome da aplicação,

para que seja verificada a configuração de armazenamento conforme apresentada nos Exemplos 5.1 e 5.22, e o grafo RDF que contém as informações de contexto a serem inseridas. Se o armazenamento for realizado em uma base de dados RDF, o serviço de armazenamento segue o algoritmo descrito no Exemplo 5.3.

```

0  <!--                                Exemplo 5.3                                -->
1  Cria um grafo vazio X em memória;
2  Adiciona ao grafo X as triplas do banco de dados da aplicação;
3  Se o grafo X é vazio então
4      Adiciona ao grafo X as triplas do grafo Y a ser armazenado;
5  Senão
6      Calcula a diferença entre o grafo X e o grafo Y a ser armazenado;
7      Se a diferença entre os grafos X e Y não for nula então
8          Adiciona ao grafo X as triplas que formam a diferença entre os grafos;
9      Senão; //o grafo Y já estava armazenado no banco de dados
10 Armazena as triplas do grafo X de volta ao banco de dados;

```

Por outro lado, se uma aplicação pretende explorar a abordagem de arquivos de contexto, cada nova informação contextual é inicialmente armazenada em memória. Esses grafos RDF, independentes entre si, servem temporariamente como a base de dados recente de uma aplicação. Dessa forma, eventuais consultas dessa aplicação a dados que constam em seus grafos são processadas de forma mais rápida.

Para evitar a perda de informações de contexto voláteis, os grafos em memória são periodicamente armazenados — configurável pelo desenvolvedor — cada qual em arquivos de registro de contexto (*context log files*).

Da mesma forma, para evitar que aplicações consultem informações de contexto inconsistentes, os arquivos de registro de contexto são periodicamente agrupados e armazenados em um único arquivo chamado arquivo de contexto, que contém todas as informações contextuais armazenadas por uma aplicação. Esse arquivo de contexto é o mesmo cujo nome pode ser configurado pelo desenvolvedor, conforme ilustrado no Exemplo 5.2 (FILENAME, linha 5).

Apesar da quantidade de memória necessária para armazenar grandes grafos RDF, a vantagem desta abordagem é a sua simplicidade e o bom desempenho para processamento de consultas, muito embora esse desempenho não tenha sido, até o momento, comprovado com experimentos.

Como apresentado nesta seção, o principal parâmetro de entrada para o serviço de armazenamento de contexto é o grafo RDF com as informações contextuais da aplicação em questão. Para construir esse grafo, independentemente se o armazenamento for em bases de dados ou em arquivos, é necessário adicionar a este recursos RDF que instanciam termos do vocabulário definido no modelo ontológico em questão. Em seguida, devem ser criados e associados a esses recursos as propriedades e seus respectivos valores, também segundo o vocabulário utilizado. Um trecho de código que ilustra a criação de um grafo RDF para armazenamento de

informações de contexto de uma aplicação é apresentado no Capítulo 7, Seção 7.2.4.

A maneira pela qual o serviço de armazenamento de contexto foi desenvolvido permite que qualquer modelo ontológico possa ser utilizado, bastando ao desenvolvedor criar classes Java a partir do vocabulário de cada ontologia do modelo em questão. Isso pode ser realizado por meio do programa distribuído com o arcabouço *Jena* chamado *schemagen*, cujas informações sobre sua utilização podem ser encontradas em Dollin [2006]. Após criadas as classes Java que representam as ontologias do modelo, estas devem ser copiadas para o pacote `BR.USP.ICMC.SCK.VOCABULARY`. Dessa maneira, o serviço de armazenamento de contexto atende à diretriz 3 de projeto da infra-estrutura SCK, independência de modelo ontológico.

5.3.2 Serviço de Consulta de Contexto

Como apresentado na Seção 5.2, o *serviço de consulta de contexto* atende a consultas de aplicações via linguagens declarativas para modelos RDF. A implementação atual do serviço de consulta permite que desenvolvedores utilizem as linguagens de consulta RDQL (*RDF Data Query Language*) [Miller et al., 2002] e SPARQL (*Simple Protocol and RDF Query Language*) [Prud'hommeaux & Seaborne, 2006].

No início do desenvolvimento da infra-estrutura SCK, o serviço de consulta utilizava apenas a linguagem RDQL, não apenas pelo fato dela ser a linguagem que o arcabouço *Jena* utilizava, mas principalmente porque ela é uma das mais utilizadas linguagens de consulta para dados RDF. Com os avanços rumo à padronização de uma linguagem de consulta pelo W3C, o autor optou por incluir o apoio a consultas expressas na linguagem SPARQL.

Tanto RDQL quanto SPARQL possuem notação semelhante à da linguagem SQL (*Structured Query Language*) [Chamberlin & Boyce, 1974], porém sua sintaxe declarativa é baseada no modelo de triplas RDF (*sujeito, predicado, objeto*). Para consulta de modelos RDF, essas linguagens utilizam o mecanismo de casamento de padrões de grafos: grafos que atendem ao padrão declarado na expressão de consulta são aqueles cujos valores são retornados à aplicação solicitante.

O Exemplo 5.4 apresenta uma típica expressão de consulta na linguagem SPARQL. Os valores dos subgrafos que atendem à consulta são armazenadas em variáveis precedidas de um ponto de interrogação, chamadas variáveis de consulta. A cláusula `SELECT` identifica o conjunto das variáveis de consulta (`?NAME` e `?AGE`, linha 4) a ser retornado para a aplicação solicitante. A cláusula `FROM` identifica a fonte de modelos RDF, no caso o arquivo *repository.rdf* na linha 5. Utilizando o serviço de consulta de contexto, é possível realizar consultas tanto sobre bases de dados RDF quanto sobre arquivos por meio das linguagens RDQL e SPARQL.

A cláusula `WHERE` especifica o padrão de grafo como uma conjunção de triplas. Por meio do mecanismo de casamento de padrões de grafos, são retornados à

aplicação solicitante os valores daqueles grafos armazenados que combinarem com o padrão declarado na cláusula WHERE. A cláusula FILTER permite definir restrições sobre variáveis de consulta SPARQL, por exemplo, ao testar o valor de uma variável (?AGE > 30, linha 8). Por fim, a cláusula PREFIX é um mecanismo para associar sinônimos a uma URI de vocabulário (VCARD, linha 3), pois URIs extensas podem dificultar o entendimento da consulta caso estivessem na cláusula WHERE.

```

0  <!--                                Exemplo 5.4                                -->
1  PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2          usr: <http://coweb.icmc.usp.br/UserGroup-ns#>
3          vcard: <http://www.w3.org/2001/vcard-rdf/3.0>
4  SELECT ?name ?age
5  FROM    <http://example.org/repository.rdf>
6  WHERE   { ?x rdf:type usr:User .
7            ?x usr:age  ?age      .
8            FILTER (?age > 30)    .
9            ?x vcard:FN ?name     . }
```

Baseado na breve explanação anterior, o Exemplo 5.4 descreve a situação em que uma aplicação consultar o nome completo (VCARD:FN, linha 9) de todos os usuários (USR:USER, linha 6) maiores de 30 anos (linha 8).

Para que uma aplicação consulte informações de contexto, são necessários três parâmetros de entrada para o serviço de consulta: o nome da aplicação, para que seja verificada a configuração e o tipo de repositório de informações de contexto, uma expressão de consulta declarada segundo a sintaxe da linguagem de consulta escolhida, e a lista de variáveis que armazena as informações recuperadas. Independentemente do tipo de repositório e da linguagem de consulta utilizados pela aplicação, o serviço de consulta segue o algoritmo descrito no Exemplo 5.5.

```

0  <!--                                Exemplo 5.5                                -->
1  Cria um grafo vazio X em memória;
2  Adiciona ao grafo X as triplas do repositório da aplicação;
3  Cria uma lista Y para armazenar os valores das variáveis de consulta;
4  Se o grafo X não está vazio então
5      Associa a expressão de consulta e o grafo X ao processador de consultas;
6      Executa o processador de consultas;
7      Para cada subgrafo encontrado como solução E cada variável de consulta
8          Obtém o valor correspondente à variável de consulta;
9          Converte o valor da variável para o tipo RDF correspondente (Recurso ou Literal);
10         Adiciona o valor convertido à lista Y;
11 Retorna a lista Y à aplicação solicitante;
```

A partir do Exemplo 5.5, deseja-se mostrar a desenvolvedores que, para utilizar o serviço de consulta de contexto, é necessário apenas escolher a linguagem que atende às necessidades de consulta de sua aplicação, bem como escrever as expressões de consulta correspondentes. Para optar por uma linguagem específica de consulta, o desenvolvedor deve considerar a riqueza das expressões de consulta que sua aplicação

requer, em geral, maior no caso da linguagem SPARQL. A linguagem RDQL, por exemplo, não oferece suporte à tipagem de literais (ex: datas, horas, números inteiros, etc.), união de grafos de consulta, múltiplas fontes de dados RDF, operadores para ordenar e delimitar soluções de consultas, entre outros. Todas essas características são oferecidas pela linguagem SPARQL.

Segundo a implementação do serviço de consulta de contexto, cujo algoritmo foi ilustrado no Exemplo 5.5, este serviço é independente do modelo ontológico utilizado por desenvolvedores. Assumindo que os desenvolvedores tenham gerado as classes Java das ontologias do modelo, conforme explicado na seção anterior, cabe-lhes referenciar os termos do vocabulário correspondente nas expressões de consulta a serem enviadas ao serviço em questão. Como as expressões de consulta são independentes do serviço de consulta, este atende à diretriz 3 de projeto da infra-estrutura SCK.

5.3.3 Serviço de Inferência de Contexto

Como descrito na Seção 5.2, o *serviço de inferência de contexto* fornece a aplicações (ou consumidores de contexto) um mecanismo configurável de inferência sobre informações de contexto baseado em ontologias e em regras. Embora este serviço ofereça atualmente duas possibilidades de inferência, inicialmente o serviço foi projetado para oferecer apenas inferência baseada em ontologias.

Inferência baseada em ontologias

Em um processo de inferência baseado em ontologias, inferem-se informações de contexto a partir da combinação da semântica definida pelos construtores da linguagem em que uma ontologia é construída e de um conjunto de fatos instanciados desta ontologia. Assim, pode-se inferir informações de contexto com base em relações de generalização, especialização, transitividade, simetria, inversão, etc.

A versão atual do serviço de inferência de contexto pode ser configurada para realizar inferência sobre ontologias codificadas nas linguagens esquema RDF e OWL. Três tipos de máquinas de inferência têm sido integrados ao serviço de inferência de contexto, a saber:

- a máquina de inferência transitiva (*TransitiveReasoner*), que infere relações de especialização e generalização entre classes e propriedades por meio dos construtores da linguagem esquema RDF para definição de subclasses (`rdfs:subClassOf`) e subpropriedades (`rdfs:subPropertyOf`);
- a máquina de inferência RDFS (*RDFSReasoner*), que implementa um subconjunto configurável dos construtores da linguagem esquema RDF, tais como

subclasses, subpropriedades e domínio (`rdfs:domain`) e imagem (`rdfs:range`) de propriedades;

- máquinas de inferência OWL (*OWLReasoner*), que implementam todas as deduções obtidas de ontologias baseadas em esquema RDF, bem como diferentes subconjuntos da linguagem OWL. No caso de ontologias OWL, como aquelas que compõem o modelo *SeCoM*, diferentes graus de expressividade lógica podem ser aceitos. Desta forma, a configuração apropriada de uma máquina de inferência OWL depende dos construtores OWL envolvidos no processo de inferência. Além das máquinas de inferência OWL oferecidas pelo arcabouço *Jena*, inclui-se a máquina de inferência *Pellet*, ideal para processar ontologias que exploram a máxima expressividade dos construtores fornecidos pela linguagem OWL.

Para que uma aplicação sensível a contexto se beneficie de processos de inferência baseados em ontologias, o serviço de inferência de contexto deve ser invocado com quatro parâmetros de entrada: o tipo de máquina de inferência (*TransitiveReasoner*, *RDFSReasoner* ou *OWLReasoner*), uma lista de parâmetros de configuração da máquina de inferência (caso necessário)¹, o conjunto de ontologias cuja semântica será utilizada para direcionar o processo de inferência e um conjunto de fatos instanciados a partir desse conjunto de ontologias. Estes dois últimos parâmetros, na verdade, são grafos com descrições RDF, e são comumente chamados *TBox* e *ABox*, respectivamente, conforme apresentado no Capítulo 3, Seção 3.3.4. Para um processo de inferência baseado em ontologia, o serviço de inferência de contexto segue o algoritmo descrito no Exemplo 5.6.

```

0  <!--                                Exemplo 5.6                                -->
1  Se o grafo TBox não for vazio
2      Identifica a máquina de inferência a utilizar;
3      Configura a máquina de inferência com parâmetros de configuração;
4      Associa os grafos TBox e ABox à máquina de inferência;
5      Executa a máquina de inferência;
6      Cria um grafo X vazio;
7      Adiciona ao grafo X o resultado da execução da máquina de inferência;
8      Retorna o grafo X à aplicação solicitante;
```

Vale ressaltar que o resultado retornado à aplicação solicitante é um grafo RDF (linhas 6, 7 e 8) contendo não apenas as informações do grafo *ABox*, mas principalmente as novas informações de contexto deduzidas. O conjunto de informações de contexto deduzidas pode ser obtido separadamente das informações de origem por meio de métodos que a API do arcabouço *Jena* oferece.

A partir do Exemplo 5.6, consegue-se ilustrar que todos os tipos de máquinas de inferência supracitados derivam novos fatos RDF a partir de uma base de informações

¹As possibilidades de configuração da máquina de inferência *OWLReasoner* são discutidas na seção seguinte.

de contexto (*Abox*) associada a um (ou mais) arquivo(s) de ontologia(s) (*TBox*). Uma base de conhecimento de contexto é, portanto, a combinação de informações de contexto instanciadas a partir da estrutura e semântica das ontologias com as quais essas instâncias são compatíveis.

Dado que é natural que surjam indagações a respeito de como cada máquina de inferência processa um conjunto de informações de contexto, o Exemplo 5.7 ilustra um trecho de uma base de informações de contexto que está associada à ontologia Activity do modelo *SeCoM* (@actvy, linha 3). A atividade descrita em questão consiste de uma reunião agendada (linha 4) da comissão de pós-graduação (linha 5).

```

0  <!--                               Exemplo 5.7                               -->
1  @prefix : <http://sck/example#>.
2  @rdf      : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
3  @actvy    : <http://linkserver.icmc.usp.br/ckonto/activity#>.
4  :activityURI rdf:type actvy:ScheduledActivity.
5  :activityURI actvy:hasSummary "Reunião da Comissão da Pós-Graduação".

```

Se o serviço de inferência de contexto for configurado para utilizar a máquina de inferência *TransitiveReasoner*, esta é capaz de inferir que o recurso identificado pela URI *activityURI* (linha 4) é um indivíduo das classes Activity, SpatioTemporalEvent e SpatioTemporalThing. Isto se deve ao fato do desse recurso ser uma instância declarada da classe ScheduledActivity (linha 4), que é subclasse de Activity, subclasse de SpatioTemporalEvent que, por sua vez, é subclasse de SpatioTemporalThing. *TransitiveReasoner* não consegue fazer, entretanto, qualquer tipo de inferência sobre a propriedade ACTVY:HASSUMMARY (linha 5).

Se for utilizada a máquina *RDFSReasoner*, esta também infere que o recurso *activityURI* é indivíduo das classes Activity, SpatioTemporalEvent e SpatioTemporalThing pela mesma razão já apresentada. A associação desse recurso à classe Activity é reforçada pelo fato de todo indivíduo dessa classe apresentar a propriedade ACTVY:HASSUMMARY. Portanto, se o recurso em questão não tivesse seu tipo declarado (linha 4), mesmo assim seria inferido que o recurso é indivíduo da classe Activity. Entretanto, essa máquina de inferência não teria como inferir se a classe de atividades a qual o recurso pertence é a de atividades agendadas (linha 4).

Quando uma máquina de inferência OWL é utilizada para processar as informações de contexto do Exemplo 5.7, ela consegue inferir além do que as outras já citadas inferem. Algumas dessas novas informações de contexto inferidas e as respectivas justificativas são apresentadas a seguir:

- o recurso *activityURI* possui uma propriedade actvy:schedulingStatus cujo valor booleano é verdadeiro, pois todo indivíduo da classe de atividades agendadas (ScheduledActivity) atende a essa restrição;

- o recurso *activityURI* não pode ser indivíduo da classe de atividades que acontecem espontaneamente (*ScheduledActivity*), pois atividades agendadas e espontâneas são declaradas como disjuntas.

Vale observar que máquinas de inferência OWL conseguem inferir informações de contexto adicionais em relação às outras dada a semântica da linguagem de ontologia OWL, superior à da linguagem esquema RDF, por exemplo. Mais detalhes a respeito da utilização de máquinas de inferência podem ser encontradas na seção seguinte e no Capítulo 7, o qual descreve um estudo de caso em que o serviço de inferência de contexto é utilizado por uma aplicação.

Inferência baseada em regras

O serviço de inferência de contexto foi estendido posteriormente com a capacidade de apoiar processos de inferência baseados em regras [Bulcão Neto et al., 2005b]. Regras devem ser expressas por desenvolvedores na forma de conjunções de predicados dispostos em premissas e uma conclusão. Para isso, é utilizada a sintaxe de construção de regras² adotada pelo subsistema de inferência do arcabouço Jena [Reynolds, 2006]. Essa sintaxe incorpora construtores específicos que podem ser utilizados como predicados, por exemplo, para manipulação de listas. Demais predicados correspondem a termos do vocabulário definido em ontologias do modelo em questão, por exemplo, termos das ontologias do modelo *SeCoM*.

Para uma aplicação sensível a contexto utilizar o mecanismo de inferência baseado em regras, o serviço de inferência de contexto deve ser invocado com dois parâmetros de entrada: uma lista de parâmetros de configuração da máquina de inferência e um grafo RDF contendo um conjunto de fatos instanciados a partir de algum modelo ontológico (*ABox*). Para um processo de inferência baseado em regras, o serviço de inferência de contexto segue o algoritmo descrito no Exemplo 5.8.

```

0  <!--                                Exemplo 5.8                                -->
1  Se o grafo ABox não for vazio
2    Configura a máquina de inferência com parâmetros de configuração;
3    Associa o grafo ABox à máquina de inferência;
4    Executa a máquina de inferência segundo o encadeamento escolhido;
5    Cria um grafo X vazio;
6    Adiciona ao grafo X o resultado da execução da máquina de inferência;
7    Retorna o grafo X à aplicação solicitante;
```

Vale observar que a diferença entre os algoritmos do serviço de inferência de contexto dos Exemplos 5.6 e 5.8 está na ausência de grafo(s) de ontologia(s) (*TBox*). Ou seja, em processos de inferência baseados em regras, a semântica dos termos

²A versão atual do serviço de inferência de contexto não permite que desenvolvedores escrevam regras por meio da linguagem de especificação de regras SWRL, apresentada no Capítulo 3, Seção 3.3.5.

que representam os predicados da regra não é considerada, nem a semântica dos construtores da linguagem de ontologia utilizada para modelar esses termos.

É importante destacar que a máquina de inferência para regras é diferente daquelas utilizadas para inferência baseada em ontologias. Seus parâmetros de configuração incluem a descrição das regras em questão, a opção pelo armazenamento temporário de processos de inferência recentes e o tipo de encadeamento de regras (linhas 2 a 4 do Exemplo 5.8). Este último parâmetro determina três maneiras que um desenvolvedor pode escolher para processar regras: encadeamento progressivo (*forward*), retrógrado (*backward*) ou híbrido (*forward-backward*).

No encadeamento progressivo a máquina de inferência investiga se uma conjunção de predicados declarada como premissa da regra é encontrada em um repositório de informações de contexto, no caso do Exemplo 5.8, o grafo *ABox*. Caso seja verdadeiro, consegue-se inferir uma nova informação de contexto, que está declarada como conclusão da regra.

O Exemplo 5.9 exibe uma regra simples que indica as condições necessárias para inferir que tio é o grau de parentesco entre dois recursos. Neste caso, se a base de informações de contexto em questão contiver triplas RDF que casam com as triplas das premissas dessa regra, então infere-se a relação de parentesco tio entre os referidos recursos, estes provavelmente do tipo pessoa.

```
0 <!--                               Exemplo 5.9                               -->
1 @prefix xyz: <http://example/family#>.
2 [r1: (?a xyz:father ?b) (?c xyz:brother ?b) -> (?c xyz:uncle ?a)]
```

No encadeamento retrógrado o processamento é inverso: se o predicado declarado na conclusão é encontrado, então a máquina de inferência adiciona ao grafo *ABox* a conjunção de predicados declarados nas premissas. Considerando a regra do Exemplo 5.9 em sua forma invertida, caso seja encontrada em uma base de informações de contexto a tripla RDF que descreve que um recurso C é tio de um recurso A, então a máquina de inferência adiciona a essa base que há relações de pai entre A e B, bem como de irmão com respeito aos recursos C e B.

Ao utilizarem o encadeamento de regras híbrido, desenvolvedores podem declarar regras como predicados de outras regras, tal como (A , B -> (C <- D)). Neste caso, regras de encadeamento progressivo são executadas primeiro.

Processos de inferência baseados em regras consideram, portanto, apenas se a partir de uma conjunção de premissas pode-se chegar a uma conclusão ou vice-versa. A partir desse tipo de processo de inferência, consegue-se atender a aplicações que necessitam declarar explicitamente as situações sobre as quais o processo de inferência é relevante. A relevância em questão é, como apresentado, prerrogativa do desenvolvedor de uma aplicação.

5.4 Avaliação do serviço de inferência de contexto

Ao discutir sobre as ontologias do modelo *SeCoM* como formalismo de representação de informações de contexto, bem como sobre a infra-estrutura *SCK* para processar a semântica desse modelo, o autor fora questionado quanto aos custos de desempenho associados à utilização de mecanismos de inferência em aplicações sensíveis a contexto. Na busca por uma resposta, o autor investigou como seria possível comparar e quantificar o desempenho de uma aplicação sensível a contexto que utilizasse inferência baseada em ontologias.

A métrica mais utilizada na literatura quanto à avaliação de desempenho de processos de inferência baseados em ontologias é o tempo total de resposta desse processo [Tempich & Volz, 2003]. No entanto, a literatura também considera um conjunto de tempos associados a cada sub-processo do processo de inferência, tais como os tempos de classificação e de carregamento dos termos de uma ontologia [Baader et al., 2003]. Esses tempos intermediários são discutidos na Seção 5.4.2.

Esta seção apresenta como o serviço de inferência de contexto foi avaliado utilizando as ontologias do modelo *SeCoM* como base de testes. Dessa forma, foi possível também avaliar o efeito no desempenho de aplicações que utilizem as ontologias do modelo *SeCoM* [Bulcão Neto & Pimentel, 2006a].

5.4.1 Configuração do experimento

O ambiente de hardware e software utilizado para a avaliação em questão inclui CPU Intel^(R) Pentium^(R)4 2.66 GHz, 1 GiB RAM, sistema operacional Linux Red Hat 7.3, plataforma Java J2SE 1.5, Eclipse 3.0, o subsistema de inferência do arcabouço *Jena* 2.3 e a máquina de inferência *Pellet* 1.3beta2.

A máquina de inferência *Pellet* foi escolhida em virtude desta oferecer a desenvolvedores uma API por meio da qual é possível medir não apenas o tempo total de inferência sobre ontologias, mas também os tempos de sub-processos do processo de inferência como um todo.

O autor também ajustou para 64MiB o tamanho da memória alocada à máquina virtual Java para evitar a falta de memória durante o processo de inferência sobre as ontologias do modelo *SeCoM*.

5.4.2 Ontologias *SeCoM* como dados de teste

O autor realizou o experimento com todas as ontologias do modelo *SeCoM*, inclusive as ontologias de apoio descritas no Apêndice A. Nesta seção, será descrito, entretanto, apenas o experimento com as ontologias principais do modelo: *Actor*, *Spatial*, *Spatial Event*, *Time*, *Temporal Event*, *Device* e *Activity*. O mesmo experimento

com as ontologias de apoio está descrito no Apêndice A.

O experimento em questão consiste em configurar o serviço de inferência de contexto para utilizar a máquina de inferência *Pellet* e o arquivo de descrição de cada ontologia do modelo *SeCoM*. Depois que o serviço de inferência de contexto é configurado, este é executado 10 vezes consecutivas, sendo cada execução precedida de uma limpeza da memória. Essa abordagem utilizada por Pan [2005] visa a eliminar quaisquer interferências nas medidas coletadas do experimento, dado que mecanismos de cache são utilizados pela máquina de inferência *Pellet* para acelerar o processo de inferência.

Para cada ontologia do modelo *SeCoM* foi criado um arquivo textual contendo informações geradas pelo processamento da ontologia pela máquina de inferência *Pellet*. Essas informações representam não apenas o tempo total da inferência sobre a ontologia em questão, mas também os tempos dos sub-processos de inferência. Uma vez coletados esses tempos, foram calculados os respectivos tempos médio e máximo e o desvio padrão. A Tabela 5.1 apresenta o tempo médio (em milissegundos) de cada um dos tempos intermediários de um processo de inferência baseado em ontologias. A descrição de cada um desses tempos intermediários é dada a seguir.

Ontologia	TCA	TCM	TVD	TVC	TC	TF	TAI
Time	1741.5	103	439.5	32.2	54.3	0	6.5
Temporal Event	1774.1	98	474	25	54	2	6.6
Spatial	1638	65.8	271.7	83.7	133	0	6.1
Spatial Event	1695.4	76.2	305	131.8	197.1	22	3.4
Actor	1650.6	48.5	122.8	12.6	35.8	0	0
Device	1998.7	290.6	855.8	58	107	0	18.1
Activity	2536.1	354.6	1134.7	288.3	491.7	46.3	17.8

Tabela 5.1: Tempo médio de sub-processos de inferência sobre ontologias do modelo *SeCoM* (em ms). Adaptado de Bulcão Neto & Pimentel [2006a].

Tempo de carregamento do arquivo da ontologia (TCA): tempo que a máquina de inferência leva para ler, analisar e carregar o arquivo de descrição de uma ontologia em memória. Isto ocorre, portanto, antes de o processo de inferência dar início. Este tempo tem um grande impacto em processos de inferência, dado que pode apresentar uma grande variabilidade, tanto para carregamento de arquivos de descrição de ontologias locais quanto de arquivos remotos. Considerando este experimento, o tempo de carregamento dos arquivos de ontologias *SeCoM* gasta, em média, 75,3% de um processo de inferência completo.

Tempo de carregamento de modelos RDF da ontologia (TCM): tempo que a máquina de inferência consome para carregar em sua memória interna todos os modelos RDF da descrição do arquivo de uma ontologia. Quanto maior o

número de triplas de uma ontologia, maior será esta medida, como é o caso das ontologias *Activity* e *Device*. No caso das ontologias descritas na Tabela 5.1, o tempo de carregamento de modelos RDF consome 16,2%, em média.

Tempo de validação de dialeto de ontologia (TVD): tempo que uma máquina de inferência consome para validar o dialeto de uma ontologia OWL, por exemplo, *OWL Lite* ou *OWL DL*. Tal qual o tempo de carregamento de modelos RDF da ontologia (TCM), esse tempo também aumenta conforme a quantidade de triplas RDF contidas nos arquivos de descrição de ontologias, como no caso das ontologias *Activity* e *Device*. Se desenvolvedores souberem previamente o dialeto e a expressividade DL de ontologias a serem utilizadas por uma aplicação, o autor recomenda que esse serviço de validação de dialeto de ontologias seja desabilitado. No caso do experimento aqui reportado, consegue-se economizar, em média, até 57,6% do tempo gasto para inferir sobre as ontologias do modelo *SeCoM*, caso esse serviço esteja desabilitado.

Tempo de cálculo de consistência de ontologia (TVC): quantidade de tempo gasta por uma máquina de inferência para determinar se a definição de uma ontologia é consistente, ou seja, se esta não contém qualquer fato que seja contraditório na sua definição. O autor percebeu que os valores de TVC são maiores quando a expressividade lógica de uma ontologia é maior. Por exemplo, as ontologias *Activity*, *Spatial Event* e *Spatial* possuem a maior expressividade lógica — SHOIN(D) — dentre as ontologias do modelo *SeCoM*, daí seus respectivos tempos médios de cálculo de consistência serem, em alguns casos, várias vezes maior que os tempos de ontologias com expressividade próxima, tal como a ontologia *Device*. Embora seja um dos principais sub-processos relacionados ao processo de inferência como um todo, o cálculo de consistência consome, em média, 9,2% do processo de inferência com as ontologias do modelo *SeCoM*.

Tempo de classificação de ontologia (TC): tempo que uma máquina de inferência consome para computar a hierarquia de classes de uma ontologia, que pode ser utilizada, por exemplo, para responder a consultas sobre todas as subclasses ou apenas as subclasses diretas de uma dada classe. A máquina de inferência *Pellet* permite otimizar o tempo de classificação de uma ontologia ao desabilitar o processamento de construtores nominais (vide Capítulo 4, Tabela 4.3). Dado que existem ontologias do modelo *SeCoM* que utilizam esse tipo de construtor, tais como aquelas que têm expressividade SHOIF(D) ou SHOIN(D), o autor utilizou a configuração padrão da máquina de inferência *Pellet*.

A Tabela 5.1 descreve que, no experimento em questão, quanto mais complexa a expressividade lógica de uma ontologia, maior é o tempo de classificação da

mesma, como pode ser evidenciado por meio das ontologias *Activity*, *Spatial Event* e *Spatial ontologies*. Entretanto, este resultado não pode ser generalizado uma vez que o tempo de classificação pode ser minimizado dado a otimizações existentes de outros sub-processos, tais como o processo de cálculo de consistência [Baader et al., 2003]. O tempo de classificação representa uma média de 16,2% do tempo total de inferência sobre as ontologias do modelo *SeCoM*.

Tempo de fusão de ontologia (TF): tempo que uma máquina de inferência consome para unir ontologias que importam outras. Como ilustrado na Figura 4.1, Capítulo 4, ontologias do modelo *SeCoM* importam outras ontologias, como a ontologia *Activity*, que importa as ontologias *Actor*, *Device*, *Temporal Event* e *Spatial Event*. É importante observar que quanto maior o número de triplas nas ontologias importadas, maior será o tempo de fusão, tal qual mostra o tempo medido para a ontologia *Activity*. Com base neste experimento, o tempo de fusão consome 10,7 ms, em média, para as ontologias do modelo *SeCoM*, o que representa apenas 1% de todo o processo de inferência sobre essas ontologias. A maneira como o modelo *SeCoM* foi modelado no sentido de poder ser estendido ou instanciado por aplicações com seu próprio vocabulário aliada às medidas dos tempos de fusão coletados corroboram a viabilidade da abordagem de modelagem em duas camadas do modelo *SeCoM*.

Tempo de associação de instâncias a classes de ontologia (TAI): tempo que uma máquina de inferência leva para calcular as classes mais específicas das quais um indivíduo é uma instância. Este sub-processo é realizado após o sub-processo de classificação de ontologias (TC), dado que as instâncias são associadas à hierarquia de classes da ontologia em questão. A Tabela 5.1 descreve que quanto mais indivíduos declarados em uma ontologia, maior será o tempo de associação de instâncias a classes. Como as ontologias do modelo *SeCoM* têm um pequeno número de indivíduos, seus respectivos tempos TAI representam aproximadamente as mesmas medidas em milissegundos. Os tempos TAI representam uma média de 1% do tempo total de inferência sobre as ontologias do modelo *SeCoM*.

5.5 Considerações finais

Este capítulo apresentou a infra-estrutura SCK que oferece a aplicações sensíveis a contexto um conjunto de serviços para o armazenamento, consulta e inferência de informações de contexto. Foi mostrado como esses serviços utilizam um modelo ontológico de informações de contexto, tal como o modelo *SeCoM*, embora tais serviços sejam independentes de modelo ontológico.

É importante destacar que os serviços de armazenamento, consulta e inferência da infra-estrutura SCK foram projetados e implementados com alta coesão (funcional) e baixo acoplamento (por dados) entre si, características desejáveis de software de qualquer natureza [Sommerville, 2006].

Com base na avaliação do serviço de inferência de contexto, foram identificadas questões de projeto que desenvolvedores de aplicações sensíveis a contexto devem considerar quando utilizam modelos ontológicos de informação contextual.

Primeiro, utilizando as ontologias do modelo *SeCoM* como dados de teste, foi apresentado como o tempo total de um processo de inferência baseado em ontologias é subdividido. Dada a influência de cada um desses tempos no processo de inferência em geral, esta é uma importante questão a ser considerada por desenvolvedores de aplicações sensíveis a contexto baseadas em ontologias.

Segundo, dada a quantidade de tempo dispensada por processos de inferência baseados em ontologias, a avaliação realizada mostra a viabilidade de um módulo independente para inferência sobre informações de contexto, tal como o serviço de inferência da infra-estrutura SCK.

Terceiro, embora o tempo de carregamento do arquivo da ontologia (TCA) consuma, em média, quase 80% do tempo total de inferência sobre ontologias *SeCoM*, desenvolvedores podem reduzir esse tempo ao introduzir em um serviço de inferência um mecanismo de armazenamento em memória das ontologias freqüentemente utilizadas.

Um Processo para Aplicações Sensíveis a Contexto

A característica de sensibilidade a contexto de aplicações de computação ubíqua tem demandado, segundo Abowd [1999], a investigação por técnicas de Engenharia de Software que apoiem a tarefa de desenvolvimento de aplicações sensíveis a contexto. As observações de Abowd incluem a demanda por projetos de arquiteturas baseadas na separação de interesses entre aplicações e infra-estruturas de software. De maneira similar, Helal [2005] e Kortuem et al. [2006] apontam a necessidade por técnicas de Engenharia de Software que tratem a complexidade e a demanda de tempo envolvida no desenvolvimento de aplicações dessa natureza.

Abordagens genéricas de modelagem de informação contextual têm sido demandadas no sentido de capturar várias características de informações de contexto, como sua diversidade de tipos e históricos de informação contextual. Nesse cenário, ontologias têm assumido um papel importante como técnica de modelagem de informação contextual devido às suas características de formalidade, expressividade, portabilidade, abstração de implementação e, ainda, por habilitar aplicações a inferirem sobre informações de contexto.

Modelos de informação contextual baseados em ontologias têm sido bastante explorados para permitir a cooperação transparente entre dispositivos heterogêneos, como em Chen et al. [2004d], assim como para descoberta, aquisição, inferência e distribuição de informação contextual, como nos trabalhos de Chen et al. [2004b], Gu et al. [2005] e Tan et al. [2005]. Entretanto, soluções de Engenharia de Software para

apoiar a construção de aplicações sensíveis a contexto ainda merecem investigação.

Este capítulo apresenta um novo processo de software para apoiar o desenvolvimento de aplicações sensíveis a contexto [Bulcão Neto et al., 2006b], especialmente aquelas cujas informações de contexto são baseadas em ontologias. O processo *POCAp* (*Process for Ontological Context-aware Applications*) assume que tais aplicações são construídas com base em uma *infra-estrutura de serviços* capaz de interpretar a semântica de informações de contexto, semântica essa fornecida por um *modelo ontológico de informação contextual*.

O capítulo descreve o processo *POCAp* sob duas perspectivas: primeiro, o processo é descrito, em linhas gerais, como um conjunto coerente de atividades (ou fases); segundo, o capítulo fornece detalhes sobre como essas atividades devem ser desempenhadas e as implicações de cada atividade que compõe o processo *POCAp* no ciclo de vida de aplicações desde a fase de análise e especificação até a fase de verificação e validação. Por fim, o autor faz suas considerações finais quanto à definição e utilização do processo *POCAp*.

6.1 Uma visão geral do processo *POCAp*

A literatura de Engenharia de Software é unânime quanto à definição de processo de software como um conjunto de atividades, que incluem, segundo Fuggetta [2000] e Sommerville [2006], análise e especificação, projeto, desenvolvimento, verificação e validação, implantação, operação, manutenção e aposentadoria.

O processo *POCAp* abrange as quatro primeiras fases: análise e especificação, projeto, desenvolvimento e verificação e validação. Este trabalho não considera as quatro fases restantes dada a dependência destas com respeito à instituição onde a aplicação deverá ser implantada.

Em geral, cada atividade *POCAp* tem os mesmos objetivos de uma fase de processo de software tradicional:¹

- (a1) Análise e especificação** é a atividade de estabelecer o que a aplicação sensível a contexto deve fazer (requisitos funcionais), as restrições na operação e desenvolvimento da aplicação (requisitos não-funcionais), e o conjunto de informações de contexto necessário;
- (a2) Projeto** é a atividade de produzir uma estrutura de software — por exemplo, um projeto arquitetural — que satisfaz a especificação da aplicação sensível a contexto;
- (a3) Desenvolvimento** é a atividade de traduzir a estrutura de software produzida na atividade de projeto para um programa executável;

¹Este trabalho utiliza as definições de Sommerville [2006] para cada atividade do processo *POCAp*.

Figura 6.1: O processo de software *POCAp*. Adaptado de Bulcão Neto et al. [2006b].

(a4) Verificação e validação corresponde às atividades de checar, revisar e testar se a aplicação sensível a contexto está em conformidade com sua especificação e se atende aos requisitos estabelecidos.

Para modelar o processo *POCAp*, este trabalho utilizou a especificação *SPEM* (*Software Process Engineering Metamodel*) [OMG, 2006], que fornece um arcabouço formal baseado em UML que permite modelar processos de desenvolvimento de software.

A Figura 6.1 apresenta o processo *POCAp* com foco em suas quatro principais atividades:² (a1) análise e especificação, (a2) projeto, (a3) desenvolvimento e (a4) verificação e validação.

Como explicitado na Figura 6.1, o processo *POCAp* considera quatro diferentes papéis de atores durante o desenvolvimento de uma aplicação: (u1) analistas, (u2) projetistas, (u3) desenvolvedores e (u4) validadores. Analistas são conduzidos ao longo da (a1) atividade de análise e especificação para produzirem artefatos que descrevam as demandas da aplicação e de seus usuários. Projetistas utilizam os artefatos de requisitos para planejar a (a3) atividade de desenvolvimento da aplicação. Desenvolvedores utilizam os artefatos de projeto para entender como a aplicação deve ser desenvolvida. Finalmente, validadores ou engenheiros de teste utilizam o documento de requisitos e a implementação da aplicação para realizar testes de validação de uma aplicação.

A notação da especificação *SPEM* permite representar as atividades, conexões e possíveis iterações entre as atividades internas do processo *POCAp*. As flechas na

²Este trabalho utiliza rótulos numerados para cada atividade, tanto nas figuras apresentadas neste capítulo quanto no texto para facilitar a apresentação do processo.

Figura 6.1 que representam iterações após cada atividade (exceto a (a1) atividade de análise e especificação) indicam que é possível mudar aspectos de uma aplicação sensível a contexto em resposta às mudanças de demanda.

A fim de simplificar a visão de mais alto nível do processo *POCAp*, este trabalho omitiu os artefatos de entrada e saída gerados a partir de cada atividade na Figura 6.1. Esses artefatos são representados nas próximas seções.

6.2 Atividade de análise e especificação (a1)

A Figura 6.2 descreve a (a1) atividade de análise e especificação do processo *POCAp*, que é composta de quatro sub-atividades: (a1.1) análise e especificação de requisitos, (a1.2) análise e especificação de informação contextual, (a1.3) análise e especificação de reuso do modelo e (a1.4) análise e especificação de extensão do modelo. A Figura 6.2 também ilustra que documentos são produzidos como *saída* de algumas atividades e que documentos são demandados como *entrada* para atividades.

Um elemento importante na (a1) atividade de análise e especificação é o (g1) modelo ontológico de informações de contexto, utilizado como entrada para duas sub-atividades: (a1.3) análise e especificação de reuso do modelo e (a1.4) análise e especificação de extensão do modelo. Um (g1) modelo ontológico de informações de contexto de alto nível, produzido por um engenheiro de ontologias, deve fornecer um vocabulário de termos a ser utilizado por aplicações. Nas sub-atividades componentes da (a1) atividade de análise e especificação são produzidos os seguintes artefatos, respectivamente: (d1) documento de requisitos, (d2) documento de informações de contexto, (d3) documento de reuso do modelo e, por fim, (d4) documento de modelo ontológico da aplicação.

As quatro sub-atividades da (a1) atividade de análise e especificação são detalhadas a seguir.

6.2.1 Análise e especificação de requisitos (a1.1)

A sub-atividade de análise e especificação de requisitos é aquela na qual o (u1) analista deve coletar conceitos relativos aos requisitos da aplicação e dos usuários dessa aplicação na forma de requisitos funcionais e não-funcionais. Como *saída*, é gerado um (d1) documento de requisitos, cuja estrutura pode seguir algum tipo de padrão tal como o IEEE/ANSI 830-1998 [ANSI/IEEE, 1998]. Esse (d1) documento de requisitos é o artefato de *entrada* para a sub-atividade (a1.2) de análise e especificação de informação contextual e, ao mesmo tempo, o artefato de *saída* para a (a2) atividade de projeto.

Figura 6.2: A atividade de análise e especificação do processo *POCAp*. Adaptado de Bulcão Neto et al. [2006b].

6.2.2 Análise e especificação de informação contextual (a1.2)

Na sub-atividade de análise e especificação de informação contextual, o (u1) analista deve identificar que informações são relevantes para cada situação de uso da aplicação. Tal relevância caracteriza uma informação de contexto, que pode ser obtida a partir dos requisitos funcionais especificados no (d1) documento de requisitos. O artefato de *saída* desta sub-atividade é o (d2) documento de informações de contexto.

6.2.3 Análise e especificação de reuso do modelo (a1.3)

A sub-atividade de análise e especificação de reuso do modelo visa a delimitar o reuso de um (g1) modelo ontológico de informações de contexto existente, que é utilizado como artefato auxiliar na modelagem de informações de contexto de uma aplicação. Além do (g1) modelo ontológico de informações de contexto, o (d2) documento de informações de contexto também é um artefato de *entrada* para esta sub-atividade.

Inicialmente, o (u1) analista deve mapear o (d2) documento de informações de contexto para a terminologia empregada em ontologias, tais como conceitos, relações e axiomas. Por isso, é recomendado que o analista seja auxiliado por um engenheiro de ontologias, que deve seguir uma metodologia de desenvolvimento de ontologias, tais como as metodologias discutidas em Cristani & Cuel [2005].

Uma vez identificados esses termos ontológicos, o (u1) analista deve então mapeá-los para os conceitos, relações e axiomas definidos no (g1) modelo ontológico de informações de contexto. Isso permite a identificação de termos que a aplicação pode reusar do modelo de informação contextual utilizado como artefato auxiliar. Como resultado, o analista produz o (d3) documento de reúso do modelo.

6.2.4 Análise e especificação de extensão do modelo (a1.4)

Na sub-atividade de análise e especificação de extensão do modelo, o (u1) analista identifica as informações de contexto da aplicação que não são modeladas pelo (g1) modelo ontológico de informações de contexto.

Durante esta sub-atividade, o analista deve especificar, com o auxílio de um engenheiro de ontologias, os termos ontológicos que compõem uma extensão ao modelo de informação contextual existente, extensão essa que deve ser formalizada no artefato de *saída* (d4) documento de modelo ontológico da aplicação, ou seja, a própria ontologia da aplicação.

Uma questão importante que o analista deve considerar durante a especificação dos novos termos ontológicos é a expressividade da linguagem de ontologia utilizada no (g1) modelo ontológico de informações de contexto. Como indicado na Figura 6.2, para gerar o (d4) documento de modelo ontológico da aplicação, o analista deve utilizar os artefatos de *entrada*: (g1) modelo ontológico de informações de contexto, (d2) documento de informações de contexto e (d3) documento de reúso do modelo. O artefato de *saída* desta sub-atividade, o (d4) documento de modelo ontológico da aplicação, é utilizado como *entrada* para a (a2) atividade de projeto.

6.3 Atividade de projeto (a2)

A Figura 6.3 ilustra a atividade de projeto do processo *POCAp*, que compreende três sub-atividades: (a2.1) projeto de reúso de serviços, (a2.2) projeto de extensão de serviços e (a2.3) projeto de novos serviços. Essas sub-atividades utilizam três artefatos de *entrada*: o (g2) documento de descrição de infra-estrutura de serviços, que é utilizado como artefato auxiliar para os propósitos de processamento de informações de contexto, o (d1) documento de requisitos, gerado pela (a1.1) sub-atividade de análise e especificação de requisitos (Figura 6.2), e o (d4) documento de modelo ontológico da aplicação, gerado pela (a1.4) sub-atividade de análise e especificação de extensão do modelo (Figura 6.2).

É importante observar que o (g2) documento de descrição de infra-estrutura de serviços fornece informações detalhadas a respeito de uma infra-estrutura de software que, por um lado, é capaz de interpretar a semântica de modelos ontológicos de informação contextual e, por outro lado, pode fornecer serviços comuns demandados por

6.3.2 Projeto de extensão de serviços (a2.2)

A sub-atividade de projeto de extensão de serviços identifica quais operações de serviços fornecidas pela infra-estrutura de serviços existente não atendem aos requisitos funcionais e não-funcionais de uma aplicação, conforme especificados no (d1) documento de requisitos.

O projetista deve combinar esses requisitos com o (g2) documento de descrição de infra-estrutura de serviços para identificar quais serviços devem ser estendidos. O nível de configurabilidade da infra-estrutura de serviços é uma importante questão que o projetista deve considerar — o projetista deve especificar essas novas operações de serviços para manipular os termos ontológicos definidos no (d4) documento de modelo ontológico da aplicação. Como resultado, o projetista gera o (d6) documento de descrição de extensão de serviços, que também é usado como artefato de *entrada* para a (a3) atividade de desenvolvimento.

6.3.3 Projeto de novos serviços (a2.3)

Na sub-atividade de projeto de novos serviços, o projetista identifica quais serviços não são fornecidos pela infra-estrutura de serviços existente, de acordo com o (g2) documento de descrição de infra-estrutura de serviços, para atender aos requisitos funcionais e não-funcionais descritos no (d1) documento de requisitos.

Assim como na (a2.2) sub-atividade de projeto de extensão de serviços, o projetista deve também especificar os novos serviços demandados pela aplicação em questão para explorar a semântica do (d4) documento de modelo ontológico da aplicação. Como resultado, esta sub-atividade produz o (d7) documento de descrição de novos serviços, que contém a especificação dos novos serviços a serem implementados. Esse documento é usado como artefato de *entrada* para a (a3) atividade de desenvolvimento.

6.4 Atividade de desenvolvimento (a3)

Para orientar as tarefas de (u3) desenvolvedores, a atividade de desenvolvimento do processo *POCap* utiliza os seguintes artefatos de *entrada*: o (d4) documento de modelo ontológico da aplicação, gerado pela sub-atividade (a1.4), como ilustra a Figura 6.2, e os documentos relacionados a serviços — (d5) documento de descrição de reuso de serviços, (d6) documento de descrição de extensão de serviços e (d7) documento de descrição de novos serviços — gerados, respectivamente, pelas sub-atividades (a2.1), (a2.2) e (a2.3), apresentadas na Figura 6.3. O artefato de *saída* da atividade de desenvolvimento é a implementação da aplicação sensível a contexto.

Baseado no (d4) documento de modelo ontológico da aplicação, o desenvolvedor deve escolher o suporte computacional apropriado para processar a linguagem de

ontologia utilizada. Por exemplo, a linguagem OWL, apresentada no Capítulo 3, é suportada pela maioria das APIs e arcabouços para processamento de ontologias.

De acordo com os artefatos de *entrada* relacionados a serviços produzidos na (a2) atividade de projeto, desenvolvedores devem considerar as linguagens, técnicas e ferramentas computacionais mais adequadas para atender às necessidades de projeto de cada serviço a ser reusado, estendido e implementado. Por exemplo, quanto ao processamento de informações de contexto baseadas em ontologias, três serviços podem ser considerados: o serviço de armazenamento, o serviço de consulta e o serviço de inferência.

O desenvolvedor deve optar por um serviço de armazenamento que forneça mecanismos eficientes para o armazenamento e para a recuperação de informações de contexto baseadas em ontologias.

Ao desenvolver um serviço de consulta, o desenvolvedor deve escolher linguagens de consulta para informações baseadas em ontologias que sejam compatíveis com a linguagem de ontologia utilizada no (d4) documento de modelo ontológico da aplicação. Além disso, a linguagem de consulta deve também atender à expressividade requisitada por cada consulta especificada na (a2) atividade de projeto.

Ao desenvolver um serviço de inferência, o desenvolvedor precisa identificar as máquinas de inferência apropriadas para atender às necessidades de projeto especificadas no (d4) documento de modelo ontológico da aplicação e nos artefatos relacionados a serviços (d5, d6 e d7). Por exemplo, existem máquinas de inferência que suportam vários tipos de inferência, assim como diferentes linguagens de ontologias e linguagens de regras com vários níveis de expressividade.

É importante observar que o processo *POCAp* é neutro com respeito à tecnologia utilizada para apoiar o desenvolvimento de aplicações cujas informações de contexto são baseadas em ontologias. Esse fato oferece liberdade ao desenvolvedor na escolha do suporte computacional necessário para transformar os artefatos da (a2) atividade de projeto na implementação da aplicação sensível a contexto.

6.5 Atividade de verificação e validação (a4)

A atividade de verificação e validação consiste em executar a aplicação sensível a contexto com casos de teste derivados da especificação das informações de contexto a serem processadas pela aplicação.

Os artefatos de *entrada* atividade de verificação e validação são o (d1) documento de requisitos, o (d3) documento de reúso do modelo, o (d4) documento de modelo ontológico da aplicação, gerados pela (a1) atividade de análise e especificação, e a implementação da aplicação sensível a contexto, produzida na (a3) atividade de desenvolvimento.

Requisitos funcionais são usados na (a4) atividade de verificação e validação para checar se a aplicação atende às especificações correspondentes. Uma vez que o analista estende o (g1) modelo ontológico de informações de contexto que o processo *POCAp* assume existir para gerar o (d4) documento de modelo ontológico da aplicação e o (d3) documento de reuso do modelo, durante esta atividade é importante avaliar se essas duas especificações são apropriadamente tratadas na implementação da aplicação. Isto é relevante a fim de melhor acomodar futuras extensões na aplicação. Os serviços de um engenheiro de ontologias podem ser úteis durante esta atividade.

No caso de aplicações cujas informações de contexto são baseadas em ontologias, (u4) validadores devem também preocupar-se com requisitos não-funcionais. Para ambientes de computação sensível a contexto, a interoperabilidade entre sistemas de software heterogêneos é um dos desafios a serem tratados. Validadores devem avaliar a adequação de padrões para promover a interoperabilidade não apenas entre aplicações, mas também entre serviços de infra-estruturas de software.

Outro requisito não-funcional importante é o armazenamento de informações de contexto. Validadores devem avaliar as estratégias de implementação utilizadas para prever o armazenamento adequado de informações de contexto baseadas em ontologias.

O desempenho de aplicações sensíveis a contexto, por sua vez, pode ser afetado segundo as características das máquinas de inferência utilizadas por desenvolvedores. Validadores devem avaliar o tempo de resposta de inferência sobre informações de contexto baseadas em ontologias considerando os requisitos de desempenho da aplicação. Além disso, validadores devem considerar os tempos intermediários que compõem um processo de inferência para identificar pontos de otimização no desempenho da aplicação.

6.6 Considerações finais

Este capítulo apresentou o processo de software *POCAp* como uma alternativa para a demanda por técnicas de Engenharia de Software para apoiar o desenvolvimento de aplicações sensíveis a contexto. Esse processo compreende um conjunto de atividades para analisar, especificar, projetar, implementar e testar aplicações cujas informações de contexto são baseadas em ontologias.

Assim, este trabalho reforça a importância de um engenheiro de ontologias para apoiar uma equipe de desenvolvimento nas atividades que exigem conhecimento relativo a metodologias para construção de ontologias, e ao projeto, implementação e teste de serviços habilitados para processar a semântica fornecida por ontologias.

Quanto à construção de aplicações sensíveis a contexto, outra característica do processo *POCAp* é que este é independente tanto do modelo ontológico de informação

contextual quanto da infra-estrutura de serviços existentes. Esse fato confere ao processo *POCAp* a característica de ser genérico quanto ao desenvolvimento de aplicações sensíveis a contexto, especialmente aquelas cujas informações de contexto são apoiadas por ontologias.

Nesse interim, o próximo capítulo descreve como o processo *POCAp* é instanciado ao utilizar como modelo e infra-estrutura de apoio, respectivamente, o modelo *SeCoM* e a infra-estrutura de serviços *SCK*. O próximo capítulo também relata a utilização do processo *POCAp* na extensão de uma aplicação Web com informações de contexto instanciadas a partir do modelo *SeCoM*, e a integração da mesma com os serviços de armazenamento, consulta e inferência da infra-estrutura *SCK*.

Instanciação do Processo *POCAp*

Este capítulo descreve como utilizar o processo *POCAp* para a construção de aplicações sensíveis a contexto baseadas em ontologias. Inicialmente, o processo é instanciado para construir aplicações cujas informações de contexto são apoiadas pelo modelo *SeCoM*, como modelo ontológico de informação contextual, e pela infra-estrutura de serviços *SCK*, como infra-estrutura computacional para interpretação da semântica desse classe de modelos [Bulcão Neto et al., 2006b].

De maneira complementar, é apresentada a instanciação do processo *POCAp* para construir uma aplicação sensível a contexto a partir do modelo *SeCoM* e da infra-estrutura de serviços *SCK*.

Para finalizar este capítulo, são apresentadas as considerações finais quanto ao mecanismo de instanciação do processo *POCAp*, que também incluem questões de projeto que desenvolvedores de aplicações sensíveis a contexto baseadas em ontologias devem considerar.

7.1 Os artefatos *SeCoM* e *SCK* no processo *POCAp*

Como o processo *POCAp* prevê a existência de um (g1) modelo de informações de contexto baseado em ontologias (atividade (a1) na Figura 6.2, página 115) e de um (g2) documento de descrição de infra-estrutura de serviços (atividade (a2) na Figura 6.3, página 117), este trabalho descreve como o modelo *SeCoM* e a infra-estrutura de serviços *SCK* podem ser utilizados como artefatos auxiliares para a construção de aplicações sensíveis a contexto. As quatro atividades do processo *POCAp* instanciadas

com o *SeCoM* e a *SCK* são apresentadas a seguir. Documentos e atividades numerados ao longo deste capítulo referem-se às Figuras 6.1, 6.2 e 6.3 do Capítulo 6, respectivamente encontradas nas páginas 113, 115 e 117.

Uma instância da (a1) atividade de análise e especificação

Uma vez que as sub-atividades de (a1.1) análise e especificação de requisitos e (a1.2) análise e especificação de informação contextual são independentes do (g1) modelo de informações de contexto baseado em ontologias, o (u1) analista produz o (d1) documento de requisitos e o (d2) documento de informações de contexto de maneira independente do modelo de informação contextual *SeCoM*.

A primeira sub-atividade dependente de um modelo ontológico é a (a1.3) sub-atividade de análise e especificação de reuso do modelo. Neste caso, o modelo *SeCoM* é usado como um artefato auxiliar para que o analista possa mapear e combinar o (d2) documento de informações de contexto de acordo com os termos ontológicos fornecidos na forma de atores, dispositivos, localização, tempo e atividades, tal como fornecido pelo modelo *SeCoM* (vide Figura 4.1, página 63). Em seguida, o analista produz o (d3) documento de reuso do modelo, que contém os termos ontológicos da aplicação que o modelo *SeCoM* representa.

Durante a (a1.4) sub-atividade de análise e especificação de extensão do modelo, os termos que o modelo *SeCoM* não representa são modelados pelo analista como uma extensão desse modelo. Por exemplo, se uma aplicação sensível a contexto demanda a representação de atributos físicos de pessoas — por exemplo, altura e peso — o analista define uma ontologia particular para representar tal tipo de informação. O modelo *SeCoM* é estendido quando a nova ontologia de atributos físicos importa a ontologia *Actor* em que pessoas já são representadas, como (`act:Person`, `rdf:subClassOf`, `act:Actor`). Os termos identificados nas (a1.3 e a1.4) sub-atividades de reuso e extensão do modelo formam o (d4) documento de modelo ontológico da aplicação.

Uma instância da (a2) atividade de projeto

Baseado tanto no (d1) documento de requisitos, gerado na sub-atividade (a1.1), quanto no (d4) documento de modelo ontológico da aplicação, gerado na sub-atividade (a1.4), o (u2) projetista deve combinar as demandas da aplicação em questão em termos de armazenamento, consulta e inferência da semântica de informações de contexto com os correspondentes serviços de uma infra-estrutura, como a *SCK*.

A característica de configurabilidade da infra-estrutura *SCK*, em particular, visa atender a uma ampla variedade de requisitos de armazenamento, consulta e inferência. Quando as operações disponíveis nos serviços da infra-estrutura *SCK* não

atendem aos requisitos de uma aplicação, os serviços precisam ser estendidos para implementar a operação requisitada — isso demanda que o projetista especifique as novas operações de acordo com o mecanismo de extensão aceito pela infra-estrutura SCK.

Se uma aplicação sensível a contexto demanda um serviço que não é fornecido pela infra-estrutura SCK, o projetista deve então especificar a estrutura desse serviço para fins de implementação. Se o novo serviço precisa manipular a semântica de informações de contexto, ele tem de explorar a semântica definida no (d4) documento de modelo ontológico da aplicação.

Uma instância da (a3) atividade de desenvolvimento

Nesta atividade, (u3) desenvolvedores podem se beneficiar da utilização do modelo semântico de informações de contexto *SeCoM* e da infra-estrutura de serviços SCK.

Com relação ao modelo *SeCoM*, todo o conhecimento de uma aplicação sensível a contexto está separado de sua lógica de programação. Essa separação de interesses é uma estratégia clássica da Engenharia de Software que tem por objetivo facilitar a manutenção, a portabilidade e a evolução da lógica de uma aplicação sensível a contexto. Uma vez que o modelo *SeCoM* é baseado na linguagem de ontologia OWL, ele adiciona a uma aplicação a habilidade de poder inferir sobre informações de contexto instanciadas a partir desse modelo.

Os serviços da infra-estrutura SCK também facilitam o desenvolvimento de aplicações sensíveis a contexto devido às seguintes razões: (a) essas aplicações não necessitam implementar toda sua infra-estrutura para armazenar, consultar e inferir informações de contexto; (b) a infra-estrutura SCK utiliza um formato uniforme e padrão de intercâmbio de informações de contexto, no caso, o modelo de triplas RDF [Manola & Miller, 2004]; e (c) aplicações se beneficiam do aspecto de configurabilidade dos serviços da infra-estrutura SCK.

Uma instância da (a4) atividade de verificação e validação

Durante esta atividade, os (u4) validadores devem ter uma preocupação especial com os requisitos não-funcionais.

Considerando a expressividade de modelos de informação contextual, algumas ontologias do modelo *SeCoM* têm o mais alto nível de expressividade que uma ontologia OWL pode alcançar. Por exemplo, as ontologias *Activity* e *Spatial* possuem a expressividade de Lógica de Descrições do tipo SHOIN(D)¹. Isto tem implicações quanto à complexidade de tempo de processos de inferência que é, por sua vez, uma

¹Lógica de atributos, complemento, propriedades transitiva e inversa, hierarquia de propriedades, nominais, restrições de número não-qualificadas e tipos de dados.

preocupação relacionada ao desempenho de sistemas computacionais.

O modelo *SeCoM* é um modelo de informação contextual viável porque sua arquitetura em duas camadas permite que aplicações o reusem e/ou estendam para seus propósitos. A viabilidade é comprovada por medidas realizadas com respeito ao tempo de importação de cada ontologia do modelo *SeCoM*, que é, em média, menor que 2% do tempo de resposta total de inferência sobre cada ontologia.

Com relação a requisitos não-funcionais, este trabalho sugere que validadores investiguem, principalmente, o comportamento temporal do serviço de inferência da infra-estrutura *SCK*. Devem ser consideradas, por exemplo, a capacidade de inferência e a variedade de serviços suportadas por cada máquina de inferência a ser utilizada com o serviço em questão. A implementação atual do serviço de inferência fornece suporte completo a informações de contexto codificadas em *OWL*, assim como a habilidade de medir tempos intermediários do processo de inferência.

7.2 Aplicação WebMemex: Estudo de caso do processo POCAp

Esta seção descreve uma instanciação do processo *POCAp* com o objetivo de estender a aplicação *WebMemex* [Macedo et al., 2003; Truong & Abowd, 2004] com informações de contexto cuja semântica é apoiada pelo modelo *SeCoM* e interpretada pela infra-estrutura de serviços *SCK*. Todas as etapas do processo *POCAp* aplicadas no desenvolvimento da aplicação *WebMemex* foram realizadas conforme detalhado por Bulcão Neto et al. [2006b].

7.2.1 Aplicação WebMemex

WebMemex é uma aplicação que captura e armazena o histórico de navegação Web de usuários; a partir do histórico, usuários podem recomendar páginas Web uns aos outros. Quando um usuário deseja recomendar uma página Web, ele pode fazê-lo para um grupo ou comunidade de usuários qualquer, a todos os usuários de suas comunidades *on-line*, ou com base em um critério específico, como recomendação a grupos de amigos, colaboradores ou colegas de trabalho. Na implementação descrita nesta seção, a recomendação é feita de modo manual (ou explícita).

Originalmente, há duas versões da aplicação *WebMemex* que exploram a recomendação automática (ou implícita) de páginas Web, ou seja, a própria aplicação recomenda páginas Web a usuários de acordo com as páginas Web visitadas. Apesar de ambas versões explorarem recomendação automática de conteúdo Web, estas se diferenciam quanto à técnica de recuperação de informação que habilita este tipo de recomendação [Macedo et al., 2003; Truong & Abowd, 2004]. Por não ser o foco deste trabalho, optou-se por habilitar apenas a recomendação de páginas Web por parte

dos próprios usuários, com o objetivo de incentivar a comunicação e a colaboração interpessoal [Bulcão Neto et al., 2005a]. Uma interface típica da aplicação *WebMemex* é apresentada na Figura 7.1.

Figura 7.1: A aplicação *WebMemex*, em primeiro plano, oferece ao usuário corrente a possibilidade de recomendar a página Web, apresentada em segundo plano, a grupos de usuários — representados por uma *combo box* — por meio do botão *Send it!* [Bulcão Neto et al., 2005a].

7.2.2 Atividade de análise e especificação (a1)

Seguindo o fluxo de sub-atividades da (a1) atividade de análise e especificação do processo *POCAp* (Figura 6.1, página 113), foi definido, primeiramente, o conjunto de requisitos funcionais e não-funcionais da aplicação *WebMemex* na (a1.1) sub-atividade de análise e especificação de requisitos (Figura 6.2, página 115).

Análise e especificação de requisitos (a1.1)

É apresentada a seguir uma lista com os requisitos funcionais voltados para a extensão da aplicação *WebMemex*, requisitos esses que têm relação com as funções que devem ser re-escritas para utilizar informação de contexto semântica. Isto é possível por se tratar da extensão de uma aplicação existente, o que fornece ao analista indícios de que funcionalidades são candidatas à re-escrita para inclusão de informação de contexto semântica.

- R1** A aplicação deve oferecer a opção para *criar usuários*;
- R2** A aplicação deve oferecer a opção para *criar grupos*;
- R3** A aplicação deve oferecer os critérios sociais de *amigos, colaboradores de trabalho e colegas de trabalho* para a criação de grupos;
- R4** A aplicação deve oferecer a opção para *inserir usuários em grupos*;
- R5** A aplicação deve oferecer a opção para *autenticar usuários*;
- R6** A aplicação deve oferecer a opção para *listar todos os grupos em que um usuário está cadastrado*;
- R7** A aplicação deve *manter o histórico de navegação Web de cada usuário*;
- R8** A aplicação deve oferecer a opção para *recomendar a página Web corrente*;
- R9** A aplicação deve oferecer a opção para *listar páginas Web recomendadas por outrem*.

Dos requisitos apresentados, apenas **[R3]** é um novo requisito em relação às versões anteriores da aplicação *WebMemex*, pois é a partir desse requisito que a aplicação deve habilitar aos seus usuários a recomendação explícita de páginas Web.

Uma vez definidos os requisitos funcionais, foram elaborados os requisitos não-funcionais para a aplicação *WebMemex*, que são listados a seguir. Para isso, foi utilizada a especificação ISO/IEC 9126-1 [ISO, 2001], que define características de qualidade de apoio ao processo de avaliação de software, características essas que podem ser utilizadas para identificar requisitos de software.

1. Funcionalidade - Interoperabilidade

- R10** Para permitir que a aplicação *WebMemex* possa interagir com outros sistemas, as informações gerenciadas por esta devem *ser representadas, armazenadas e consultadas por meio de mecanismos-padrão*.

2. Eficiência - Comportamento no Tempo

- R11** Ao substituir os mecanismos de recuperação de informação das versões anteriores da aplicação *WebMemex*, o analista definiu que a recomendação de páginas Web fosse apoiada por mecanismos de interpretação de informações de contexto representadas por ontologias. Assim, deve-se *utilizar mecanismos de interpretação que não degradem o tempo de resposta da aplicação*.

Juntos, requisitos funcionais e não-funcionais formam o (d1) documento de requisitos da aplicação *WebMemex*, delimitando assim, o término da (a1.1) sub-atividade de análise e especificação de requisitos.

Análise e especificação de informação contextual (a1.2)

Durante esta sub-atividade, o analista identificou o conjunto de termos utilizado pela aplicação *WebMemex* para habilitar o serviço de recomendação explícita de navegação Web. O (d1) documento de requisitos, gerado na (a1.1) sub-atividade de análise e especificação de requisitos, foi utilizado como base para essa tarefa de identificação, como listado a seguir.

Com base nos requisitos **[R1]** e **[R2]**, foi decidido que deve haver um conjunto de metadados para descrever usuários e grupos de usuários durante sua criação. Para usuários, foram incluídos “nome completo”, “primeiro nome” e “sobrenome”; para grupos de usuários, incluiu-se um nome que descreve o grupo.

A partir do requisito **[R3]**, o analista criou três relacionamentos entre usuários: de amigos, de colaboradores de trabalho e de colegas de trabalho. Foi decidido também que um grupo deve ter um rótulo referente ao tipo de relação social existente entre os membros desse grupo, rótulo este chamado de “tipo de grupo”.

O requisito **[R4]** requer a definição de uma propriedade que indique que um usuário pode ser membro de um ou mais grupos de usuários. Ao mesmo tempo, faz-necessária a definição de uma propriedade que indique que um grupo pode ter um ou mais usuários.

O requisito **[R5]** aponta a necessidade por metadados para autenticação de usuários, como “nome de acesso” e “senha”.

Para atender ao requisito **R6**, foi incluída uma propriedade que relaciona cada grupo ou comunidade ao usuário que criou o grupo, chamada “criador”.

Com base no requisito **[R7]**, o analista incluiu a entidade “página Web” ao modelo da aplicação *WebMemex*. Foi também criada uma relação para descrever que cada usuário tem seu próprio histórico de páginas Web visitadas. Como cada página Web pode ser navegada por um ou mais usuários, criou-se uma propriedade que relaciona páginas Web a usuários, chamada “navegada por”. Esse requisito também demanda uma propriedade que descreva o instante de tempo em que cada página foi navegada por um determinado usuário, chamada “data de navegação”. Metadados comuns de páginas Web foram também incluídos, como “URL” e “título” da página.

A partir dos requisitos **[R8]** e **[R9]**, criou-se uma diferenciação entre páginas Web navegadas e páginas Web recomendadas, podendo assim gerar dois tipos de históricos diferentes para cada usuário. Para isso, foram criadas a entidade “página Web recomendada”, a relação “histórico de recomendação” e propriedades que relacionam

essa entidade ao instante de tempo em que foi recomendada e ao usuário que a recomendou, respectivamente, chamadas “data de recomendação” e “recomendador”.

Todos os termos identificados nesta atividade compõem o (d2) documento de informações de contexto, que é utilizado como entrada das (a1.3 e a1.4) sub-atividades de análise e especificação de reuso e extensão do modelo ontológico subjacente, neste caso, do modelo *SeCoM*. Um editor de ontologias para a linguagem OWL pode auxiliar o analista durante ambas atividades, como as ferramentas Protégé [Gennari et al., 2003] e SWOOP [Kalyanpur et al., 2005].

Análise e especificação de reuso do modelo (a1.3)

Nesta sub-atividade, o analista identifica os conceitos, as relações e os axiomas encontrados nas ontologias do modelo *SeCoM* que não necessitam de mudanças em suas definições para atender ao modelo de dados da aplicação definido no (d2) documento de informações de contexto. As ontologias que contiverem tais termos serão importadas a fim de compor o modelo ontológico da aplicação *WebMemex*.

Entidades de usuários e grupos de usuários são modeladas como subclasses de atores na ontologia *Actor* do modelo *SeCoM*. Usuários são modelados como classes de pessoas (`act:Person`), e grupos de usuários são subclasses de grupos (`act:PersonGroup`, `rdf:subClassOf, act:Group`), cujos membros são todas instâncias da classe de pessoas.

Os metadados “primeiro nome” e “sobrenome” de usuários são também fornecidos pela ontologia *Actor*, representados por `act:hasFirstName` e `act:hasSurname`, respectivamente. A propriedade `act:hasName` associa um nome a qualquer tipo de ator na ontologia *Actor*, ou seja, serve tanto para descrever o “nome completo” de usuários, quanto para descrever um grupo de usuários. Assim, a ontologia *Actor* é importada para a definição da ontologia da aplicação *WebMemex*.

Os três tipos de relações sociais definidos para a aplicação *WebMemex* podem ser importados diretamente da ontologia *Relationship* do modelo *SeCoM*. Tais relações entre usuários (ou pessoas) são `rel:isFriendOf` para amigos, `rel:cooperatesWith` para colaboradores de trabalho e `rel:worksWith` para colegas de trabalho. Logo, a ontologia *Relationship* é importada pela ontologia da aplicação *WebMemex*.

A ontologia *Actor* também fornece as propriedades `act:hasGroupMember`, que relaciona um grupo a um ou mais usuários, e `act:isMemberOf`, que é inversa da primeira, ou seja, relaciona um usuário a mais de um grupo.

A ontologia *Actor* também inclui a propriedade `act:maker` que relaciona cada grupo ao usuário que o criou.

A ontologia *Document* fornece a propriedade `doc:hasTitle`, que pode ser utilizada para descrever o “título” de páginas Web, que podem ser modeladas como documentos, mas que não são definidas no modelo *SeCoM*. A restrição de cardinalidade

definida para documentos é a mesma para páginas Web, ou seja, apenas um título descritivo por documento.

O conjunto dos termos identificados nesta atividade forma o (d3) documento de reuso do modelo que inclui, no caso, as ontologias *Actor*, *Document* e *Relationship* sem alterações nas definições dos respectivos termos citados.

Análise e especificação de extensão do modelo (a1.4)

Nesta sub-atividade, o analista identifica os termos encontrados nas ontologias do modelo *SeCoM* que necessitam de mudanças em suas definições, ou que não são aceitos para atender às definições dos termos descritos no (d2) documento de informações de contexto. As ontologias que contiverem tais termos também são importadas para compor o modelo ontológico da aplicação *WebMemex*.

A ontologia *Actor*, por exemplo, não inclui a propriedade que descreve o tipo de relação social entre os membros desse grupo. Por isso, o analista estende essa ontologia ao criar a propriedade *wmx:hasGroupType* com respectivo domínio de grupos de usuários (*act:PersonGroup*) e valor como uma seqüência de caracteres (*xsd:string*). Foi definido também que essa propriedade pode ocorrer uma única vez para cada grupo, ou seja, de cardinalidade máxima igual a 1. O espaço de nomes identificado como *wmx:* é utilizado para referenciar os termos da ontologia da aplicação *WebMemex*.

Os metadados “nome de acesso” e “senha”, utilizados para autenticação de usuários, não são também definidos na ontologia *Actor*. Logo, o analista criou duas propriedades para representar esses termos, *wmx:hasLogin* e *wmx:hasPassword*, respectivamente, ambas com domínio de usuários (*act:Person*), valor como seqüências de caracteres e de cardinalidade máxima igual a 1.

Para modelar “páginas Web”, o analista importou a ontologia *Document* e incluir páginas Web como subclasses da classe *doc:Document*. A propriedade *wmx:hasURL* de páginas Web com restrição de cardinalidade igual a 1 foi adicionada ao modelo da aplicação *WebMemex*. Para manter o histórico de navegação de usuários foi criada a relação *wmx:hasWebLog*, cujo valor assumido pode ser zero ou mais instâncias da classe de páginas Web. A propriedade *wmx:hasBrowsingDate* foi adicionada para armazenar o instante de tempo, definido na ontologia *Time*, em que cada página Web é navegada por um usuário. Assim, tanto a ontologia *Document* quanto a ontologia *Time* são importadas para compor a ontologia da aplicação *WebMemex*.

Para diferenciar entre páginas Web navegadas e páginas Web recomendadas, o analista criou a classe de “páginas Web recomendadas” como subclasse de páginas Web (*wmx:RecommendedWebPage*, *rdf:subClassOf*, *wmx:WebPage*). Da mesma forma, para gerar o histórico de páginas Web recomendadas, o analista criou a propriedade *wmx:hasRecommendedWebLog* como sub-propriedade de *wmx:hasWebLog*, cujo valor assumido pode ser zero ou mais instâncias da classe de páginas Web recomendadas.

Figura 7.2: Representação gráfica da ontologia da aplicação *WebMemex* gerada por um plugin [ezOWL, 2006] do editor de ontologias Protégé [Gennari et al., 2003].

Foram também criadas propriedades para relacionar o instante de tempo em que uma página Web foi recomendada e o usuário que a recomendou, respectivamente indicadas por *wmx:hasRecommendationDate* e *wmx:isRecommendedBy*.

O conjunto dos termos identificados nesta atividade forma o (d4) documento de modelo ontológico da aplicação, ou seja, a ontologia da aplicação *WebMemex*.

A Figura 7.2 ilustra a ontologia da aplicação *WebMemex* com termos reusados e estendidos das ontologias *Actor*, *Document*, *Relationship* e *Time*. Relações inversas às relações *wmx:hasBrowsingDate*, *wmx:hasRecommendationDate* e *wmx:isRecommendedBy* foram também definidas. *wmx:isBrowsingDateOf* e *wmx:isRecommendationDateOf* são, respectivamente, os instantes de tempo em que uma página Web foi visitada e recomendada, enquanto que *wmx:recommends* relaciona as páginas Web que um usuário recomendou. Essas novas relações facilitam a recuperação de informações referentes aos históricos de navegação e de recomendação de páginas Web.

A Tabela 7.1 caracteriza a ontologia da aplicação *WebMemex*. Como ela importa ontologias baseadas em Lógica de Descrições com expressividade SHOIF(D)² — como as ontologias *Document* e *Time* — a ontologia da aplicação *WebMemex* herda essas características. Os números de triplas, classes, propriedades e instâncias foram calculados por meio de um programa distribuído com a máquina de inferência *Pellet 1.3beta2* [Sirin et al., 2006].

²Lógica de atributos, complemento, propriedades funcional, transitiva e inversa, hierarquia de propriedades, nominais e tipos de dados.

Tabela 7.1: Caracterização da ontologia da aplicação *WebMemex*. Adaptado de Bulcão Neto & Pimentel [2006a].

Ontologia	Tipo OWL	Complexidade	Triplas	Classes	Propriedades	Instâncias
WebMemex	DL	SHOIF(D)	720	62	94	19

7.2.3 Atividade de projeto (a2)

Utilizando a infra-estrutura de serviços *SCK* como artefato auxiliar para a extensão da aplicação *WebMemex*, o (u2) projetista deve definir que serviços devem ser reusados e estendidos e quais devem ser criados para atender à demanda dessa aplicação, como mostrado na Figura 6.3, página 117.

Projeto de reúso de serviços (a2.1)

A arquitetura da aplicação *WebMemex* inclui cinco elementos básicos: (a) um servidor *Web proxy*, que recebe cada solicitação HTTP de navegadores dos usuários; (b) o componente *filtrador*, que checa se a página Web que o usuário está navegando é uma página convencional — de extensões *.HTML* ou *.htm* — ou uma página de controle de usuários e grupos — de extensão *.wm* (acrônimo de *WebMemex*); (c) o componente *gerenciador de usuários e grupos* (ou comunidades); (e) o componente *extrator*, que extrai e armazena metadados de cada página Web navegada; (e) o componente *recomendador*, que permite que usuários recomendem páginas Web uns aos outros.

A partir do (d1) documento de requisitos da aplicação *WebMemex* e da disposição dos elementos de sua arquitetura, apenas o servidor *Web proxy* e o componente *filtrador* não precisam ser modificados. Ambos componentes não necessitam inserir, consultar, inferir, ou outra função que manipule a semântica dos termos definidos na (a1) atividade de análise e especificação.

Considerando os requisitos [R1] a [R7], o componente *gerenciador de usuários e grupos* e o componente *extrator* são candidatos à re-implementação. Ambos componentes atuam como fontes de informações de contexto *SCK*, como informações de usuários, grupos e relações sociais entre usuários, e páginas Web e respectivos metadados. O componente *gerenciador de usuários e grupos* atua também como consumidor de informações de contexto *SCK*, uma vez que consulta as informações por ele armazenadas.

Quanto aos serviços de armazenamento e consulta da infra-estrutura *SCK*, ambos atendem à demanda de inserção e consulta de modelos RDF referentes a usuários, grupos, relações sociais entre usuários, e páginas Web. Independentemente do (d4) documento de modelo ontológico da aplicação, o serviço de armazenamento registra modelos RDF de forma genérica, e o serviço de consulta se apóia no poder de

expressão das linguagens suportadas, como as linguagens RDQL [Miller et al., 2002] e SPARQL [Prud'hommeaux & Seaborne, 2006]. Todas as operações de consulta necessárias para atender aos requisitos funcionais da aplicação *WebMemex* são suportadas pelas linguagens mencionadas.

Para atender ao requisito **[R10]** de interoperabilidade, todas as informações de aplicações gerenciadas pela infra-estrutura SCK são armazenadas segundo um esquema relacional específico para modelos RDF, amplamente explorado para Web Semântica. O mesmo requisito é também atendido pelo serviço de consulta devido à utilização de linguagens que são padrões de consulta de dados RDF, como a SPARQL.

O projetista definiu para o componente gerenciador de usuários e grupos que este deveria fornecer os seguintes métodos: (a) cadastrar usuário, (b) checar se usuário é válido via nome de acesso e senha, (c) checar se usuário existe via nome de acesso, (d) modificar senha de usuário, (e) cadastrar grupo com o critério de relação social, (f) associar usuários a um grupo, (g) checar se grupo existe via nome do grupo e tipo de relação social, (h) obter nome e tipo de relação social de um grupo, (i) obter nome do criador de um grupo, (j) obter membros de um grupo, (k) obter grupos cadastrados, (l) verificar se usuário é membro de um grupo particular e (m) obter todos os grupos dos quais um usuário é membro.

Para o componente extrator de metadados de páginas Web navegadas, o projetista definiu os seguintes métodos: (a) cadastrar página Web, (b) checar se página Web fora navegada por um usuário específico, (c) checar se página Web fora navegada por qualquer usuário, (d) obter histórico de navegação de usuário; (e) obter metadados de página Web.

O componente recomendador da aplicação *WebMemex* também é candidato a modificações para processar a semântica do modelo ontológico da aplicação. O serviço de inferência da infra-estrutura SCK foi utilizado para habilitar a recomendação manual de páginas Web, proposta no requisito **[R8]**. Para isso, foi definido que na recomendação de páginas Web, o tipo de relação social entre usuários deveria ser o critério para recomendação, tal como recomendar uma página a todos os amigos de um usuário, ou àqueles com quem este trabalha diretamente (colaboradores).

Para que o componente recomendador atenda ao requisito **[R11]**, o projetista selecionou uma máquina de inferência que, embora com simples capacidade de processamento, é capaz de interpretar a semântica de relações sociais entre usuários definida no (d4) documento de modelo ontológico da aplicação, com tempo de resposta aceitável. Ao criar um grupo, um usuário escolhe o tipo de relação social — *rel:isFriendOf*, *rel:worksWith* ou *rel:cooperatesWith* — existente entre os membros desse grupo. Esses três tipos de relação social são modelados na ontologia *Relationship* como sub-propriedades da relação *rel:knows*, que tem característica simétrica, ou seja, se um usuário X é amigo de um usuário Y, então Y também é amigo de X. Como

Figura 7.3: A arquitetura de componentes da aplicação *WebMemex* integrada aos serviços da infra-estrutura SCK. Sobre cada componente estão as informações de contexto que esse componente gerencia. Adaptado de Bulcão Neto et al. [2005a].

a relação de sub-propriedade é transitiva, então se um usuário X é amigo de um usuário Y, então X conhece Y. Por isso, dentre as máquinas de inferência suportadas pela infra-estrutura SCK, o projetista optou pela máquina de inferência para relações transitivas *TransitiveReasoner* [Carroll et al., 2004].

O componente recomendador deve dar suporte ao requisito **[R9]** para apresentar o histórico pessoal de páginas Web recomendadas. Como no caso dos componentes extrator e gerenciador de usuários e grupos, o serviço de consulta da infra-estrutura SCK atende à demanda de consulta de modelos RDF referentes a páginas Web. O mesmo se aplica ainda em relação ao requisito **[R10]** de interoperabilidade.

Segundo o projetista, o componente recomendador deve fornecer os seguintes métodos: (a) recomendar página Web, (b) inferir lista de usuários beneficiados com recomendação de páginas Web e (c) obter histórico de páginas Web recomendadas. Portanto, este é tanto fonte quanto consumidor de informações de contexto SCK.

Todas as decisões de projeto referentes à (a2.1) sub-atividade de projeto de reúso de serviços foram descritas no (d5) documento de descrição de reúso de serviços, que serviu de entrada para a (a3) atividade de desenvolvimento da aplicação *WebMemex*.

Em virtude dos serviços da infra-estrutura SCK atenderem aos requisitos relatados no (d1) documento de requisitos juntamente com o (d4) documento de modelo ontológico da aplicação *WebMemex*, as sub-atividades de (a2.2 e a2.3) projeto de extensão de serviços e de projeto de novos serviços não demandaram esforços. A Figura 7.3 mostra a arquitetura da aplicação *WebMemex* definida ao término da atividade de projeto.

As extensões SCK adicionadas a componentes na Figura 7.3 fazem o papel de tradutores de informações de contexto SCK, uma vez que mapeiam as informações

que os componentes manipulam segundo a (d4) ontologia da aplicação *WebMemex*.

A cada um desses componentes estão também relacionados os tipos de informações de contexto que devem ser processadas, bem como o respectivo serviço *SCK* a ser utilizado. Por exemplo, o componente extrator extrai metadados das páginas Web navegadas e os armazena no histórico de navegação do respectivo usuário ao invocar o serviço de armazenamento da infra-estrutura *SCK*.

7.2.4 Atividade de desenvolvimento (a3)

Nesta fase, o (u3) desenvolvedor da aplicação *WebMemex* utilizou como documentos de apoio o (d4) documento de modelo ontológico da aplicação, gerado na (a1.4) sub-atividade de análise e especificação de extensão do modelo, e o (d5) documento de descrição de reúso de serviços, produzido na (a2.1) sub-atividade de projeto de reúso de serviços. O primeiro documento contém as informações de contexto (ou conhecimento) da aplicação *WebMemex*, enquanto que o segundo contém a lógica que manipula essas informações.

Em geral, os métodos da aplicação *WebMemex* para inserir informações de contexto seguem a seguinte estrutura: (a) cria-se um grafo RDF para armazenar informação; (b) cria-se nesse grafo um recurso (ou nó) RDF para armazenar uma entidade em particular — usuário, grupo, página Web; e (c) associa-se a essa entidade um conjunto de metadados segundo o modelo ontológico subjacente. O grafo RDF como um todo é um dos parâmetros de entrada para o serviço de armazenamento da infra-estrutura subjacente [Bulcão Neto et al., 2005a].

O Exemplo 7.1 apresenta um trecho do método para cadastrar usuário. A linha 1 cria um grafo RDF vazio. A linha 2 insere nesse grafo um recurso referente ao usuário a ser cadastrado, associando-lhe uma URI que combina o espaço de nomes definido para a aplicação *WebMemex* (variável *wmx_ns*) e uma seqüência de caracteres aleatória (variável *userURI*). A linha 3 associa o tipo do recurso RDF, no caso, o recurso é da classe *oct:Person*, definida na ontologia *Actor*. As linhas 4 a 6 associam ao recurso corrente metadados definidos na ontologia *Actor*, respectivamente, “nome completo”, “primeiro nome” e “sobrenome”. Por fim, as linhas 7 e 8 associam ao recurso metadados definidos na própria ontologia da aplicação *WebMemex*, no caso, o “nome de acesso” e a “senha” do usuário.

```

0  <!--                                Exemplo 7.1                                -->
1  Model newUser = ModelFactory.createDefaultModel();
2  newUser.createResource(wmx_ns + userURI)
3      .addProperty(RDF.type, Actor.Person)
4      .addProperty(Actor.hasName, userName)
5      .addProperty(Actor.hasFirstName, firstName)
6      .addProperty(Actor.hasSurname, surname)
7      .addProperty(Webmemex.hasLogin, login)
8      .addProperty(Webmemex.hasPassword, password);

```


Métodos da aplicação *WebMemex* para consultar informações de contexto seguem, em geral, a seguinte estrutura: (a) define-se o conjunto de informações de contexto cujo valor se deseja recuperar; (b) cria-se uma consulta segundo a notação de uma linguagem para consulta de modelos RDF que, em geral, adota o mecanismo de triplas RDF; e (c) associa-se essa consulta a uma variável que será parâmetro de entrada do serviço de consulta da infra-estrutura subjacente [Bulcão Neto et al., 2005a].

O Exemplo 7.2 a seguir descreve um trecho de código com a consulta necessária para obter a lista de grupos dos quais um usuário é membro. A linha 1 representa a associação da consulta RDF a uma variável (identificada por *query*) que será entrada para um processador de consultas. A linha 2 indica a variável cujo valor se deseja recuperar, no caso, os nomes dos grupos (linha 7) dos quais faz parte uma pessoa (linha 3), cujo nome de acesso é identificado pela variável *login* (linha 4). As linhas 5 e 6 indicam que essa pessoa é membro de um grupo de pessoas do qual se deseja recuperar o nome, como indicado pelas linhas 2 e 7.

```
0  <!--                               Exemplo 7.2                               -->
1  String query =
2  "SELECT ?groupName " +
3  "WHERE (?user, <rdf:type>, <act:Person>)" +
4  "      (?user, <wmx:hasLogin>, '" + login + "')" +
5  "      (?group, <rdf:type>, <act:PersonGroup>)" +
6  "      (?group, <act:hasGroupMember>, ?user)" +
7  "      (?group, <act:hasName>, ?groupName)" +
```

O Exemplo 7.3 ilustra uma consulta realizada pelo componente extrator da aplicação *WebMemex*, em que é verificado se a página Web capturada pelo servidor Web *proxy* já consta no histórico de páginas navegadas do usuário corrente (variável *login* na linha 4). As linhas 1 e 5 indicam que a variável de retorno deve armazenar a URI de um recurso do tipo página Web (vide linha 6), cuja URL é indicada pela variável *url*, apresentada na linha 7.

```
0  <!--                               Exemplo 7.3                               -->
1  String query =
2  "SELECT ?webPage " +
3  "WHERE (?user, <rdf:type>, <act:Person>)" +
4  "      (?user, <wmx:hasLogin>, '" + login + "')" +
5  "      (?user, <wmx:hasWebLog>, ?webpage)" +
6  "      (?webpage, <rdf:type>, <wmx:WebPage>)" +
7  "      (?webpage, <wmx:hasURL>, '" + url + "')" +
```

O Exemplo 7.4 descreve um trecho de código do componente recomendador que infere qual usuário terá seu histórico de páginas Web recomendadas acrescido da página Web corrente. A linha 1 cria um grafo RDF em memória específico para o armazenamento de ontologias, no caso, dos termos da ontologia de relacionamento entre pessoas (vide linha 2), que será utilizada como critério para inferência. As

linhas 3 e 4 invocam o serviço de inferência da infra-estrutura SCK informando que a máquina de inferência a ser utilizada é a *TransitiveReasoner*, passando a configuração dessa máquina de inferência (variável *config*), o grafo RDF em memória da ontologia *Relationship* (variável *tBox*) e um grafo RDF com a lista de grupos dos quais o usuário que recomenda páginas é membro (variável *aBox*). Com esses dois grafos RDF de entrada, a máquina de inferência cria o grafo de inferência com o resultado de seu processamento (vide linha 4, variável *inf*).

```

0  <!--                                Exemplo 7.4                                -->
1  OntModel tBox = ModelFactory.createOntologyModel();
2  tBox.read(Relationship.getURI());
3  CKInference ckInf = new CKInference();
4  InfModel inf = ckInf.makeInference("TRANS", config, tBox, aBox);
5  Resource user = inf.createResource(app_ns + recommender);
6  NodeIterator it = inf.listObjectsOfProperty(user, Relationship.knows);

```

A linha 5 adiciona ao grafo de inferência um recurso que identifica o usuário que recomenda a página Web atualmente navegada (variável *user*). A linha 6 realiza uma consulta nesse grafo procurando por todas as pessoas que esse usuário conhece. Ou seja, este trecho de código é utilizado para o caso de esse usuário recomendar a página corrente a todas as pessoas com as quais mantém algum tipo de relação social. Considerando esta implementação da aplicação *WebMemex*, qualquer uma das propriedades suportadas pela aplicação é sub-propriedade da relação *rel:knows*, definida na ontologia *Relationship* do modelo *SeCoM* [Bulcão Neto et al., 2005a].

7.2.5 Atividade de verificação e validação (a4)

Durante esta atividade, o (u4) validador teve o trabalho principal de avaliar o requisito não-funcional **R11** de comportamento temporal da aplicação *WebMemex*. A seguir são apresentados os testes de avaliação de comportamento temporal da aplicação *WebMemex* realizados em um computador Intel^(R) Pentium^(R) 4 2.66 GHz, 1 GiB de memória principal com sistema operacional Linux Red Hat 7.3. Maiores detalhes podem ser encontrados em Bulcão Neto & Pimentel [2006a].

Tempos de inferência da ontologia da aplicação WebMemex

O primeiro teste realizado pelo validador teve como objetivo obter tempos intermediários do processo de inferência sobre a ontologia da aplicação *WebMemex* [Bulcão Neto & Pimentel, 2006a]. Para isso, o serviço de inferência da infra-estrutura SCK foi configurado para utilizar a máquina de inferência *Pellet 1.3beta2* [Sirin et al., 2006] e o (d4) documento de modelo ontológico da aplicação como parâmetros de entrada. A máquina de inferência *Pellet* foi utilizada por fornecer mecanismos de programação para a obtenção desses tempos intermediários.

Essa configuração do serviço de inferência foi executada 10 vezes, sendo cada execução seguida de uma limpeza de memória para não haver interferências nos resultados. A Tabela 7.2 apresenta o tempo médio (em milisegundos) de cada tarefa relacionada ao processo de inferência sobre a (d4) ontologia da aplicação *WebMemex*.

Tabela 7.2: Tempo médio de cada sub-processo de inferência sob a ontologia da aplicação *WebMemex* (em ms). Adaptado de Bulcão Neto & Pimentel [2006a].

Ontologia	TCA	TCM	TVD	TC	TF	TAI
WebMemex	1993.4	198.3	609.9	121.7	26.7	13.4

Assim como descrito no experimento com as ontologias do modelo *SeCoM* no Capítulo 5, Seção 5.4.2, o tempo de carregamento do arquivo de ontologia (TCA) tem grande participação no tempo total do processo de inferência sobre a ontologia da aplicação *WebMemex*, algo em torno de 72%.

Sendo, em média, 20% do tempo total de inferência, é considerado baixo o tempo de carregamento do modelo RDF (TCM na Tabela 7.2) que representa a ontologia da aplicação *WebMemex* na memória da máquina de inferência. É importante observar que, na verdade, são também carregadas as definições das ontologias *Actor*, *Relationship*, *Document* e *Time* do modelo *SeCoM*.

O tempo de validação do tipo de linguagem OWL (TVD) da ontologia da aplicação *WebMemex* consome uma alta porcentagem de tempo de inferência, 60% em média, o que significa que esse tipo de serviço deve ser desabilitado quando utilizada a máquina *Pellet* para inferência.

O tempo de classificação (TC) da ontologia da aplicação *WebMemex* foi, em média, 12% do tempo total de inferência. O tempo de fusão das ontologias (TF) que compõem a ontologia da aplicação *WebMemex*, em média 2,6%, ratificou a viabilidade da abordagem ontológica em duas camadas para modelagem de informação contextual.

Por fim, o tempo de associação de instâncias às respectivas classes (TAI) foi baixo, em torno de 1%, por não haver muitas instâncias declaradas nessa ontologia sendo, a maioria delas, provenientes da ontologia *Time*.

Aplicação *WebMemex* com diferentes máquinas de inferência

O segundo experimento realizado teve o objetivo de comparar tempos de resposta de diferentes máquinas de inferência para uso pela aplicação *WebMemex* [Bulcão Neto & Pimentel, 2006a]. O presente trabalho considera esta avaliação como um cenário real, uma vez que os requisitos de inferência da aplicação podem demandar a dedução de fatos que a máquina de inferência *TransitiveReasoner*, escolhida na (a3) atividade de desenvolvimento, pode não ser capaz de atender. Assim, o validador considerou como válida a investigação de como o serviço de inferência da infra-estrutura *SCK* se

comportaria com diferentes máquinas de inferência operando sobre uma mesma base de informações de contexto.

Para efeitos de comparação de tempos de resposta, foram testadas três máquinas de inferência e cinco bases de informações de contexto da aplicação em questão. As máquinas de inferência testadas foram a *TransitiveReasoner*, a *Pellet 1.3beta2* e a *OWL*, que assim como a primeira, é uma máquina de inferência integrada ao subsistema de inferência do arcabouço Jena 2.3 [Carroll et al., 2004]. As bases de informações de contexto diferenciavam entre si quanto ao tamanho total de informações de contexto armazenadas, em torno de 1K triplas RDF. Ou seja, enquanto a primeira base testada continha 1000 triplas RDF de informações de contexto, a segunda continha 2000 triplas, e assim sucessivamente.

Para cada máquina de inferência testada, o serviço de inferência da infra-estrutura SCK obtém, como parâmetros de entrada, a respectiva configuração da máquina de inferência e uma base de informações de contexto. A fim de obter melhor desempenho, a máquina de inferência *OWL* foi configurada como *OWL Micro*, que é uma configuração que a habilita para manipular todos os construtores da linguagem *OWL Lite* e alguns construtores de *OWL DL* com certas restrições [Reynolds, 2006]. Para prevenir falta de memória principal no experimento, o tamanho da memória destinada à máquina virtual Java foi configurada para 64MiB. Cada configuração [base de informações de contexto & máquina de inferência] foi executada 10 vezes, cada execução seguida de uma limpeza de memória como prevenção contra interferências nos resultados.

Figura 7.4: Múltiplas configurações do serviço de inferência de contexto da infra-estrutura SCK sobre diferentes bases de informações de contexto da aplicação *WebMemex*. Adaptado de Bulcão Neto & Pimentel [2006a].

A Figura 7.4 apresenta o tempo de resposta médio (em milissegundos) de cada configuração [base de informações de contexto & máquina de inferência] a qual o serviço de inferência da infra-estrutura SCK foi submetido [Bulcão Neto & Pimentel, 2006a]. Dado que as máquinas de inferência *TransitiveReasoner* e *OWL Micro* não fornecem mecanismos para obter os tempos intermediários de um processo de inferência, como a máquina de inferência *Pellet* o faz, foram obtidas apenas as médias dos tempos de resposta totais para cada máquina de inferência.

Apesar de ser menos expressiva que as máquinas de inferência *Pellet* e *OWL Micro*, é possível perceber que a *TransitiveReasoner* supera em desempenho essas duas máquinas de inferência em todos os casos. Entretanto, para explorar a abordagem de uma máquina de inferência para relações transitivas, a aplicação *WebMemex* sempre armazena os dois sentidos de uma relação simétrica: se um usuário X é amigo de um usuário Y, então a aplicação armazena a informação de que o usuário Y é amigo do usuário X, sem utilizar qualquer processo de inferência. A viabilidade desta abordagem é sustentada à medida que a aplicação *WebMemex* mantém seu requisito de inferir a partir de propriedades relacionadas à rede social de usuários.

Considerando a Figura 7.4, a máquina de inferência *OWL Micro* foi, em geral, 40% mais lenta que a *TransitiveReasoner*, e 15% mais lenta que a *Pellet*. É importante observar também que *Pellet* executou, em média, 25% mais rápida que a *TransitiveReasoner*, quando o serviço de validação de tipo de OWL estava desabilitado.

Embora seja possível que desenvolvedores modifiquem a lista de construtores que uma máquina de inferência para a linguagem OWL deve processar, a máquina de inferência *OWL Micro* não é recomendada para ontologias do mundo real — desenvolvedores Jena sugerem que essa máquina de inferência seja utilizada apenas para investigações experimentais [Reynolds, 2006].

A Figura 7.4 também descreve que o tempo de resposta de inferência baseada em ontologias foi aproximadamente proporcional ao tamanho das bases de informações de contexto da aplicação para cada máquina de inferência testada.

7.3 Considerações finais

Este capítulo apresentou um exemplo de instanciação de cada atividade do processo *POCAp* a partir do modelo de informação contextual *SeCoM* e da infra-estrutura de serviços SCK, ambos utilizados como artefatos auxiliares. Para tornar mais clara a utilização do processo *POCAp*, foi descrita a utilização desse processo na extensão da aplicação *WebMemex*, que utiliza informação contextual para permitir que usuários recomendem páginas Web uns aos outros.

Considerando a (a4) atividade de verificação e validação do processo *POCAp*, no contexto da aplicação *WebMemex*, este trabalho mostrou que mesmo uma máquina de

inferência com recursos mais simples, como a *TransitiveReasoner*, é capaz de inferir a partir de ontologias com alto grau de expressividade. Esta é uma importante decisão de projeto a ser considerada por desenvolvedores.

Segundo, dada a quantidade de tempo gasta por um processo de inferência baseada em ontologias, este trabalho reforça a necessidade de um módulo independente para inferência sobre informação contextual, tal qual o serviço de inferência da infra-estrutura SCK.

Terceiro, este trabalho também constatou que, devido aos diferentes tipos de serviços fornecidos por máquinas de inferência, como validação de tipo de linguagem de ontologia (tempo TVD na Tabela 7.2), não é possível compará-las de forma qualitativa. Máquinas de inferência podem utilizar estratégias de otimização para apoiar diferentes graus de expressividade quanto a uma linguagem de ontologia, ou mesmo algoritmos de classificação customizados para minimizar a quantidade de consultas à árvore de classificação de uma ontologia, tal qual faz a máquina de inferência *Pellet 1.3beta2*.

Por fim, com a experiência adquirida no desenvolvimento da infra-estrutura SCK e da aplicação *WebMemex*, o autor teve um curso aprovado sobre a teoria e a prática de desenvolvimento de aplicações que utilizam padrões de Web Semântica, como RDF e OWL [Bulcão Neto, Prazeres & Pimentel, 2006].

Trabalhos Relacionados

Considerando o histórico da computação sensível a contexto, é possível identificar que as primeiras aplicações sensíveis a contexto, como a *Active Badge* [Want et al., 1992] e a *ParcTab* [Schilit et al., 1993], implementavam mecanismos proprietários de captura, modelagem, armazenamento, interpretação e recuperação de informações de contexto. Seguindo a mesma linha do tempo, aplicações pioneiras seguiam abordagens *ad hoc* de desenvolvimento, sem a devida preocupação com aspectos de definição de processos de software para computação sensível a contexto.

Com a evolução de tecnologias e pesquisas de apoio à computação sensível a contexto, é possível identificar uma tendência ao desenvolvimento de serviços especializados em ambientes distribuídos de larga escala [Román et al., 2002; Grimm et al., 2004; Henricksen & Indulska, 2006]. A partir dessa tendência, surgem os arcabouços de software, os *toolkits*, as infra-estruturas de software e os *middlewares*.

Um arcabouço de software é uma estrutura de apoio definida a partir da qual um projeto de software pode ser organizado e desenvolvido. Arcabouços podem incluir programas de apoio, bibliotecas de código, uma linguagem de *scripts*, ou mesmo outros softwares que auxiliam a desenvolver e unificar diferentes componentes de um projeto de software. Um *toolkit* é desenvolvido com base em um arcabouço de software e oferece um conjunto de componentes reusáveis para funcionalidades comuns. Uma infra-estrutura corresponde a um conjunto de tecnologias bem definidas, confiáveis e publicamente acessíveis que atuam como base para outros sistemas. Um *middleware* é um software que conecta duas ou mais aplicações, geralmente distribuídas, para que estas possam trocar dados sem que se atenham a detalhes de protocolos de comunicação, gerenciamento de dados, balanceamento de carga, dentre outros.

Este capítulo apresenta arcabouços, *toolkits*, infra-estruturas e *middlewares* para a construção de sistemas sensíveis a contexto. Para cada trabalho, são descritos seus respectivos componentes responsáveis pela captura, modelagem, armazenamento, interpretação e recuperação de informações de contexto. É também identificada para cada trabalho a existência de métodos, ferramentas e processos diretamente relacionadas à engenharia de software sensível a contexto.

Em seguida, é realizada uma análise comparativa entre os trabalhos elencados e a abordagem proposta pelo autor representada pela infra-estrutura *SCK*, pelo modelo *SeCoM* e pelo processo *POCap*. A partir dessa análise comparativa são relacionados os pontos positivos e as limitações da abordagem proposta neste trabalho.

8.1 Context Toolkit

Context Toolkit é uma implementação do arcabouço conceitual descrito por Dey et al. [2001] e consiste em abstrações e serviços genéricos para suprir a carência de suporte ao desenvolvimento de aplicações sensíveis a contexto baseadas em sensores [Dey et al., 1999a; Salber et al., 1999]. A arquitetura do *Context Toolkit*, ilustrada na Figura 8.1, fornece as seguintes abstrações ou componentes: widgets, agregadores e interpretadores de informação de contexto.

Figura 8.1: Interações típicas entre aplicações e componentes da arquitetura do *Context Toolkit*. Aplicações podem obter informações de contexto de sensores diretamente via widgets, ou após processamento realizado por agregadores ou interpretadores. Adaptado de Dey et al. [2001].

Quando widgets, agregadores e interpretadores de informação de contexto são instanciados, estes registram-se com o serviço de descoberta. Quando uma aplicação é iniciada, esta contacta o serviço de descoberta para localizar os componentes necessários a sua funcionalidade. Por exemplo, uma aplicação de detecção de

presença de pessoas em um ambiente físico acionaria o serviço de descoberto para encontrar os widgets responsáveis por fornecerem informações de localização.

Cada widget é responsável pela captura de um tipo de informação de contexto via sensores físicos — por exemplo, informação de localização — e pela transmissão dessa mesma informação a agregadores ou aplicações. Informações de contexto capturadas são também armazenadas por widgets de modo que possam ser acessadas por aplicações via mecanismos de *polling* ou *callback*.

A comunicação entre aplicações e componentes do *Context Toolkit* dá-se na forma de mensagens XML transportadas pelo protocolo HTTP. Informações de contexto são representadas segundo um modelo de sintaxe XML, tal como ilustra o Exemplo 8.1, no qual é descrita uma mensagem que uma aplicação envia a um widget específico (linha 2) para a obtenção (linhas 1 e 3) da temperatura média armazenada (linhas 5 e 6) de um determinado ambiente físico (linha 10).

```
0  <!--                                Exemplo 8.1                                -->
1  <retrieveData>
2    <id> identificador do widget consultado </id>
3    <retrieval>
4      <attributeFunctions>
5        <attributeFunction attributeType = temperature,
6                               function = avg>
7      </attributeFunction>
8    </attributeFunctions>
9    <conditions>
10     <condition>location, =, "sala 3009"</condition>
11  </conditions>
12 </retrieval>
13 </retrieveData>
```

Os agregadores são similares aos widgets quanto à função de coletar, armazenar e permitir a consulta de informações de contexto. Entretanto, agregadores e widgets diferem quanto ao escopo. Um widget representa todas as informações de contexto de um determinado tipo de sensor físico, enquanto que um agregador agrupa e associa informações de contexto a uma entidade específica do mundo real. Por exemplo, um agregador pode agrupar e relacionar informações de localização e de identificação a uma pessoa ou a um objeto.

Interpretadores são componentes responsáveis por abstrair informações de contexto de baixo nível — como aquelas obtidas diretamente de sensores físicos — em informações de alto nível. Essa capacidade de abstração é realizada normalmente por meio da combinação de um ou mais tipos de informação de contexto para produzir uma única informação. Por exemplo, um interpretador pode converter coordenadas GPS para identificar uma rua em particular. Interpretações mais complexas incluem reunir informações de identificação, localização e agenda de uma pessoa para deduzir que uma reunião está ocorrendo.

Várias aplicações foram implementadas para validar a arquitetura oferecida pelo *Context Toolkit*, dentre elas a *Conference Assistant*, descrita no Capítulo 2, Seção 2.5, e a *Family Intercom*, que visa facilitar a comunicação entre moradores de uma residência instrumentada com diversos tipos de sensores [Nagel et al., 2001]. Foram implementados, por exemplo, widgets específicos para o reconhecimento das vozes dos moradores e para a detecção de presença e de localização desses. Quando a aplicação é notificada da presença e respectiva localização de um morador na casa, é habilitado um canal de áudio que possibilita que esse morador se comunique com outros moradores na mesma residência.

Apesar da diversidade de aplicações desenvolvidas a partir do *Context Toolkit* [Dey et al., 2001; Dey, 2000], para o conhecimento do autor não há nenhum trabalho que reporte algum tipo de avaliação, quer seja voltada para usuários finais, quer seja voltada para desenvolvedores dessas aplicações.

Apesar disso, a partir da experiência adquirida com o *Context Toolkit* no desenvolvimento de aplicações, Dey et al. [2001] propuseram um processo de software simplificado que inclui três fases distintas: especificação, aquisição e ação, conforme descrito a seguir.

Especificação: tanto da solução quanto do problema a ser tratado;

1. Especificação dos comportamentos de sensibilidade a contexto a serem implementados na aplicação;
2. Determinação das informações de contexto necessárias para os comportamentos especificados;

Aquisição: determinação e posterior instalação do hardware ou de sensores disponíveis para fornecer as informações de contexto requeridas;

Ação: escolha e execução de comportamentos sensíveis a contexto.

8.2 ConFab

ConFab é uma infra-estrutura de software que visa reduzir o esforço no desenvolvimento de aplicações sensíveis a contexto por meio de um conjunto de serviços básicos e de uma linguagem para descrição de informações de contexto [Hong & Landay, 2001; 2004]. A arquitetura do *ConFab*, ilustrada na Figura 8.2, mostra a interação entre aplicações e seus serviços básicos: o serviço de gerenciamento de sensores, o serviço de criação automática de caminho, o serviço de eventos e o serviço de consulta.

O serviço de gerenciamento de sensores registra e gerencia todos os sensores físicos locais utilizados pelos demais serviços da infra-estrutura. Além disso, fornece um formato uniforme de dados de sensores para ser utilizado pelo serviço de consulta

Figura 8.2: Arquitetura da infra-estrutura de software *ConFab*: sensores físicos e serviços básicos são distribuídos entre aplicações e a infra-estrutura. Adaptado de Hong & Landay [2001].

e pelo serviço de eventos. As informações de contexto capturadas de sensores são armazenadas em estruturas, codificadas no formato XML, chamadas espaços de informação [Heer et al., 2003b]. A unidade básica de armazenamento de um espaço de informação é uma tupla de contexto, que representa informação contextual de entidades, tais como uma pessoa, um objeto, ou uma localização.

O Exemplo 8.2 ilustra um trecho XML de uma tupla de contexto para uma entidade do tipo localização (linha 2) de uma entidade do tipo pessoa (linhas 4 e 5). O modelo subjacente a uma tupla de contexto inclui metadados que descrevem a entidade em questão (linhas 1 a 3), relacionamentos a outras entidades (linhas 4 e 5), informação temporal de armazenamento (linha 6), o valor associado à entidade descrita (linha 8) e a fonte geradora da informação de contexto atual (linhas 11 a 14).

```

0  <!--                               Exemplo 8.2                               -->
1  <ContextTuple dataformat="edu.school.building"
2      datatype="location"
3      description="location of an entity"
4      entity-link="http://myhost.com/~jdoe"
5      entity-name="John Doe"
6      timestamp-created="2003.Feb.13 16:06 PST">

```

```
7   <Values>
8     <Value value="523" />
9   </Values>
10  <Sources>
11    <Source datatype="location"
12      link="http://localhost/map.jsp"
13      source="Location Simulator"
14      timestamp="2003.Feb.13 16:06 PST" value="523" />
15  </Sources>
16 </ContextTuple>
```

O serviço de criação automática de caminho combina o fluxo de dados de sensores e os componentes de software necessários para refinar, unir e transformar dados provenientes de sensores para informações de contexto de alto nível. Por exemplo, se existem três serviços que fornecem informação de localização em um ambiente, o serviço de criação automática de caminho deve monitorar as informações vindas desses serviços para identificar a ocorrência do evento desejado.

O serviço de eventos permite que aplicações registrem que eventos lhes interessam de um ambiente físico via mecanismo de *callback*. Em vez de as aplicações obterem os dados necessários diretamente de sensores físicos, o serviço de eventos notifica as aplicações de forma assíncrona no instante em que o evento desejado ocorre.

O serviço de consulta, por sua vez, permite que aplicações consultem informações de contexto de forma síncrona via mecanismo de *polling*. Internamente, esse serviço realiza o processamento distribuído das consultas emitidas por aplicações e as redireciona segundo mudanças de contexto em um ambiente físico [Heer et al., 2003b].

Para representar informações de contexto, a infra-estrutura *ConFab* utiliza a linguagem de especificação de informações de contexto (*Context Specification Language - CSL*). Essa linguagem, cuja sintaxe baseia-se no padrão XML, permite que aplicações declarem consultas a informações de contexto, bem como eventos na comunicação com o serviço de consulta e com o serviço de eventos, respectivamente [Hong & Landay, 2001; Hong, 2002].

Como ilustra a Figura 8.2, os serviços que compõem a infra-estrutura *ConFab* podem ser executados localmente em um dispositivo computacional, ou podem servir remotamente a aplicações locais de dispositivos. Entre respectivas instâncias local e remota do serviço de consulta ou do serviço de eventos a comunicação ocorre por meio da linguagem CSL sobre o protocolo HTTP. O mesmo se aplica à comunicação entre aplicações locais de dispositivos e instâncias remotas de ambos os serviços.

Um exemplo de aplicação que possibilita a validação da infra-estrutura *ConFab* é a *Co-occupant Awareness* [Heer et al., 2003a], apresentada no Capítulo 2, Seção 2.5. Para incentivar a comunicação inter-pessoal, essa aplicação permite o compartilhamento de endereços eletrônicos e páginas Web de pessoas que se

encontram em um mesmo ambiente físico. Hong & Landay [2004] também desenvolveram um comunicador instantâneo que explora a localização de seus usuários chamado *Lemming* e uma aplicação para resposta a situações de emergência chamada *Enhanced 911*. *Lemming* permite que seus usuários compartilhem (e protejam) suas respectivas informações de localização, que são atualizadas à medida que esses usuários movem-se em um ambiente. *Enhanced 911* permite que usuários disponibilizem suas respectivas informações de localização quando fazem chamadas de emergência via telefones celulares, como em incêndios ou seqüestros.

A partir da utilização dessas aplicações, Hong [2005] realiza uma avaliação com usuários para obter suas percepções de privacidade quanto a disponibilizar suas respectivas localizações. Além de afirmarem que as aplicações são intuitivas quanto ao uso, todos os usuários avaliados declaram que o compartilhamento de informações de localização foi de sua própria iniciativa (independentemente da informação estar correta ou não) e que o controle de disponibilizar esse tipo de informação é útil para situações em que privacidade é desejada.

Apesar do desenvolvimento de várias aplicações com base na infra-estrutura *ConFab*, seus autores não se têm preocupado em propor métodos, técnicas ou processos que norteiem a engenharia de software sensível a contexto.

8.3 Gaia

Gaia é uma infra-estrutura de software que fornece serviços de gerenciamento de mobilidade de usuários entre espaços ativos [Román et al., 2002]. Um espaço ativo é uma abstração pró-ativa que engloba informações de contexto, tarefas e dispositivos associados a cada usuário de um ambiente físico. A pró-atividade de um espaço ativo advém de sua necessidade de reconfiguração quando seus componentes mudam, por exemplo, a localização de um usuário.

Segundo a arquitetura ilustrada na Figura 8.3, a infra-estrutura *Gaia* disponibiliza cinco serviços básicos para o gerenciamento de espaços ativos: o serviço de gerência de eventos, o serviço de contexto, o serviço de presença de entidades, o serviço de repositório de espaços e o sistema de arquivos de contexto. Além desses serviços, a infra-estrutura *Gaia* também fornece um arcabouço para a construção, execução e adaptação de aplicações em espaços ativos [Hess et al., 2002].

O serviço de gerência de eventos é responsável pela distribuição de eventos em espaços ativos por meio da implementação de um modelo de comunicação baseado em componentes fornecedores e consumidores e canais de comunicação. Todos os componentes da infra-estrutura *Gaia* utilizam o serviço de gerência de eventos para saberem as mudanças do estado de espaços ativos, bem como para reagirem segundo essas mudanças, por exemplo, a entrada e saída de pessoas de uma sala.

Figura 8.3: Componentes da infra-estrutura *Gaia*. Adaptado de Román et al. [2002].

O serviço de contexto permite que aplicações consultem e registrem informações de contexto para que possam se adaptar ao espaço ativo em que operam. Há componentes especializados tanto para a captura e o envio de informações ao serviço de contexto quanto para a interpretação dessas informações referentes ao estado de espaços ativos. Esse serviço utiliza um modelo de representação de informação contextual baseado em lógica de primeira ordem e álgebra booleana.

Toda informação contextual é descrita por meio de um predicado quaternário na forma `CONTEXT(<CONTEXTTYPE>,<SUBJECT>,<RELATER>,<OBJECT>)`. O termo `<CONTEXTTYPE>` refere-se ao tipo de informação contextual que está sendo descrita; `<SUBJECT>` é a pessoa, local ou objeto ao qual o contexto está relacionado (sujeito); `<OBJECT>` é o valor associado ao sujeito em questão; por fim, `<RELATER>` é uma relação entre o sujeito e o objeto declarados, que pode ser um operador de comparação (`>`, `<`, `=`), um verbo ou uma preposição. É também possível criar informações de contexto mais complexas ao utilizar regras com operações de lógica de primeira ordem, tais como conjunção, disjunção e negação de predicados de contexto.

O Exemplo 8.3 ilustra exemplos de informações de contexto descritas segundo o modelo subjacente à infra-estrutura *Gaia* [Ranganathan & Campbell, 2003], tais como a localização física de uma entidade (linha 1), a temperatura de um espaço físico (linha 2), a relação social entre duas pessoas (linha 3), o status da fila de impressão de uma impressora (linha 4) e a data e hora locais (linha 5). Por fim, as linhas 6 a 8 descrevem uma informação de contexto na forma de uma regra com o operador de conjunção: SE há mais de quatro pessoas em uma determinada sala E uma aplicação para apresentações eletrônicas está sendo utilizada ENTÃO está ocorrendo uma apresentação na referida sala.

O serviço de presença de entidades é responsável por detectar e atualizar informações de entidades físicas e digitais presentes em um espaço ativo. Esse serviço monitora entidades físicas (pessoas e dispositivos) via sensores físicos, obtendo suas respectivas localizações em um espaço ativo. Entidades digitais (serviços e aplicações)

são monitoradas pelo serviço de presença de entidades e enviam-lhe sinais periódicos para notificar sua presença em um espaço ativo.

```

0  <!--                               Exemplo 8.3                               -->
1  Context(location, chris, entering, room 3231)
2  Context(temperature, room 3231, is, 98 F)
3  Context(social relationship, venus, sister, serena)
4  Context(printer status, srgalw1 printer queue, is, empty)
5  Context(time, , Is, 12:00 01/01/01).
6  Context(Number of people, Room 2401, >, 4)
7      AND Context(Application, Powerpoint, is, Running)
8      => Context(Social Activity, Room 2401, Is, Presentation)

```

O serviço de repositório de espaços é um banco de dados centralizado com informação sobre todas as entidades ativas de um espaço ativo. Essa informação é atualizada ao monitorar canais de comunicação que informam eventos sobre novas entidades e sobre aquelas não mais ativas. Entidades e respectivas propriedades são associadas a representações XML, que podem ser consultadas por aplicações.

O sistema de arquivos de contexto utiliza informação de contexto para fornecer armazenamento pessoal automático a aplicações segundo a presença de usuários em um ambiente físico de interação. Esse sistema cria um espaço de nomes para cada espaço ativo para que informações sejam acessadas por aplicações de forma semelhante a um sistema de arquivos tradicional [Hess & Campbell, 2003].

A infra-estrutura *Gaia* tem papel similar ao de sistemas operacionais tradicionais, pois seus serviços criam e gerenciam o estado de espaços ativos. Suas aplicações são baseadas em componentes distribuídos com apoio à execução e ao gerenciamento de componentes remotos fornecidos pelo núcleo de gerenciamento de componentes.

Foram desenvolvidas várias aplicações para validar a infra-estrutura *Gaia*, dentre elas a *Music Player*, que executa arquivos de música segundo as preferências de um usuário, bem como sua mobilidade entre espaços ativos [Román & Campbell, 2003], e a *ConChat*, uma aplicação de bate-papo em que usuários comunicam-se mediante a troca de informações de contexto, que incluem a localização, a identidade e a atividade realizada por esses usuários.

Para o conhecimento do autor deste trabalho, não há trabalhos referentes à infra-estrutura *Gaia* quanto à avaliação de características da mesma e a técnicas de Engenharia de Software para a construção de software sensível a contexto.

8.4 one.world

one.world é uma arquitetura de software que fornece um arcabouço integrado de serviços com vistas principalmente a aplicações que se adaptam automaticamente a ambientes computacionais dinâmicos [Grimm et al., 2004]. A Figura 8.4 ilustra os

componentes da arquitetura *one.world* que, segundo Grimm [2004], facilitam a tarefa de gerenciar mudanças constantes em ambientes de computação ubíqua: os serviços de base, os serviços de sistema e o espaço do usuário.

Figura 8.4: Visão geral da arquitetura *one.world*. Os serviços de base e os serviços de sistema constituem o núcleo da arquitetura *one.world*, enquanto que aplicações, bibliotecas e utilitários de sistema executam no espaço do usuário. Adaptado de Grimm et al. [2004].

No espaço de usuário, a arquitetura *one.world* fornece bibliotecas que incluem funções para construção de interfaces com o usuário e para a execução de controladores de eventos.

Os serviços de base lidam com os requisitos de mudança, composição *ad hoc* e compartilhamento em ambientes de computação ubíqua. A máquina virtual garante a composição entre aplicações e dispositivos, uma vez que não se pode prever todos os dispositivos que serão utilizados em um ambiente.

Tuplas definem um modelo de dados comum, na forma de pares nome e valor, para facilitar o compartilhamento de informações entre aplicações. O Exemplo 8.4 ilustra a classe abstrata *Tuple* e seus respectivos métodos de acesso aos campos de uma tupla. Por meio dessa classe o modelo de dados da arquitetura *one.world* é disponibilizado para aplicações [Grimm et al., 2004].

Ambientes são o mecanismo central para estruturação e composição de aplicações, servindo como repositórios para armazenamento de tuplas, componentes da aplicação e outros ambientes de forma aninhada. Eventos assíncronos são utilizados para comunicação local ou remota e visam notificar aplicações para se adaptarem a mudanças no ambiente físico ou em dispositivos.


```
0  <!--                               Exemplo 8.4                               -->
1  public abstract class Tuple {
2      public Guid id; // ID de uma tupla
3      public DynamicTuple metaData; // metadados de uma tupla
4
5      // acesso programático a campos de uma tupla
6      public final Object get(String name); { ... }
7      public final void set(String name, Object value); { ... }
8      public final Object remove(String name); { ... }
9      public final List fields(); { ... }
10     public final Class getType(String name); { ... }
11 }
```

Os serviços de sistema podem ser usados como blocos de construção de aplicações. A máquina de consulta disponibiliza um mecanismo de consulta por tuplas via instanciação de filtros. O serviço de eventos remotos é o mecanismo básico de comunicação por meio da rede, a partir do qual são enviados eventos para outros serviços remotos. O serviço de checkpointing captura o estado de execução de um ambiente e salva-o como uma tupla, possibilitando reverter posteriormente o estado de execução do ambiente. Esse serviço facilita a tarefa de reativar uma aplicação após ela ter sido desativada, ou após uma falha em um dispositivo, por exemplo.

O serviço de armazenamento de tuplas permite às aplicações acessarem as tuplas armazenadas em cada ambiente. Para Grimm et al. [2004], os mecanismos de consulta e de armazenamento de tuplas simplificam a entrada e saída de dados porque as aplicações podem acessar diretamente itens de dados relevantes. O serviço de descoberta localiza serviços por meio de suas respectivas descrições. O serviço de migração permite mover ou copiar um ambiente e seu conteúdo — tuplas armazenadas, componentes da aplicação e ambientes aninhados — tanto localmente quanto para um outro dispositivo. Essa abordagem é útil para aplicações que seguem uma pessoa de dispositivo para dispositivo conforme ela se move no mundo físico, por exemplo.

Após desenvolverem a arquitetura *one.world*, seus criadores realizaram uma avaliação com base em quatro critérios específicos: se a arquitetura proposta é suficientemente poderosa e extensível para construir aplicações com diferentes necessidades (completude), o esforço envolvido na construção de aplicações (complexidade), o impacto no desempenho de aplicações que mudam constantemente (desempenho), e se desenvolvedores, que não sejam os criadores da arquitetura *one.world*, conseguem construir aplicações a partir dessa arquitetura (utilidade).

Para obter respostas a esses critérios, foram desenvolvidos serviços, utilitários e aplicações com base na arquitetura *one.world* [Grimm, 2004]. Dentre estes, destacam-se a aplicação *Emcee*, que permite a usuários mover ou copiar aplicações e respectivos dados entre diferentes dispositivos, e a aplicação *Labscape*, que permite que dados de experimentos sigam pesquisadores conforme eles se movem ao longo

de um laboratório. Essa aplicação utiliza sensores para capturar e atualizar a informação de localização, que por sua vez é utilizada para direcionar os dados de experimentos para o serviço mais próximo do pesquisador.

Quanto aos critérios supracitados, Grimm et al. [2004] concluem que: (a) é possível construir novos serviços para atender a demandas específicas; (b) a produtividade de uma equipe para desenvolver aplicações adaptáveis não é significativamente menor em relação a desenvolver aplicações convencionais; (c) os serviços de migração e de descoberta de recursos não têm grande impacto no desempenho de aplicações; e (d) desenvolvedores podem focar na construção de suas aplicações sem se preocuparem com detalhes dos serviços necessários.

Para o conhecimento do autor deste trabalho, os trabalhos referentes à arquitetura *one.world* não mencionam técnicas, métodos, procedimentos ou processos para sistematizar a construção de software sensível a contexto.

8.5 Cooltown

Cooltown é um projeto cuja finalidade é apoiar atividades que envolvam a mobilidade de usuários por meio de um conceito denominado “presença na Web” [Kindberg et al., 2002]. Com base nesse conceito, entidades do mundo físico — objetos, locais e pessoas — são interligados a serviços disponibilizados na Web. O princípio do projeto *Cooltown* consiste, pois, em utilizar a infra-estrutura da Web atual para apoiar cenários de computação ubíqua e, conseqüentemente, identificar serviços da Web que precisam ser estendidos para atingir o propósito do projeto. A arquitetura em camadas da infra-estrutura de serviços do projeto *Cooltown* é apresentada na Figura 8.5.

Figura 8.5: Infra-estrutura de presença na Web do projeto *Cooltown*. Adaptado de Kindberg et al. [2002].

Na camada inferior da infra-estrutura estão os mecanismos para a obtenção dos pontos de presença na Web para pessoas, locais e objetos. Esses mecanismos incluem serviços de descoberta e de captura de URLs e de identificadores.

Assumindo que uma pessoa pode utilizar um dispositivo móvel, como um PDA, este pode ser munido de sensores que emitem identificadores para esse dispositivo. Ao mesmo tempo, uma pessoa pode interagir com um objeto, como uma pintura em quadro, que tem um marcador que, quando lido, transmite o identificador para esse objeto. Ambos os identificadores citados são convertidos para URLs pelo serviço de resolução de ID que servirão como referências na Web para os objetos em questão. Em combinação com mecanismos automáticos tradicionais de descoberta de serviços, o serviço de descoberta de rede também permite que uma pessoa controle que serviços deseja cadastrar e localizar.

Em relação à Figura 8.5, a camada intermediária abriga recursos denominados presenças na Web, especialmente projetados para capturar as respectivas funções e relacionamentos entre objetos, pessoas e locais.

Para tornarem-se presentes na Web, objetos, como impressoras e câmeras digitais, são estendidos com servidores Web, ou disponibilizam sua identificação em um servidor Web específico. Pessoas tornam-se presentes na Web ao fornecerem páginas Web pessoais a um serviço de ligação, para facilitar a comunicação entre indivíduos, e também ao fornecerem informações por meio de um serviço gerenciador de locais específico. Locais tornam-se presentes na Web ao organizarem os objetos que contêm em coleções sob o gerenciamento de um serviço gerenciador de locais.

O serviço gerenciador de locais é um servidor Web que cria, gerencia e disponibiliza para acesso uma organização contextual e física de uma localidade, que é representada como coleções de ligações Web para as respectivas presenças Web de pessoas, objetos e outras localidades. Todas as presenças Web gerenciadas por um serviço gerenciador de locais específico podem ser localizadas por meio de seus respectivos identificadores, ou por meio do tipo de serviço que oferecem. O serviço gerenciador de locais é capaz de disponibilizar informações sobre cada recurso de um local nos formatos HTML, para navegação por um usuário, e XML, para acesso programático.

Na camada superior da infra-estrutura *Cooltown* estão os serviços para intercâmbio de conteúdo e de URLs sobre HTTP entre entidades presentes na Web, bem como aqueles serviços específicos para usuários móveis de acordo com seu contexto, que pode incluir parâmetros como, por exemplo, a localização de um usuário.

Squirt é um protocolo baseado em XML de transferência de URLs que permite que um dispositivo presente na Web com recursos computacionais restritos, tal como um PDA, envie a URL do conteúdo em questão para um outro dispositivo, por exemplo, uma impressora. A principal vantagem do protocolo *Squirt* é a independência de dispositivo, dado que o que muda é apenas o processamento da URL pelo destinatário.

Como complemento às facilidades adquiridas do protocolo *Squirt*, são também fornecidos mecanismos para a transferência de conteúdo entre dispositivos, bem como para controle de dispositivos presentes na Web. Para ambos os casos, é utilizado o modelo padrão de preenchimento de formulários da Web, cujos dados são enviados via operação POST do protocolo HTTP. As únicas diferenças em relação à prática convencional da Web estão no transporte, que pode utilizar protocolos específicos de redes sem fio, e no servidor, que pode ser executado localmente em um dispositivo em vez de remotamente em algum ponto na Internet. Vários cenários de uso do protocolo *Squirt* e da técnica de intercâmbio de conteúdo via formulários Web são apresentados por Kindberg & Barton [2001]; Kindberg et al. [2002].

Para validar a infra-estrutura proposta, o projeto *Cooltown* tem implementado várias aplicações que exploram mobilidade. A aplicação *WebBus* fornece informações sobre pontos turísticos segundo a localização de um ônibus para turismo [Kindberg et al., 2002]. A localização do ônibus é obtida por meio de um transmissor de coordenadas GPS que lhe é acoplado, enquanto que as informações sobre os pontos turísticos, disponibilizadas como páginas HTML, são transmitidas aos PDAs dos passageiros via uma rede sem fio instalada no veículo.

O sistema *Rememberer* permite que visitantes de museus registrem e acessem posteriormente suas experiências pessoais por meio de dispositivos portáteis e redes sem fio [Fleck et al., 2002]. Para capturar a localização de cada visitante, é utilizada identificação por rádio-frequência via cartões ou relógios de pulso. Câmeras fotográficas digitais registram as interações dos usuários no museu, como a visita uma escultura ou pintura, e as disponibiliza em forma de páginas Web exibindo a ordem em que esses locais foram visitados.

Apesar da utilização da infra-estrutura do projeto *Cooltown* em várias aplicações, para o conhecimento do presente autor não há estudos de avaliação de aspectos da mesma, bem como de técnicas voltadas para a engenharia de software de computação ubíqua em geral, ou de software sensível a contexto em particular.

8.6 QSI

A infra-estrutura de software *QSI* (*Queensland's software infrastructure*)¹ oferece um conjunto de serviços que realizam o processamento e o gerenciamento de informações de contexto coletadas a partir de múltiplas fontes. A arquitetura em camadas da infra-estrutura *QSI* é ilustrada na Figura 8.6, que inclui: a camada de coleta de contexto, camada de recepção de contexto, camada de gerenciamento de contexto, camada de consulta, camada de adaptação e camada de aplicação.

¹Como os criadores dessa infra-estrutura não lhe têm associado um nome em nenhuma de suas publicações, o autor deste trabalho nomeou-a quanto à universidade em que tem sido desenvolvida.

Figura 8.6: Arquitetura em camadas da infra-estrutura QSI. Comunicação síncrona é representada por uma seta contínua, enquanto que comunicação assíncrona, por uma seta tracejada. Adaptado de Henriksen & Indulska [2006].

A camada de coleta de contexto adquire informação de contexto de sensores via componentes a receptores e processa essa informação utilizando técnicas de interpretação e fusão de dados realizadas, respectivamente, por componentes interpretadores e agregadores. Dessa maneira, é possível utilizar dados de sensores tanto em seu estado natural, obtido diretamente de sensores físicos, bem como em um nível maior de abstração para ser disponibilizado para o sistema gerenciador de contexto. Os componentes da camada de coleta de contexto seguem a mesma abordagem dos widgets, interpretadores e agregadores do *Context Toolkit*.

A camada de recepção de contexto fornece um mapeamento bidirecional entre as camadas de coleta e de gerenciamento de contexto. Esse mapeamento é obtido por meio de traduções de dados heterogêneos coletados de sensores em uma representação baseada em fatos manipulada pela camada de gerenciamento. Além do mecanismo de tradução mencionado, o mapeamento entre as camadas de coleta e de gerenciamento de contexto se dá por meio do roteamento de consultas da camada de gerenciamento aos correspondentes componentes da camada de coleta.

A camada de gerenciamento de contexto tem o papel de um repositório de informações de contexto que atende múltiplas aplicações. Sua representação de

informação de contexto é baseada na linguagem CML (*Context Modeling Language*) [Henricksen & Indulska, 2004b], cujos construtores permitem o tratamento de diferentes aspectos de informações de contexto (consideradas fatos), tais como dependência, histórico, classificação (estática, dinâmica, etc.) e qualidade (precisão, atualização, grau de certeza, etc.) [Henricksen et al., 2002; Henricksen & Indulska, 2004a]. A Figura 8.7 ilustra um modelo CML de uma aplicação que permite que usuários escolham os canais de comunicação de apoio a suas interações. A legenda associada descreve características de informações de contexto segundo a notação gráfica da linguagem CML.

Figura 8.7: Instância de modelo de contexto na notação da linguagem CML. A relação *located near* representa uma informação de contexto derivada, enquanto que a relação *engaged in* tem relação de dependência com a relação *located at*. Adaptado de Henricksen & Indulska [2004b].

A camada de consulta da infra-estrutura *QSI* fornece uma interface por meio da qual aplicações e a camada de adaptação consultam informações do sistema gerenciador de contexto. Três tipos de consulta são permitidos por essa camada: consultas lógicas, com graus de variação entre falso e verdadeiro; consultas baseadas em associações entre entidades, atributos e respectivos valores; consultas de situação, uma combinação de fatos ligados por operadores lógicos semelhante a regras; e consultas baseadas em eventos, que ao invés de receber uma resposta imediata, gera notificações assíncronas quando os eventos em questão ocorrem.

A camada de adaptação da infra-estrutura *QSI* gerencia repositórios comuns de situações, de eventos e de preferências de usuários, bem como avalia tais repositórios em prol das aplicações que usam os serviços da camada de consulta. Um único repositório é compartilhado entre um agrupamento lógico de aplicações, onde um grupo compreende tipicamente todas as aplicações que residem em um dispositivo, ou todas aquelas que pertençam a um mesmo usuário.

Para finalizar a descrição das camadas da infra-estrutura *QSI*, a camada de aplicação fornece um *toolkit* de programação para desenvolvedores explorarem os serviços da infra-estrutura em suas aplicações sensíveis a contexto. O *toolkit* oferece um conjunto de classes que encapsulam serviços de camadas inferiores, como a classe *Context*, que representa um modelo de informação de contexto tal como utilizado pela camada de gerenciamento de contexto, bem como a classe *Triggering*, que fornece métodos para criação e ativação dinâmica de gatilhos associados a eventos.

Para validar sua proposta, Henricksen & Indulska [2006] desenvolveram a aplicação *Mercury*, em que usuários definem suas preferências quanto aos canais de comunicação que podem dar apoio a interações em cenários de interação por texto assíncrono (correio eletrônico), por texto síncrono (comunicador instantâneo) e por áudio e vídeo (conferência multimídia). Para essa escolha, podem ser utilizados parâmetros como o tipo de diálogo (anúncios, correio eletrônico e discussões), o número de pares envolvidos, prioridade (para cenários que necessitem canais síncronos) e preferências de usuários (por exemplo, uso de correio eletrônico em vez de telefone).

Henricksen & Indulska [2006] realizaram uma avaliação comparativa entre o estudo de caso da aplicação *Mercury* e implementações alternativas sem o uso da infra-estrutura de software *QSI*. Foram avaliadas três características de qualidade, que incluem complexidade de codificação, capacidade de manutenção e capacidade de reúso. De acordo com sua avaliação, Henricksen & Indulska [2006] concluíram que, com o uso da infra-estrutura *QSI*, é exigida mínima codificação para processar, gerenciar e consultar informações de contexto, bem como para modificar ou remover novos fatos. A avaliação também descreve a facilidade de reúso de abstrações relacionadas a contexto (fatos, situações, preferências e eventos) e de componentes de processamento de informações de contexto (agregadores e interpretadores).

A partir das experiências com o desenvolvimento de aplicações baseadas nas facilidades oferecidas pela infra-estrutura *QSI*, Henricksen & Indulska [2006] definiram um processo de software para a construção de aplicações sensíveis a contexto. A Figura 8.8 ilustra esse processo, que envolve as fases de análise (A), projeto (P), implementação (I), customização da infra-estrutura (C) e teste (T). Tarefas de uma mesma fase são numeradas para identificar a ordem em que são executadas.

Figura 8.8: Um processo para a construção de software sensível a contexto. Os artefatos utilizados em cada passo são exibidos entre parênteses. Os passos A3, T2 e T3 podem solicitar reiterações para o passo A2. Adaptado de Henricksen & Indulska [2006].

A fase de análise inicia com a documentação dos requisitos da aplicação (passo A1). Em seguida, há dois passos específicos para aplicações sensíveis a contexto: o passo A2, que busca identificar os tipos de informação de contexto (fatos e situações) a partir do documento definido no passo A1; e o passo A3, que refina requisitos levantados no passo A1 que são dependentes de contexto, tais como a definição de eventos e de preferências de usuários.

Após a fase de análise, dois conjuntos de tarefas podem ser executadas em paralelo: uma que envolve projeto (P) e implementação (I), e outras que abrange a customização (C) da infra-estrutura de software.

Tarefas de projeto e de implementação exploram as APIs de consulta, ramificação e eventos para incorporar comportamento sensível a contexto às aplicações. Por outro lado, esses passos adotam metodologias e linguagens de programação tradicionais.

A fase de customização ocorre em situações que demandam novos tipos de fatos ou situações. Novos tipos de fatos coletados de sensores pode demandar a implementação de novos receptores, agregadores e interpretadores para as camadas de

coleta e de recepção de contexto (passo C1). Para o teste da aplicação, devem ser geradas amostras de informações de contexto, de gatilhos associados a eventos e de preferências de usuários: tudo será utilizado pelos gerenciadores de contexto e de adaptação (passo C2). Antes da implantação da aplicação, deve-se mapear (passo C3) as interfaces de customização com o modelo de informação de contexto e os conjuntos de preferências e de gatilhos definidos nos passos A2, A3, C2 e T2.

Na fase de teste de unidade (T1), são aplicados métodos de teste de caixa branca com base no arcabouço de teste *JUnit*². Na fase de teste de sistema (T2), são realizados testes de caixa preta por meio de amostras de conjuntos de informações de contexto, de preferências de usuários e de gatilhos associados a eventos. O estágio final da fase de teste (T3) avalia a aplicação usando um ambiente real de hardware, com dados reais de sensores e usuários finais.

8.7 CoBrA

Para tratar questões relacionadas ao controle de privacidade de usuários, bem como à modelagem, à interpretação e ao compartilhamento de informações de contexto, Chen et al. [2004b] propõem o *middleware CoBrA (Context Broker Architecture)*. *CoBrA* é um *middleware* baseado em agentes para dar apoio a agentes sensíveis a contexto em espaços inteligentes, por exemplo, salas de reunião.

O elemento central da arquitetura *CoBrA* é o intermediador de contexto, um agente inteligente cujas responsabilidades incluem: (a) o fornecimento de um modelo centralizado de informações de contexto que possa ser compartilhado entre dispositivos, serviços e agentes de um mesmo espaço inteligente; (b) a obtenção de informações de contexto de sensores físicos ou servidores de informação; (c) a inferência de informações de contexto que não podem ser adquiridas diretamente de sensores; (d) a detecção e resolução de inconsistências no modelo de informações de contexto armazenado; (e) proteger a privacidade de usuários por meio de políticas que controlam o uso e o compartilhamento de suas informações. A Figura 8.9 apresenta a arquitetura *CoBrA* e as interações entre cada um de seus elementos.

O intermediador de contexto é composto dos seguintes elementos: módulo de aquisição de contexto, base de conhecimento de contexto, máquina de inferência de contexto e módulo gerenciador de contexto.

O módulo de aquisição de contexto contém uma biblioteca de procedimentos que abstraem a complexidade de obtenção de informações de contexto de baixo nível, por exemplo, quando adquiridas de sensores físicos. Chen et al. [2004b] descrevem um cenário no qual procedimentos dessa biblioteca obtêm endereços de enlace quando detectam a presença de dispositivos que se comunicam via *Bluetooth*.

²Disponível na Internet em <http://junit.sourceforge.net>.

Figura 8.9: Arquitetura do *middleware CoBra*. O intermediador de contexto adquire informações de contexto de diversas fontes e as combina em um modelo compartilhado entre entidades computacionais de um mesmo espaço físico. Adaptado de Chen et al. [2004b].

Todo o conhecimento de um espaço inteligente é armazenado na base de conhecimento de contexto na forma de triplas RDF. Informações de contexto armazenadas nesta base são instâncias de ontologias definidas a partir da ontologia de nível superior *SOUPA* (*Standard Ontology for Ubiquitous and Pervasive Applications*) [Chen et al., 2004c]. Codificada na linguagem OWL, a ontologia *SOUPA* inclui vocabulários modulares para representar agentes segundo as dimensões de espaço, tempo, eventos, ações, perfis de usuários, crenças, desejos, intenções, e políticas de privacidade.

A Figura 8.10 ilustra a arquitetura da ontologia *SOUPA* composta de dois conjuntos de documentos de ontologias: o núcleo *SOUPA*, que define um vocabulário genérico para diferentes aplicações de computação ubíqua; e as extensões *SOUPA*, que definem vocabulários para aplicações específicas a partir do núcleo *SOUPA*.

A máquina de inferência de contexto define como deduzir novas informações de contexto a partir de informações obtidas das fontes de informação de contexto. Dois tipos de inferência são permitidos: inferência baseada na semântica da ontologia *SOUPA* e inferência baseada em regras do domínio da aplicação.

Com base na semântica da ontologia *SOUPA*, é possível inferir ordenação temporal de eventos, a localização de pessoas e dispositivos, relações sociais entre pessoas,

Figura 8.10: Arquitetura da ontologia de nível superior SOUPA. Ontologias específicas podem ser construídas a partir das ontologias que compõem o núcleo SOUPA, cada qual em um diferente espaço de nomes XML. Adaptado de Chen et al. [2004c].

relações espaciais e mereológicas entre regiões geográficas, e características de dispositivos computacionais.

Quando inferência baseada em regras é requisitada, o intermediador de contexto busca todas as informações de contexto da aplicação em questão e as converte para triplas RDF. Estas são enviadas para a máquina de inferência de contexto, que as processa por meio de encadeamento progressivo (não-configurável), e retorna ao intermediador de contexto os novos fatos deduzidos também no formato de triplas RDF.

O Exemplo 8.5 descreve uma regra (linhas 1 e 2), que declara que uma pessoa deseja fazer uma apresentação em uma dada reunião caso ocorram os seguintes fatos: (a) o sistema deve reconhecer que a pessoa está em uma sala onde a reunião está agendada; (b) a pessoa tem o papel de apresentador na reunião; e (c) não existe nenhuma evidência de que a pessoa não esteja na referida sala. Com base nos fatos descritos nas linhas 4 a 6, a máquina de inferência de contexto deduz que “*Harry deseja fazer uma apresentação na reunião m1203 que ocorre na sala rm338*”.

```

0  <!--                               Exemplo 8.5                               -->
1  locatedIn(Person,Room), meeting(Meeting,Room), speakerOf(Person,Meeting),
2      not(notLocatedIn(Person,Room)) --> intends(Person,give_prst(Meeting))
3
4  locatedIn(harry,rm338)
5  meeting(m1203,rm338)
6  speakerOf(harry,m1203)

```

Antes do intermediador de contexto compartilhar informações de um usuário com outro agente, ele consulta o módulo gerenciador de contexto para determinar se o agente em questão tem permissão para receber essas informações. O intermediador compartilha tais informações apenas se a política de privacidade definida pelo usuário assim permitir. Por meio da ontologia *SOUPA* de políticas de privacidade, usuários podem definir regras que concedem ou negam acesso a suas informações privadas. Para processar essas regras, é utilizado um algoritmo de inferência de políticas que explora inferência baseada em regras.

Para validar a arquitetura do *middleware CoBrA*, Chen et al. [2004a] desenvolveram o sistema *EasyMeeting*, cuja meta é dar apoio a atividades típicas de usuários em um ambiente de reuniões. Serviços sensíveis a contexto auxiliam palestrantes e espectadores durante uma apresentação de uma reunião, por exemplo, para controlar uma apresentação por meio de comandos de voz (“próximo”, “anterior”, “pára” e “inicia”), bem como para controlar a iluminação da sala de reuniões.

A partir de um experimento realizado com usuários e o sistema *EasyMeeting*, Chen et al. [2004a] relatam que a principal preocupação dos usuários envolvidos diz respeito a questões de segurança e privacidade. No entanto, nenhuma publicação referente ao *middleware CoBrA*, de conhecimento do autor deste trabalho, descreve um processo de avaliação em que se considere características desse *middleware*, como desempenho, reusabilidade, complexidade de codificação de aplicações, expressividade da ontologia *SOUPA*, entre outros.

Analogamente, nenhum dos artigos publicados sobre o *middleware CoBrA* descreve métodos, metodologias, procedimentos ou processos que facilitem a engenharia de software sensível a contexto.

8.8 SOCAM

O tratamento de cenários em que aplicações e serviços adaptam-se de maneira dinâmica é a principal proposta do *SOCAM* (*Service-Oriented Context-Aware Middleware*) [Gu et al., 2004; 2005]. *SOCAM* é um *middleware* distribuído e sensível a contexto cuja arquitetura é composta de serviços independentes que realizam as tarefas de descoberta, aquisição e inferência de informações de contexto. A Figura 8.11 descreve a arquitetura do *middleware SOCAM* que, segundo Gu et al. [2004], permite a rápida prototipação de aplicações sensíveis a contexto.

A camada de coleta de contexto diz respeito às fontes de informação de contexto, que incluem sensores físicos e virtuais. Sensores virtuais são representados por serviços Web e servidores de informação, como um serviço de previsão meteorológica.

A camada de aplicação sensível a contexto faz menção a serviços e aplicações que utilizam diferentes níveis de informação de contexto: no formato capturado de

Figura 8.11: Arquitetura do *middleware* SOCAM orientado a serviços sensíveis a contexto. Setas mais espessas indicam fluxo de dados; caso contrário, indicam fluxo de controle. Adaptado de Gu et al. [2005].

sensores físicos ou na forma processada pelo serviço interpretador de contexto. Os componentes dessa camada adaptam seu comportamento segundo o seu contexto corrente especificado na forma de regras.

A principal camada da arquitetura do SOCAM é a camada de *middleware* de contexto, que engloba o serviço de descoberta, a base de informações de contexto, os fornecedores de contexto e o interpretador de contexto.

O serviço de descoberta fornece um mecanismo que permite que os serviços fornecedor de contexto e interpretador de contexto possam anunciar sua presença. Se uma aplicação requisita, por exemplo, a localização de uma dada pessoa, essa aplicação deve enviar uma mensagem de consulta ao serviço de descoberta. Este carregará as ontologias da base de informações de contexto e as respectivas instâncias anunciadas por diferentes fornecedores de contexto. Em seguida, identificará qual fornecedor de contexto é responsável pela informação de localização requisitada, e enviará à aplicação solicitante uma referência para acesso ao serviço fornecedor de contexto em questão.

A base de informações de contexto armazena ontologias de informações de contexto, bem como históricos de informação de contexto de sub-domínios em particular. Cada domínio tem uma base lógica para armazenamento de informações de contexto.

Os fornecedores de contexto abstraem informações de contexto obtidas de fontes heterogêneas, externas ou internas, e converte tais informações para representações codificadas nas linguagens RDF e OWL. Isso é feito para que os demais serviços

componentes do *middleware* possam não apenas compartilhar informações de contexto entre si, mas também reusá-las. Fornecedores externos são aqueles que obtêm informação de contexto de sensores virtuais, por exemplo, como um servidor que fornece a localização *outdoor* de pessoas. Fornecedores internos, por sua vez, são aqueles responsáveis por tratar informações de contexto obtidas diretamente de sensores físicos, por exemplo, sensores de localização baseados em rádio-frequência.

O interpretador de contexto fornece mecanismos de raciocínio lógico para processar informações de contexto, que inclui as tarefas de derivação de informações de alto nível a partir de informações capturadas de sensores, a consulta e manutenção da consistência da base de conhecimento de contexto e a resolução de conflitos envolvendo informações de contexto.

O interpretador de contexto inclui dois componentes básicos: uma máquina de inferência de contexto e uma base de conhecimento de contexto. Os tipos de inferência aceitos pelo interpretador incluem inferência baseada em ontologias e em regras de lógica de primeira ordem. A inferência baseada em regras pode utilizar encadeamento progressivo e regressivo. A base de conhecimento em questão contém uma ontologia de informações de contexto para um dado sub-domínio e suas respectivas instâncias. O Exemplo 8.6 ilustra uma instância RDF e uma regra, ambas associadas a uma ontologia para o domínio de residências. As linhas 1 a 4 descrevem que uma pessoa encontra-se deitada em um quarto. As linhas 6 a 8 descrevem as condições necessárias para deduzir que uma pessoa encontra-se dormindo em um quarto: ela deveria estar deitado no quarto em questão com a porta fechada e intensidade de luz baixa.

```

0  <!--                               Exemplo 8.6                               -->
1  <socam:Person rdf:about="John">
2    <socam:hasPosture rdf:resource="#LieDown"/>
3    <socam:locatedIn rdf:resource="#Bedroom"/>
4  </socam:Person>
5
6  (?user rdf:type socam:Person), (?user socam:locatedIn socam:Bedroom),
7  (?user socam:hasPosture 'LieDown'), (socam:Bedroom socam:LightLevel 'LOW'),
8  (socam:Bedroom socam:DoorStatus 'CLOSED') --> (?user socam:status 'SLEEPING')

```

Informações de contexto gerenciadas pelo *middleware* SOCAM são instanciadas a partir de ontologias OWL de domínios específicos em computação sensível a contexto que, por sua vez, são especializações da ontologia de nível superior CONON (*Context Ontology*). A ontologia CONON define os conceitos de pessoa, localização, entidade computacional e atividade, e consiste de 14 classes e 06 classes [Wang et al., 2004b], conforme mostra a Figura 8.12.

A classe ContextEntity é a super-classe da ontologia CONON, sendo que existe uma instância sua para cada serviço ou usuário, e essa mesma instância apresenta um conjunto de subclasses Person, Location, CompEntity e Activity. Detalhes desses

Figura 8.12: Hierarquia de classes da ontologia de nível superior CONON. Adaptado de Gu et al. [2004].

conceitos básicos que formam a ontologia CONON são definidos nas ontologias de domínio específico, tal como a ontologia para o domínio de residências apresentada por Wang et al. [2004b]; Gu et al. [2005].

Para validar os serviços oferecidos pelo *middleware* SOCAM, foi desenvolvida uma aplicação para residências cujo cenário envolve o *playback* de músicas e o ajuste da iluminação da sala de jantar no momento em que a família preparam-se para fazer sua refeição [Gu et al., 2004]. Outra aplicação desenvolvida foi a *SituAwarePhone*, que adapta o estado de telefones celulares segundo a dinâmica de situações corriqueiras [Wang et al., 2004a]. Por exemplo, se ao receber uma chamada telefônica essa aplicação inferir — via regras — que o usuário está em uma reunião, então o modo de resposta do telefone pode ser configurado automaticamente para modo de vibração, envio de SMS, ou mensagem de voz na caixa de mensagens.

Quanto à avaliação do *middleware* SOCAM, vários de seus trabalhos publicados relatam avaliação de desempenho do serviço interpretador de contexto [Wang et al., 2004b; Gu et al., 2005] e do serviço de descoberta [Gu et al., 2005].

Com base no tempo de inferência total obtido em seu experimento, Wang et al. [2004b] concluem que o mecanismo de inferência de informações de contexto não é adequado para aplicações com requisitos críticos de tempo. Além disso, concluem que o mecanismo de inferência baseado em regras depende de três fatores, que incluem o tamanho do repositório de informações de contexto, a complexidade das regras e a velocidade do processador do computador em que a inferência ocorre. Ambas afirmações foram ratificadas em experimentos relatados por Gu et al. [2003; 2005].

Gu et al. [2005] também concluem que inferência baseada em ontologias consome mais tempo de processamento que inferência baseada em regras. Isto se deve principalmente à quantidade de informação que deve ser processada por ambos tipos de inferência, em geral menor para inferência baseada em regras. Outra constatação refere-se ao comportamento linear do tempo total de inferência baseada em ontologias em relação ao número de classes e instâncias envolvidas.

Quanto à avaliação do serviço de descoberta, Gu et al. [2003] relatam um experimento que mede a quantidade de solicitações concorrentes que esse serviço consegue atender. Foi constatado que o tempo médio de atendimento de cada solicitação é aproximadamente proporcional ao número de solicitações concorrentes.

Para o conhecimento do autor deste trabalho, nenhum dos artigos publicados sobre o *middleware* SOCAM, ou sobre a *ontologia* CONON, propõe métodos, metodologias ou processos no que tange a engenharia de software sensível a contexto.

8.9 Comparação com o trabalho proposto

Esta seção apresenta uma análise comparativa entre os trabalhos supracitados e abordagem proposta representada pela tríade SCK-SeCoM-POCAp.

Inicialmente, a infra-estrutura SCK é comparada sob a perspectiva dos requisitos propostos por Dey [2000], apresentados no Capítulo 2, Seção 2.4.

Em seguida, são analisados os aspectos de formalidade e expressividade de cada abordagem de modelagem de informação contextual, com principal foco nos trabalhos que utilizam ontologias como técnica de representação de conhecimento, tal como o modelo SeCoM.

Por fim, são comparados ao processo POCAp aqueles trabalhos que relatam processos de software sensível a contexto, considerando como parâmetros a definição do escopo das fases envolvidas, a linguagem de modelagem de processos, e a definição dos artefatos de entrada e saída.

8.9.1 Comparação com a infra-estrutura SCK

A Tabela 8.1 ilustra a comparação entre a infra-estrutura SCK e trabalhos relacionados com respeito a requisitos para construção de software sensível a contexto [Dey, 2000; Dey et al., 2001]. Para efeitos de simplificação, é utilizada uma notação que relaciona cada requisito a um identificador Rn , onde:

R1 Especificação de informações de contexto

R2 Separar aquisição de utilização de informações de contexto

R3 Interpretação de informações de contexto

R4 Comunicação distribuída e transparente

R5 Disponibilidade contínua de componentes de captura de informações de contexto

R6 Armazenamento de informações de contexto

R7 Descoberta de recursos

Tabela 8.1: Comparação entre a infra-estrutura SCK e trabalhos relacionados.

	R1	R2	R3	R4	R5	R6	R7
<i>Context Toolkit</i>	XML	<i>widgets</i>	interpretador	XML sobre HTTP	S	BD	S
<i>ConFab</i>	CSL	serviço de gerenciamento de sensors	criação automática de caminho	XML sobre HTTP	S	BD	N
<i>Gaia</i>	lógica	N	serviço de contexto	CORBA	S	N	S
<i>one.world</i>	tuplas	máquina virtual	tuplas	protocolo proprietário	S	arq	S
<i>SCK</i>	OWL	tradutores de contexto	serviço de inferência de contexto	N	N	arq BD	N
<i>Cooltown</i>	HTML	captura de URL e de ID	serviço gerenciador de locais	XML sobre HTTP	S	XML	S
<i>QSI</i>	CML	receptor	gerenciador de contexto	I	S	BD	N
<i>CoBrA</i>	OWL	módulo de aquisição de contexto	máquina de inferência de contexto	RDF/XML sobre HTTP	S	BD	S
<i>SOCAM</i>	OWL	fornecedor de contexto	interpretador de contexto	RDF/XML sobre HTTP	S	BD	S

Vale ressaltar que aqueles requisitos não contemplados por um determinado trabalho apresentam o valor *N* associado. Por exemplo, a infra-estrutura *Gaia* não separa aquisição e utilização de informações de contexto (*R2*).

Os requisitos *R5* e *R7* descrevem apenas se são considerados (S) ou não (N) por cada um dos trabalhos elencados. Quando não é possível determinar se um requisito

é tratado por um trabalho, é assumido que a informação está indisponível (I), caso do requisito *R4* para a infra-estrutura *QSI*.

No caso do requisito *R6*, os trabalhos cujo armazenamento é feito em base de dados relacional recebem o valor *BD*; os que armazenam em uma base de dados XML, o valor *XML*; e aqueles que armazenam em arquivos têm o valor *arq* associado.

Considerando o requisito *R1*, a infra-estrutura *SCK* tem a vantagem de ser independente de modelo ontológico de informação contextual. Dessa forma, é possível utilizar as ontologias *SOUPA* ou *CONON* em conjunto com a infra-estrutura *SCK*, com mínimo (ou nenhum) esforço de implementação.

Os tradutores de contexto da infra-estrutura *SCK* são os componentes que atendem ao requisito *R2*. Esses elementos convertem informações de contexto obtidas de sensores, serviços Web ou aplicações para triplas RDF segundo a semântica do modelo ontológico subjacente.

Com respeito ao requisito *R3*, a vantagem da infra-estrutura *SCK* é a capacidade de configuração de seus serviços. Embora *Gaia*, *CoBrA* e *SOCAM* também ofereçam mecanismos de inferência sobre ontologias e/ou regras, o serviço de inferência de contexto oferece a desenvolvedores a possibilidade de configurar ambos mecanismos de inferência. Desenvolvedores podem escolher máquinas de inferência com graus de expressividade diferentes (para ontologias RDFS e OWL), bem como o tipo de encadeamento de regras (progressivo, retrógrado e misto) que melhor atende aos diversos tipos de demanda de aplicações sensíveis a contexto.

Os requisitos *R4* e *R5* apontam duas linhas de investigação que não têm sido bastante exploradas pela infra-estrutura *SCK*. Embora ofereça a aplicações um formato padrão de intercâmbio de informações de contexto, no caso triplas RDF/XML, a infra-estrutura *SCK* está implementada na forma de uma API local, sem quaisquer funções que explorem comunicação remota e disponibilidade de serviços.

Em relação ao requisito *R6*, confere vantagem sobre os demais trabalhos analisados a configurabilidade do serviço de armazenamento de informações de contexto da infra-estrutura *SCK*.

Quanto ao requisito *R7*, a infra-estrutura *SCK* ainda não oferece um serviço de descoberta, embora o descreva em sua arquitetura. Isto se deve a mudanças de decisão ao longo da execução do presente trabalho. O serviço de descoberta deve ser ponto de investigação para futuras extensões da infra-estrutura *SCK*.

8.9.2 Comparação com o modelo SeCoM

A partir das informações apresentadas na Tabela 8.1, a especificação de informações de contexto (*R1*) é tratada em cada trabalho sob diferentes perspectivas: orientada a tuplas (*one.world*), orientada a sintaxe (*Cooltown*, *Context Toolkit*, *ConFab*), orientada a modelos (*QSI*) orientada a semântica (*Gaia*, *SCK*, *CoBrA* e *SOCAM*).

Na abordagem orientada a tuplas da arquitetura *one.world* informações de contexto estão representadas e são acessadas via classes Java, o que dificulta o compartilhamento e o reúso de informações de contexto entre aplicações. Isto tem efeito direto sobre a interoperabilidade entre aplicações sensíveis a contexto, uma característica importante em cenários de computação ubíqua. Baseado em ontologias, o modelo *SeCoM* permite o compartilhamento e a interoperabilidade semântica de informação contextual entre aplicações, bem como o reúso dessa informação, por exemplo, para modelar informações de contexto de um domínio específico.

As dificuldades apontadas pela abordagem orientada a tuplas são também compartilhadas pelos trabalhos adeptos da abordagem orientada a sintaxe. No projeto *Cooltown* informações de contexto são representadas no formato HTML, uma linguagem voltada, principalmente, à apresentação de informação, e que carece de formalidade, expressividade e adequação para a interpretação de informações de contexto. Embora *Context Toolkit* e *ConFab* ofereçam interoperabilidade sintática entre aplicações, dada a utilização da linguagem XML em seus respectivos modelos de informação contextual, seus modelos também carecem de formalidade e expressividade para a interpretação de informações de contexto. Dada a natureza ontológica do modelo *SeCoM*, é possível inferir novas informações de contexto a partir da semântica de informações instanciadas do modelo.

Na abordagem orientada a modelos da infra-estrutura *QSI*, o modelo de informação contextual apóia-se sobre a linguagem CML com influência direta de técnicas consagradas de modelagem conceitual, como os modelos de entidades e relacionamentos e a linguagem UML. A linguagem CML oferece maior expressividade e formalidade que as técnicas de modelagem citadas, e tem a vantagem de representar características de informação contextual não consideradas no modelo *SeCoM*, como dependência, classificação e qualidade. Por sua vez, o modelo *SeCoM* oferece o benefício de explorar expressividade e formalidade semântica superior à oferecida pela linguagem CML.

A abordagem orientada a semântica fornece um alto nível de formalidade e expressividade para modelagem de informação contextual. A semântica de informação contextual viabiliza mecanismos de inferência a partir dos quais são geradas novas informações. O modelo baseado em lógica da infra-estrutura *Gaia* não favorece o compartilhamento e o reúso de informação contextual, tal qual fazem os modelos ontológicos *SOUPA*, *CONON* e *SeCoM* das respectivas iniciativas *CoBrA*, *SOCAM* e *SCK*. A Tabela 8.2 descreve uma comparação entre esses modelos de informação contextual cuja semântica baseia-se em ontologias.

Os modelos *SOUPA*, *CONON* e *SeCoM* compartilham a característica de um modelo ontológico em duas camadas, sendo que a camada inferior herda e/ou estende os conceitos definidos na camada superior. No entanto, apenas *CONON* e *SeCoM* foram avaliados quanto ao tempo para a importação (ou fusão) de ontologias (TF).

O modelo *SOUPA* tem a vantagem de descrever uma diversidade maior de informação contextual. Porém nenhuma avaliação de suas ontologias tem sido feita, tal como nos modelos *CONON* e *SeCoM*.

Tabela 8.2: Comparação entre o modelo *SeCoM* e modelos semânticos de informação contextual baseados em ontologias.

Modelo	Informação contextual	Avaliação
<i>SOUPA</i>	perfil de usuário, localização, tempo, dispositivo, evento, ação, crença, desejo, intenção e privacidade	N
<i>SeCoM</i>	perfil de usuário, localização, tempo, dispositivo, evento e atividade	expressividade lógica e desempenho
<i>CONON</i>	usuário, localização, tempo, dispositivo, rede, aplicação, serviço, agente e atividade	desempenho

Ontologias derivadas do modelo *CONON* foram avaliadas quanto ao tempo de resposta total de processos de inferência sobre as mesmas. O modelo *SeCoM*, entretanto, avaliou não apenas o tempo de resposta total, mas também o tempo de cada sub-processo de inferência sobre as ontologias do modelo, bem como sobre a ontologia da aplicação *WebMemex*. A viabilidade desse tipo de avaliação está relacionada à influência que a semântica de ontologias acarreta em um processo de inferência.

Outro ponto favorável ao modelo *SeCoM* advém do fato de ser o único cuja expressividade lógica de suas ontologias é avaliada. Isto é importante para que se saiba a complexidade computacional de processos de inferência ao usar ontologias do modelo.

8.9.3 Comparação com o processo *POCAp*

Dentre os trabalhos apresentados neste capítulo, apenas dois descrevem processos de software que podem ser utilizados para comparação com o processo *POCAp*: o *Context Toolkit* e a infra-estrutura *QSI*. A Tabela 8.3 resume a comparação entre o processo *POCAp* e as iniciativas dos trabalhos mencionados.

O processo definido por Dey et al. [2001] a partir de sua experiência com o *Context Toolkit* é simplificado e segue uma estrutura de fases sequencial. É dada atenção especial a aspectos de baixo nível, como instalação de sensores físicos, que não são cobertos pelos outros dois processos em questão. O processo de Dey et al.

[2001] também não é formalizado quanto à modelagem subjacente de um processo de software, além de não descrever os artefatos produzidos em cada fase, nem as possíveis iterações do processo.

Tabela 8.3: Comparação entre o processo *POCAp* e processos de software sensível a contexto.

Processo	Escopo	Modelagem	Definição artefatos	Definição iterações
<i>Context Toolkit</i>	especificação, aquisição e ação	N	N	N
<i>POCAp</i>	análise, especificação, projeto, desenvolvimento e verificação e validação	linguagem <i>SPEM</i>	S	S
<i>QSI</i>	análise, projeto, customização, implementação e teste	N	S	S

Já o processo proposto por Henricksen & Indulska [2006] delimita um conjunto de fases clássicas de engenharia de software, embora não utilize uma linguagem de modelagem de processos, tal como a linguagem *SPEM* descreve o processo *POCAp*. A ocorrência de iterações e a geração de artefatos no processo de Henricksen & Indulska [2006] podem ser mais bem descritas com a adoção de uma linguagem de modelagem de processos.

A partir da modelagem de cada fase do processo *POCAp*, verifica-se que este também inclui atividades voltadas à customização de uma aplicação, tal como faz o processo de Henricksen & Indulska [2006]. Essas atividades do processo *POCAp* incluem (a1.4) análise e especificação de extensão do modelo, (a2.2) projeto de extensão de serviços e (a2.3) projeto de novos serviços. Por fim, o processo *POCAp* é independente de modelo ontológico de informação contextual e de infra-estrutura de serviços, o que lhe confere uma grande vantagem sobre as outras propostas.

8.10 Considerações finais

Este capítulo apresentou uma análise comparativa entre trabalhos da área de computação sensível a contexto e a abordagem proposta nesta tese na forma da infra-estrutura *SCK*, do modelo *SeCoM* e do processo *POCAp*. Incluindo a proposta desta tese, todos os trabalhos relacionados compartilham do objetivo de apoiar a construção de sistemas sensíveis a contexto.

Quanto à infra-estrutura *SCK*, foi destacada a configurabilidade de seus serviços voltados para o armazenamento, consulta e inferência de informações de contexto. Em relação ao modelo *SeCoM*, ressaltou-se a avaliação da expressividade lógica de suas ontologias, bem como a medição e análise dos tempos intermediários de um processo de inferência baseado em ontologias. Quanto à caracterização do processo *POCAp*, prevalecem o escopo de suas fases, a definição formal dos artefatos e iterações existentes, e sua independência de modelo e de infra-estrutura de software de apoio.

As limitações da abordagem proposta neste trabalho foram também evidenciadas, principalmente quanto a aspectos de comunicação e distribuição de informações de contexto da infra-estrutura *SCK*, tratamento de características de informação contextual e diversidade das dimensões de informação contextual tratadas pelo modelo *SeCoM*, e o refinamento do processo *POCAp* para tratar aplicações sensíveis a contexto em geral, em complemento àquelas baseadas em ontologias em particular.

Conclusão

Este capítulo apresenta um resumo do trabalho realizado e discute trabalhos futuros.

São revisitados os problemas que motivaram a realização deste trabalho, assim como suas contribuições e respectivas limitações. Para finalizar, são apresentadas linhas de investigação para o desenvolvimento de trabalhos futuros.

9.1 Problemas

Este trabalho trata a carência de uma abordagem que considere, em termos de processo de software, a complexidade de desenvolvimento de software sensível a contexto. O trabalho realizado trata esse problema por meio de três linhas de investigação, que são discutidas a seguir: modelagem de informações de contexto, serviços para tratamento de informações de contexto e processo de software para computação sensível a contexto.

9.1.1 Modelagem de informação contextual

A partir de uma investigação de aspectos dos modelos de informação contextual existentes, foram identificadas as seguintes demandas:

- aumento da expressividade e formalidade de modelos de informação contextual;
- uso de técnicas de modelagem que apóiem a expressividade e formalidade do modelo;

- diversificação de tipos de informação contextual;
- e uso de padrões de representação de informação.

A importância da expressividade está na sua relação direta com o grau de representação da semântica dos conceitos gerenciados por um software sensível a contexto. Em outras palavras, quanto maior a expressividade de um modelo de informação contextual, mais informação semântica é possível representar sobre os conceitos de um software sensível a contexto.

Quanto à formalidade de um modelo de informação contextual, quanto mais formal um modelo for, maior é a capacidade do software que o utiliza de realizar inferências a partir de informações de contexto.

Com a adoção de técnicas de modelagem, que permitam a expressividade e formalidade necessárias, de padrões de representação de informação, que promovam o reúso e o compartilhamento de informações de contexto entre aplicações, bem como com a adoção de diversificação de tipos de informação contextual, é possível abranger uma maior variedade de domínios de aplicação sensível a contexto.

9.1.2 Serviços para tratamento de informação contextual

Após investigação acerca da diversidade e complexidade das tarefas que um software sensível a contexto realiza, a literatura propõe a separação de funções entre aplicações e infra-estruturas de software. Essas infra-estruturas devem fornecer a aplicações um conjunto de serviços especializados a aplicações sensíveis a contexto.

Dado que progressos em modelagem de informação contextual demandam alterações na arquitetura de infra-estruturas de serviços, este trabalho explora a demanda por uma infra-estrutura de software cujos serviços devam explorar um modelo de informação contextual expressivo, formal e que utilize padrões de representação de informação.

9.1.3 Processo de software para computação sensível a contexto

Embora a literatura relate um leque variado de soluções sob a forma de modelos e serviços, é reconhecida a relevância de uma investigação quanto a contribuições da Engenharia de Software para o desenvolvimento de software sensível a contexto.

Nesse sentido, foi identificada a demanda por um processo de software que complemente uma solução baseada em modelos e serviços em que, juntos, processo, modelos e serviços tratem a complexidade de desenvolvimento de um software sensível a contexto.

9.2 Contribuições

Após a identificação dos problemas a serem tratados, os esforços realizados neste trabalho foram concentrados em:

- desenvolver um modelo que apóie a semântica de informações de contexto;
- desenvolver uma infra-estrutura de serviços que interprete a semântica do modelo de informação contextual;
- e definir um processo de software para computação sensível a contexto que utilize o modelo e os serviços mencionados.

As seções seguintes sumarizam cada uma das frentes de desenvolvimento deste trabalho: o modelo semântico de informação contextual *SeCoM* (*Semantic Context Model*), a infra-estrutura de serviços *SCK* (*Semantic Context Kernel*) e o processo de software *POCAp* (*Process for Ontological Context-aware Applications*).

9.2.1 Modelo *SeCoM* (Modelo de Contexto Semântico)

Uma das contribuições deste trabalho consiste no modelo *SeCoM* (*Semantic Context Model*), que foi desenvolvido com base em ontologias e em padrões de Web Semântica para a modelagem e respectiva representação de informação contextual, tal como relatado nos trabalhos de Bulcão Neto & Pimentel [2003a; 2004; 2005; 2006a].

O uso de ontologias como técnica de modelagem é justificado pelas suas características de formalidade, semântica explícita e abstração de implementação, que são ortogonais às características do modelo pretendido. A adoção de padrões de Web Semântica para representação de informação contextual busca apoiar o intercâmbio, o reúso e o compartilhamento de informação contextual entre aplicações.

Para atender a uma variedade de aplicações, o modelo *SeCoM* é composto de ontologias inter-relacionadas que descrevem perfis de usuários, localização, tempo, dispositivos computacionais, eventos e atividades. Essas ontologias foram desenvolvidas segundo diferentes metodologias, porém prevalece o enfoque no reúso de definições de metadados e de ontologias existentes na Web.

As ontologias que compõem o modelo *SeCoM* são organizadas segundo uma abordagem em duas camadas, onde a camada superior representa o modelo em si, e uma camada inferior herda e/ou estende os conceitos do modelo para representar o conhecimento particular de uma aplicação sensível a contexto.

Cada ontologia do modelo *SeCoM* foi avaliada quanto a sua expressividade lógica, quanto ao tempo de resposta de um processo de inferência e, mais importante, quanto

ao tempo de cada sub-processo de inferência. O processo de avaliação das ontologias do modelo *SeCoM* foram descritas nos trabalhos de Bulcão Neto & Pimentel [2006a;b].

Como resultado desse processo de avaliação, é possível contribuir com desenvolvedores de aplicações sensíveis a contexto ontológico ao fornecer-lhes informações úteis sobre a complexidade computacional de processos de inferência baseados nas ontologias do modelo *SeCoM*, assim como a influência que a semântica de informação contextual modelada por tais ontologias acarreta em um processo de inferência.

9.2.2 Infra-estrutura *SCK* (Núcleo de Contexto Semântico)

Outra contribuição deste trabalho é a infra-estrutura de software *SCK* (*Semantic Context Kernel*), que fornece a aplicações sensíveis a contexto um conjunto de serviços que processam a semântica explícita de informações de contexto instanciadas a partir de um modelo ontológico baseado em padrões de Web Semântica, como o modelo *SeCoM*.

Considerando as publicações sobre a infra-estrutura *SCK*, Bulcão Neto et al. [2005b] relatam decisões de projeto e aspectos de implementação de sua arquitetura de serviços, e Bulcão Neto et al. [2005a] descrevem como foi utilizado cada serviço para estender uma aplicação de captura, acesso e recomendação de navegação Web chamada *WebMemex*.

Com respeito a outros trabalhos relacionados, destaca-se a infra-estrutura *SCK* em virtude da configurabilidade de seus serviços de armazenamento, consulta e inferência de informações de contexto, bem como do fato da mesma ser independente de modelo ontológico de informação contextual. Ou seja, é possível utilizar outros modelos de informação contextual baseados em ontologias com mínima (ou nenhuma) modificação na infra-estrutura *SCK*.

A partir da avaliação do serviço de inferência de contexto da infra-estrutura *SCK*, foram identificadas questões de projeto que desenvolvedores de software sensível a contexto devem considerar quando utilizam modelos ontológicos de informação contextual, que incluem: a importância de sub-processos de inferência, a viabilidade de um módulo independente para inferência, e a necessidade de armazenamento em memória das ontologias frequentemente utilizadas, conforme descritos nos trabalhos de Bulcão Neto & Pimentel [2006a;b].

9.2.3 Processo *POCAp* (Processo para Aplicações Sensíveis a Contexto Baseadas em Ontologias)

Para complementar a solução deste trabalho em relação à complexidade de desenvolvimento de aplicações sensíveis a contexto, foi definido o processo *POCAp* (*Process for Ontological Context-aware Applications*). *POCAp* é um processo de software de

apoio ao desenvolvimento de aplicações sensíveis a contexto, especialmente aquelas cuja semântica de informação contextual é baseada em ontologias, tal como descrito nos trabalhos de Bulcão Neto et al. [2006a;b].

A origem do processo *POCAp* advém da experiência da extensão da aplicação *WebMemex*, relatada em Bulcão Neto et al. [2005a], por meio de informações de contexto instanciadas do modelo *SeCoM* e de serviços da infra-estrutura *SCK*. O processo *POCAp* é orientado a situações em que aplicações sensíveis a contexto são construídas a partir de uma infra-estrutura de serviços capaz de interpretar a semântica de informações de contexto instanciadas de um modelo baseado em ontologias.

A principal vantagem do processo *POCAp* reside sobre o fato de ser independente tanto de modelo ontológico de informação contextual quanto de infra-estrutura de serviços. Além disso, as fases que compõem o processo abrangem um leque maior de atividades em relação a trabalhos relacionados, além de preocupar-se com a definição dos artefatos e iterações existentes por meio de uma linguagem de modelagem de processos de software.

9.3 Limitações

Após reflexão sobre o trabalho desenvolvido e a análise comparativa com trabalhos relacionados, foram identificadas limitações de cada uma das três linhas de investigação seguidas neste trabalho. Essas mesmas limitações servem de motivação para a realização de esforços futuros com o intuito de aperfeiçoar o trabalho realizado.

9.3.1 Limitações do modelo *SeCoM*

São apresentadas a seguir limitações identificadas com respeito ao modelo de informação contextual *SeCoM*.

- a validação do modelo por uma aplicação que explora um conjunto restrito de ontologias, no caso, as ontologias Actor, Time, Document e Relationship;
- a ausência de representação de características de classificação, dependência e qualidade de informação contextual, que incluem dinamismo, precisão, grau de certeza, entre outros;
- e a abrangência das dimensões de informação contextual tratadas pelo modelo, além de perfis de usuários, localização, tempo, dispositivos computacionais, eventos e atividades.

O fato de poucas ontologias do modelo *SeCoM* terem sido utilizadas pela aplicação *WebMemex* não demonstra a potencialidade do mesmo quanto a sua expressividade e

capacidade de inferência sobre as demais dimensões de informação contextual, como localização, dispositivo computacional, evento e atividade.

Embora a aplicação *WebMemex* utilize a ontologia *Time*, o uso desta ontologia é restrito dado que seu vocabulário apenas registra os instantes temporais de navegação e de recomendação de uma página Web. No caso da aplicação em questão, não há demanda por processos de inferência sobre relações temporais de páginas Web, embora a ontologia *Time* seja habilitada para tal.

Ao longo do desenvolvimento do modelo *SeCoM* não foi realizada uma caracterização de informações de contexto em geral, mas sim um agrupamento de informações na forma de categorias semânticas, tais como localização, tempo e atividade. Ao expressar características de informação contextual — por exemplo, qualidade — é possível assistir processos de inferência quando estes manipulam um mesmo tipo de informação obtida de fontes diferentes. Por exemplo, se em um campus universitário existem mecanismos de GPS, ou aqueles baseados em redes 802.11, para fornecer a localização de indivíduos e objetos, o modelo em questão deve permitir representar que, mesmo para localizações *outdoor*, coordenadas de GPS podem não ser as mais adequadas para fornecer a localização precisa de um indivíduo.

Quanto à terceira limitação do modelo *SeCoM*, embora sua utilização pela aplicação *WebMemex* não tenha demandado a inclusão de outras dimensões de informação contextual, a literatura tem relatado vários sistemas que exploram informações de contexto não tratadas pelo modelo *SeCoM*, tais como aspectos de privacidade [Chen et al., 2004a; Lahlou et al., 2005] e o estado emocional de um usuário [Aizawa et al., 2004; Kim et al., 2005].

9.3.2 Limitações da infra-estrutura *SCK*

Com respeito à infra-estrutura *SCK*, são apresentadas as seguintes limitações:

- o desenvolvimento da infra-estrutura de serviços na forma de uma API;
- a ausência de mecanismos de comunicação remota entre aplicações;
- a ausência de um serviço gerenciador de eventos, um dos requisitos básicos para a construção de software sensível a contexto;
- a validação da infra-estrutura de serviços por uma aplicação que demanda apenas os serviços já desenvolvidos;
- não foi desenvolvido o serviço de descoberta da infra-estrutura *SCK*;
- a avaliação do serviço de inferência de contexto não trata problemas comuns em computação sensível a contexto, como heterogeneidade de conectividade;

- os serviços orientados ao armazenamento e à consulta de informações de contexto não foram avaliados; apenas o serviço de inferência de contexto o foi;
- e processos de inferência baseados em regras não foram objeto de avaliação; apenas processos baseados em ontologias o foram;

O fato da infra-estrutura SCK ter sido implementada como API restringe sua utilização por aplicações remotas. A API em questão tem de ser instalada na máquina servidora em que cada aplicação é executada, servindo assim como um gerenciador particular de informação contextual. Isto inviabiliza o compartilhamento e o reúso de informação contextual entre aplicações sensíveis a contexto. Esta limitação tem influência direta sobre o fato da infra-estrutura não oferecer facilidades de comunicação entre aplicações.

A ausência de um serviço gerenciador de eventos na infra-estrutura SCK limita a capacidade desta de permitir que aplicações sejam notificadas automaticamente dada a ocorrência de eventos que lhes sejam relevantes. Historicamente, esse serviço deveria ser baseado no serviço de eventos da infra-estrutura *Context Kernel* [Arruda Jr. et al., 2003; Jardim et al., 2005a], que contribuiu indiretamente com o trabalho realizado nesta tese. Porém, em função de sua dependência em relação ao modelo XML subjacente, optou-se por não reusá-lo.

Para atender a uma diversidade de aplicações sensíveis a contexto, a infra-estrutura SCK não pode ser avaliada por uma única aplicação que, por sua vez, também demande apenas aqueles serviços oferecidos. A literatura tem apresentado trabalhos que exploram uma grande diversidade de serviços para computação sensível a contexto, como descoberta [Kindberg et al., 2002], migração [Grimm et al., 2004], adaptação [Henricksen & Indulska, 2006], entre outros.

Quanto à avaliação de desempenho do serviço de inferência de contexto, falta-lhe o relacionamento a situações reais de computação sensível a contexto. Por exemplo, investigar como se comporta o serviço em questão quando precisa lidar com inferência sobre a heterogeneidade e abundância de dispositivos, ou sobre a diversidade de tipos de conexão entre esses dispositivos.

Como os serviços de armazenamento e de consulta não foram avaliados, não é possível determinar o impacto sobre esses serviços quando expostos a demandas comuns de computação sensível a contexto, como um grande número de fontes de informação contextual — por exemplo, sensores — requisitando armazenamento de informações, e diversas aplicações consultando essas informações capturadas.

O fato do mecanismo de inferência baseada em regras não ter sido avaliado também não permite quantificar a influência que os tipos de encadeamento e a complexidade dos construtores de regras utilizados acarretam em resposta a demandas usuais de computação sensível a contexto.

9.3.3 Limitações do processo *POCAp*

Após investigação acerca do processo *POCAp*, foram identificadas as seguintes limitações:

- a instanciação do processo foi realizada uma única vez, o que não garante sua completude em relação às diferentes demandas de aplicações sensíveis a contexto;
- a única instanciação do processo foi realizada pelo seu próprio autor, o que não garante que terceiros o aplicarão da forma correta;
- ausência de comparação do processo *POCAp* com processos de software estabelecidos das áreas de Engenharia Web e de Engenharia de Ontologias, dada a possibilidade de que estes tratem aspectos que deveriam ser abordados pelo processo *POCAp*;
- a ausência de fases que lidem com aspectos de baixo nível quanto ao desenvolvimento de aplicações sensíveis a contexto, como a utilização de sensores físicos e a comunicação destes com a infra-estrutura de software corrente;
- e a carência de atenção à estrutura interna dos artefatos produzidos, o que poderia facilitar a aplicação do processo por terceiros.

9.4 Trabalhos futuros

No intuito de tratar limitações da abordagem proposta e de dar continuidade a pesquisas sobre este trabalho, são sugeridos os seguintes trabalhos futuros:

- a extensão das capacidades do modelo *SeCoM*, proposta que pode ser obtida por meio da modelagem de novas dimensões de informação contextual, como aspectos de privacidade, segurança e motivação de usuários (Why). Outro procedimento para estender o modelo *SeCoM* inclui a modelagem de características de informação contextual, como dependência, classificação (estática, dinâmica e derivada) e qualidade (precisão, grau de certeza, taxa de atualização);
- a extensão da infra-estrutura *SCK*, proposta que pode ser feita por meio da adição de dois serviços relevantes para a infra-estrutura: os serviços de descoberta e de gerenciamento de eventos. Outra maneira de estender a infra-estrutura *SCK* inclui adicionar novas capacidades aos serviços desenvolvidos, como o processamento de regras no padrão SWRL pelo serviço de inferência de contexto, assim como a utilização de mecanismos de interpretação

mais sofisticados, como redes neurais [McCulloch & Pitts, 1988], cadeias de Markov [Fayolle et al., 1995] e redes bayesianas [Bernardo & Smith, 2000];

- o acesso remoto aos serviços da infra-estrutura SCK, que pode ser realizada por meio da disponibilização dessa infra-estrutura como serviços Web semânticos descritos por ontologias OWL-S [Martin et al., 2005]. Assim, a abordagem proposta neste trabalho atenderia a aspectos de interoperabilidade semântica não apenas quanto ao acesso a informação contextual, mas também quanto ao acesso aos serviços em questão. O doutorando possui experiência com práticas de serviços Web semânticos a partir de trabalho realizado [Bulcão Neto et al., 2004c] em seu estágio na *Hewlett Packard Laboratories* em Bristol, Inglaterra;
- a avaliação da infra-estrutura SCK, proposta que deve considerar todos os serviços desenvolvidos, por exemplo, quanto ao desempenho no atendimento a requisições de aplicações. O serviço de armazenamento pode ser avaliado considerando a quantidade de triplas RDF a armazenar e o processo de atualização de triplas existentes. O serviço de consulta pode ser avaliado por meio do número de grafos RDF que atendem à consulta e da complexidade dos construtores utilizados em expressões de consulta. O serviço de inferência de contexto pode ser avaliado quanto ao número de regras de contexto utilizadas e a complexidade tanto dos construtores quanto do encadeamento dessas regras;
- a extensão do processo *POCAp*, que pode ser realizada por meio de uma ampla investigação da literatura sobre a maneira pela qual tem-se desenvolvido aplicações sensíveis a contexto. Pode-se também investigar a adequação e eventual extensão para instanciamento do processo *POCAp* em relação à norma ISO 12207 [ISO, 1995]. Para facilitar a utilização do processo *POCAp* por terceiros, seria importante definir modelos que descrevam a estrutura dos artefatos de entrada e saída de cada atividade do processo;
- a validação da abordagem *SeCoM-SCK-POCAp* por terceiros, proposta que pode ser desempenhada por alunos de disciplinas que abordem o tema de computação sensível a contexto. Um experimento pode ser planejado de tal forma que cada participante desenvolva uma aplicação sensível a contexto utilizando a abordagem em questão. Tais aplicações podem gerar a demanda por novos tipos de informações de contexto, novos serviços, bem como refinamentos do processo de software definido.

Ontologias de Apoio do Modelo SeCoM

As seções seguintes descrevem as ontologias de apoio à descrição de perfis de atores do modelo *SeCoM*: Contact, Role, Relationship, Knowledge, Document e Project. Por esse motivo, todas as ontologias citadas importam a ontologia Actor.

Para o desenvolvimento das ontologias de apoio foi utilizada a metodologia *ONIONS* de reuso de ontologias. Quanto à coleta dos termos relevantes de cada ontologia, foram reusados conceitos da ontologia FOAF e dos padrões de metadados *vCard*, *Dublin Core* e *iCalendar*.

As ontologias de apoio foram avaliadas segundo os mesmos critérios das ontologias principais do modelo *SeCoM*: dialeto OWL, expressividade lógica, números de triplas, classes, propriedades e instâncias, e a avaliação de desempenho quanto à inferência baseada em ontologias. Nesse sentido, a configuração de hardware e software do experimento também é o mesmo daquele em que foram avaliadas as ontologias principais do modelo *SeCoM*, tal como descrito no Capítulo 5, Seção 5.4.1.

A.1 Ontologia Contact

A ontologia Contact descreve informações de contato de um ator por meio da relação `hasContactInformation`, que interliga a classe `act:Actor`¹ à classe `ContactInformation`, como ilustra a Figura A.1. Essa classe tem um atributo booleano `isPrefferableContact` que indica qual a informação preferida para contato de um ator.

¹Os caracteres *act:* denotam o espaço de nomes XML associado à ontologia Actor.

Figura A.1: Ilustração da ontologia Contact.

São modelados três tipos de informações de contato na forma de subclasses da classe **ContactInformation**: **OnlineAccount**, **Phone** e **Address**. A classe **OnlineAccount** representa todas as contas *online* que atores podem ter, como contas de aplicações para comunicação instantânea (classe **IMAccount**) e contas de correio eletrônico (classe **EmailAccount**). Indivíduos da classe **OnlineAccount** podem ter um descritor textual (atributo **hasAccountName**) que os identifica.

Indivíduos da classe **IMAccount** possuem os atributos **hasIMType** e **hasIMValue**, que descrevem, respectivamente, o tipo e o valor da referida conta. Por exemplo, uma conta do aplicativo tipo *ICQ* com valor “243516”. De maneira análoga, indivíduos da classe **EmailAccount** possuem os atributos **hasEmailType** e **hasEmailValue**. O atributo **hasEmailType** descreve contas de correio eletrônico pessoais (*HomeEmail*) ou profissionais (*WorkplaceEmail*) na forma de uma lista de literais OWL (*owl:DataRange*).

Informações de contato telefônicas são modeladas pela classe **Phone**, que possui dois atributos **hasPhoneType** e **hasPhoneValue**. Os tipos de contatos telefônicos modelados são também descritos na forma de listas de literais OWL: pessoal (*HomePhone*), profissional (*WorkplacePhone*), móvel (*CellPhone*) e fax (*FaxPhone*).

Informações de contato via endereços são representadas pela classe **Address**, que possui os atributos **hasAddressType** e **hasAddressValue**. Os tipos de contatos por endereço são pessoal (*HomeAddress*) e profissional (*WorkplaceAddress*). O atributo **hasAddressValue** aceita cadeias de caracteres para descrever um endereço.

A aplicação *ConChat* [Ranganathan et al., 2002] é um exemplo de aplicação sensível a contexto que poderia utilizar informações de contato modeladas pela ontologia **Contact**.

A.2 Ontologia Role

A ontologia Role descreve informações sobre o papel social de um ator. Existe uma relação OWL *hasSocialRole*, que interliga as classes *act:Actor* e *Role*, tal como ilustrado na Figura A.2.

Figura A.2: Ilustração da ontologia Role.

São representados dois tipos de papel na ontologia Role: os indivíduos cujo papel é o de aluno (classe *Student*), e aqueles cujo papel é o de empregado (classe *Employee*). Essa classificação tem influências do fato do modelo *SeCoM* ter sido concebido, inicialmente, para o domínio de ensino universitário [Bulcão Neto & Pimentel, 2003a; 2004].

Indivíduos da classe *Student* relacionam-se à classe *act:Organization* por meio da relação *studiesAt*, enquanto que aqueles da classe *Employee* relacionam-se com a mesma classe da ontologia Actor pelas relações inversas entre si *worksFor* e *employs*.

Para finalizar, a classe *StudentGroup* é subclasse de *act:PersonGroup* e representa o grupo de pessoas cujo papel é de estudante.

A aplicação *SmartClassroom* [Shi et al., 2003] é um exemplo de aplicação sensível a contexto que poderia utilizar informações de contato modeladas pela ontologia Role.

A.3 Ontologia Relationship

A ontologia Relationship, descrita na Figura A.3, modela os tipos de relacionamentos sociais existentes entre pessoas. Todos esses relacionamentos são modelados como relações simétricas OWL (*owl:SymmetricProperty*) ligadas à classe *act:Person*.

A principal relação modelada é a relação *knows*, que representa um tipo básico de relacionamento social entre pessoas. Derivadas da relação *knows*, existem as relações

Figura A.3: Ilustração da ontologia Relationship.

isFriendOf, worksWith, isColleagueOf, hasAcquaintanceOf e isEmployerOf. Ou seja, se as pessoas são amigas, trabalham juntas, são colegas, têm algum conhecimento sobre as mesmas e têm relação patrão-empregado, então elas se conhecem.

Existe ainda a relação isCloseFriendOf, derivada da relação isFriendOf. Ou seja, se há relação de amizade próxima entre pessoas, então elas continuam sendo amigas.

A aplicação *WebMemex*, descrita no Capítulo 7, Seção 7.2, fez uso da ontologia Relationship. A aplicação *OntoMail* [Khedr & Karmouch, 2004] é um outro exemplo de aplicação sensível a contexto que poderia utilizar informações de contato modeladas pela ontologia Relationship.

A.4 Ontologia Knowledge

A ontologia Knowledge descreve informações sobre o conhecimento de uma pessoa a respeito de uma área de conhecimento específica. A classe KnowledgeArea descreve áreas do conhecimento humano como subclasses, como ilustra a Figura A.4

Figura A.4: Ilustração da ontologia Knowledge.

Por definição, todo indivíduo da classe *Document* tem, obrigatoriamente, que apresentar um identificador único, dado pelo atributo *hasIdentifier* (*owl:InverseFunctionalProperty*), um título, descrito pelo atributo *hasTitle*, e uma data de criação, indicada pelo atributo *hasCreationDate*. Este último atributo é modelado como sub-propriedade do atributo *time:beginPointOf*², cujo valor assumido é uma entidade cuja extensão temporal é representada por instantes de tempo (*time:InstantThing*). Além de data de criação de um documento, o atributo *hasModificationDate* descreve a data de modificação de um documento.

Alguns tipos de documentos são modelados nesta ontologia, tais como documentos de vídeo, áudio, texto e imagens. Documentos de vídeo são representados pela classe *Video*, enquanto que documentos de áudio são representados pela classe *Audio*, que pode incluir áudio ambiente, efeitos sonoros, músicas, narrações e discursos. Documentos textuais, tais como artigos, correspondências, formulários e manuais, são representados pela classe *Text*. Por fim, documentos de imagens, como fotografias e gráficos, são representados pela classe *Image*.

No caso de documentos da classe *Image*, há relações que interligam qualquer objeto a uma imagem que o descreve, como as relações *depicts* e sua inversa *depiction*.

Existe um conjunto de relações OWL que interligam as classes *Document* e *act:Actor*, todas essas relações sob influência do padrão de metadados *Dublin Core* para bibliotecas digitais.

Dentre essas relações, há aquelas utilizadas para descrever a autoria de um documento, no caso, as relações *creator* e sua inversa *isCreatedBy*. Há relações que descrevem a contribuição de atores para a produção de um documento, neste caso, as relações *contributor* e sua inversa *isContributedBy*. Ainda, existem relações que descrevem o ator responsável pela publicação de um documento: as relações *publisher* e sua inversa *isPublishedBy*.

Dado que documentos podem estar contidos em outros, é utilizada a relação transitiva *isContainedIn*. Versionamento de documentos é também representado por uma relação transitiva, neste caso, pela relação *isVersionOf*. O número da versão de um documento é representado pelo atributo *hasVersionNumber*.

Indivíduos da classe *Document*, em geral, possuem algum tópico ou assunto relacionado; isto é representado pelo atributo *hasTopic*. O conteúdo de indivíduos da classe *Document* é representado pelo atributo *hasContent*.

Vale observar que esta ontologia foi estendida pela aplicação *WebMemex* para que pudesse ser realizada a captura, o acesso e a recomendação de páginas Web. A aplicação *ActiveTheatre* [Hansen & Bardram, 2005] é um outro exemplo de aplicação sensível a contexto que poderia utilizar a ontologia *Document* para modelar prontuários eletrônicos de pacientes que se submetem a atividades cirúrgicas.

²Os caracteres *time:* denotam o espaço de nomes XML associado à ontologia *Time*.

A.6 Ontologia Project

A ontologia Project, descrita na Figura A.6, modela projetos que atores podem estar envolvidos. Projetos são modelados como eventos com extensão intervalar de tempo, por isso esta ontologia importa não apenas a ontologia Actor, mas também a ontologia Temporal Event³.

São representadas duas subclasses de projetos: projetos passados (PastProject) e projetos atuais (CurrentProject). Por meio das relações temporais *time:before* e *time:after* é possível representar a ordem temporal entre ambos tipos de projetos.

Figura A.6: Ilustração da ontologia Project.

Todos os indivíduos da classe Project devem ter um título descritivo, representado pelo atributo *hasTitle*; pelo menos um ator que encabece o projeto, descrito por meio da relação *isHeadedBy* e sua inversa *headsProject*; e pelo menos um ator-membro do projeto, modelado pela relação *hasProjectMember* e sua inversa *isProjectMemberOf*.

Projetos têm sempre uma data de início e uma de término, respectivamente, representadas pelos atributos *hasStartDate* e *hasEndDate*, sub-propriedades de *time:beginPointOf* e *time:endPointOf*.

³Os caracteres *tEve:* denotam o espaço de nomes XML associado à ontologia Temporal Event.

A.7 Avaliação das ontologias de apoio

Segundo a Tabela A.1, ontologias de apoio do modelo *SeCoM* utilizam tanto construtores de *OWL Lite* quanto de *OWL DL*. Ou seja, aplicações que utilizam apenas ontologias *OWL Lite*, como a ontologia *Relationship*, podem utilizar máquinas de inferência com essa expressividade apenas. Para aplicações que necessitam utilizar ontologias de apoio *OWL DL*, como a ontologia *Document*, é sugerido que utilizem máquinas de inferência mais completas, tal como a *Pellet*.

Tabela A.1: Expressividade lógica e caracterização estrutural das ontologias de apoio do modelo *SeCoM*. Adaptado de Bulcão Neto & Pimentel [2006a].

Ontologia	OWL	Expressividade	Triplas	Classes	Propriedades	Instâncias
Contact	DL	SOIF(D)	164	17	21	0
Role	Lite	ALR+IF(D)	104	15	14	0
Relationship	Lite	ALR+HIF(D)	109	11	19	0
Knowledge	Lite	ALR+IF(D)	91	12	13	0
Document	DL	SHOIF(D)	511	27	72	19
Project	DL	SHOIF(D)	517	55	69	19

Nos casos das ontologias *Contact*, *Document* e *Project*, cujas complexidades estão em torno de SHOIF(D)⁴, configura-se como alta a complexidade de tempo no pior caso de um processo de inferência. O mesmo não se aplica nos casos das ontologias *OWL Lite*, caso das ontologias *Relationship*, *Role* e *Knowledge*.

Quanto à caracterização estrutural das ontologias ilustradas na Tabela A.1, além de *Actor*, *Document* também importa a ontologia *Time*, e *Project* importa a ontologia *Temporal Event* que, por sua vez, importa a ontologia *Time*. Por isso as ontologias *Document* e *Project* apresentam um grande número de triplas em relação às demais ontologias de apoio.

Quanto à avaliação de desempenho de processos de inferência sobre as ontologias de apoio do modelo *SeCoM*, a Tabela A.2 apresenta os tempos médios (em milissegundos) de cada um dos tempos intermediários medidos para cada ontologia.

Com respeito ao tempo de carregamento do arquivo da ontologia (TCA), consumiu-se, em média, 82,8% de um processo de inferência completo. Por terem o maior número de triplas, as ontologias *Document* e *Project* foram responsáveis pela maior parcela dessa porcentagem.

Considerando o tempo de carregamento de modelos RDF de ontologias (TCM) em memória, as ontologias *Document* e *Project* têm grande influência sobre os 16,6% consumidos em relação ao tempo total de inferência sobre cada ontologia de apoio. Isto também se deve ao fato dessas conterem o maior número de triplas em relação

⁴Construtores OWL de lógica de atributos, complemento, propriedades funcional, transitiva e inversa, hierarquia de propriedades, nominais e tipos de dados.

Tabela A.2: Tempo médio de sub-processos de inferência sobre ontologias de apoio do modelo *SeCoM* (em ms). Adaptado de Bulcão Neto & Pimentel [2006a].

Ontologia	TCA	TCM	TVD	TVC	TC	TF	TAI
Contact	1695.4	71.2	210.4	22.6	56.8	9.4	0
Role	1652.3	45.8	144.9	19.3	49.6	6.5	0
Relationship	1655.5	46.6	153.9	12.7	37.6	2.6	0
Knowledge	1652.4	47.6	137.6	14.6	42.8	5.7	0
Document	1866.7	115.1	610.3	39.7	102.8	23.7	10.5
Project	1857.3	111.7	596.3	44.3	100.8	20.8	12

às demais ontologias analisadas na Tabela A.2.

Assim como para as ontologias principais do modelo *SeCoM*, o tempo de validação de dialeto de ontologia (TVD) é bastante oneroso, consumindo, em média, 60,7% do tempo total de inferência. Mantém-se, assim, a recomendação de desabilitar este serviço de validação em máquinas de inferências que o oferecem.

O tempo de cálculo de consistência de ontologia (TVC) foi menor para ontologias com menor expressividade lógica, como Role, Relationship e Knowledge. O cálculo de consistência representa, em média, 5,7% do processo de inferência no experimento.

No experimento com as ontologias de apoio, aquelas com maior expressividade lógica tiveram mais alto tempo de classificação de ontologia (TC), no caso as ontologias Document e Project. Neste experimento, o tempo de classificação representa uma média de 15% do tempo total de inferência sobre as ontologias de apoio.

O tempo de fusão de ontologia (TF) apresentou-se maior para as ontologias que importam um conjunto maior de triplas, como Document e Project. O tempo de fusão consome, em média, 2,2% de todo o processo de inferência sobre as ontologias de apoio, o que corrobora mais uma vez a viabilidade da abordagem de duas camadas do modelo *SeCoM*.

Ontologias sem instâncias declaradas apresenta tempo de associação de instâncias a classes de ontologia (TAI) nulo; caso das ontologias Contact, Role, Relationship e Knowledge. Neste experimento, os tempos TAI representam uma média de 1% do tempo total de inferência sobre as ontologias analisadas.

A.8 Considerações finais

As mesmas questões de projeto identificadas quando da experiência das ontologias principais do modelo *SeCoM* são verdadeiras em relação às ontologias de apoio. Ou seja, é importante para um desenvolvedor de aplicação sensível a contexto baseada em ontologias entender a influência dos sub-processos de inferência no tempo de resposta dessa aplicação.

Referências Bibliográficas

- Abowd, G., Atkeson, C., Hong, J., Long, S., Kooper, R., e Pinkerton, M. (1997). Cyberguide: a mobile context-aware tour guide. *ACM Wireless Networks*, 3:421–433.
- Abowd, G. D. (1999). Software engineering issues for ubiquitous computing. In *Proceedings of the International Conference on Software Engineering (ICSE'99)*, pp. 75–84, Los Angeles, CA, USA.
- Abowd, G. D. e Mynatt, E. D. (2000). Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58.
- Abowd, G. D., Mynatt, E. D., e Rodden, T. (2002). The human experience. *IEEE Pervasive Computing*, 1(1):48–57.
- Aizawa, K., Tancharoen, D., Kawasaki, S., e Yamasaki, T. (2004). Efficient retrieval of life log based on context and content. In *Proceedings of the ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE'04)*, pp. 22–31, New York, USA. ACM Press.
- Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154.
- Allen, J. e Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579.
- ANSI/IEEE (1998). *830-1998 Recommended Practice for Software Requirements Specifications (ANSI/IEEE)*. ISBN: 1-7381-0332-2. IEEE CS Press. Disponível na Internet em <http://ieeexplore.ieee.org/xpl/standardstoc.jsp?isnumber=15571&isYear=1998>. Visitado em 22/05/2006.

- Antoniou, G. e van Harmelen, F. (2004). *A Semantic Web Primer (Cooperative Information Systems)*. ISBN: 0262012103. The MIT Press. 272 páginas.
- Arruda Jr., C. R. E., Bulcão Neto, R. F., e Pimentel, M. G. C. (2003). Open context-aware storage as a Web Service. In *Proceedings of the International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'03) in conjunction with the ACM/IFIP/USENIX International Middleware Conference (Middleware'03)*, pp. 81–87, Rio de Janeiro, Brazil. Disponível na Internet em <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedInternationalWorkshopPapers/ArrudaJrEtAl-MPAC-2003.pdf>. Visitado em 23/05/2006.
- Assad, M., Kay, J., e Kummerfeld, B. (2005). The Keep-in-Touch system. In *Proceedings of the Workshop on Situating Ubiquitous Computing in Everyday Life: Bridging the Social and Technical Divide in conjunction with the International Conference on Ubiquitous Computing (UbiComp'05)*, pp. 1–4, Tokyo, Japan.
- Ayars, J., Bulterman, D., Cohen, A., Day, K., Hodge, E., Hoschka, P., Hyche, E., Jourdan, M., Kim, M., Kubota, K., Lanphier, R., Layaïda, N., Michel, T., Newman, D., van Ossenbruggen, J., Rutledge, L., Saccocio, B., Schmitz, P., e ten Kate, W. (2005). Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/SMIL2/>. Visitado em 19/05/2006.
- Baader, F., Calvanese, D., McGuinness, D., Nardi, D., e Patel-Schneider, P. F., editores (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. 574 páginas.
- Back, M., Cohen, J., Gold, R., Harrison, S., e Minneman, S. (2001). Listen Reader: An electronically augmented paper-based book. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'01)*, pp. 23–29, Seattle, Washington, United States. ACM Press.
- Banavar, G. e Bernstein, A. (2002). Software infrastructure and design challenges for ubiquitous computing applications. *Communications of the ACM*, 45(12):92–96.
- Bardram, J. E. (2004). Applications of context-aware computing in hospital work: examples and design principles. In *Proceedings of the ACM Symposium on Applied Computing (SAC'04)*, pp. 1574–1579. ACM Press.
- Barwise, J. e Etchemendy, J. (2002). *Language, Proof and Logic*. ISBN: 157586374X. Center for the Study of Language and Information, package ed. 598 páginas.

- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., e Stein, L. A. (2004). OWL Web Ontology Language Reference, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/owl-ref/>. Visitado em 10/03/2006.
- Beckett, D. (2004). RDF/XML syntax specification (revised), W3C recommendation. Disponível na Internet em <http://www.w3.org/TR/rdf-syntax-grammar/>. Visitado em 10/03/2006.
- Bernardo, J. M. e Smith, A. F. M. (2000). *Bayesian Theory*. ISBN: 047149464X. John Wiley & Sons.
- Berners-Lee, T. (1997). Metadata architecture. Disponível na Internet em <http://www.w3.org/DesignIssues/Metadata>. Visitado em 10/03/2006.
- Berners-Lee, T. (2000). Semantic Web on XML. Disponível na Internet em <http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html>. Visitado em 10/03/2006.
- Berners-Lee, T., Fielding, R., e Masinter, L. (1998). IETF RFC 2396: Uniform resource identifiers (URI): Generic syntax. Disponível na Internet em <http://www.ietf.org/rfc/rfc2396.txt>. Visitado em 10/03/2006.
- Berners-Lee, T., Hendler, J., e Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):35–43.
- Biron, P. V. e Malhotra, A. (2004). XML Schema Part 2: Datatypes, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/xmlschema-2/>. Visitado em 10/03/2006.
- Bizer, C. e Westphal, D. (2006). Developers guide to Semantic Web toolkits for different programming languages. Disponível na Internet em <http://www.wiwiss.fu-berlin.de/suhl/bizer/toolkits/>. Visitado em 12/09/2006.
- Blom, J., Chipchase, J., e Lehtikainen, J. (2005). Contextual and cultural challenges for user mobility research. *Communications of the ACM*, 48(7):37–41.
- Borriello, G., Chalmers, M., LaMarca, A., e Nixon, P. (2005). Delivering real-world ubiquitous location systems. *Communications of the ACM*, 48(3):36–41.
- Brand, A., Daly, F., e Meyers, B. (2003). *Metadata Demystified: A guide for publishers*. The Sheridan Press and NISO Press. 19 páginas. Disponível na Internet em http://www.niso.org/standards/resources/Metadata_Demystified.pdf. Visitado em 19/05/2006.

- Brank, J., Grobelnik, M., e Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD'05) at 7th International Multi-conference on Information Society (IS'05)*, Ljubljana, Slovenia. 4 páginas.
- Bravo, J., Hervás, R., Chavira, G., e Nava, S. (2006). Modeling contexts by RFID-sensor fusion. In *Proceedings of the Workshop on Context Modelling and Reasoning (CoMoRea'06) in conjunction with the 4th IEEE Conference on Pervasive Computing and Communications (PerCom'06)*, pp. 30–34, Pisa, Italy. IEEE CS Press.
- Bray, T., Hollander, D., Layman, A., e Tobin, R. (2004). Namespaces in XML 1.1, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/xml-names11/>. Visitado em 30/05/2006.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., e Yergeau, F. (2006). Extensible Markup Language, (XML) 1.0 (Fourth Edition), W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/REC-xml>. Visitado em 08/11/2006.
- Brewster, C., Alani, H., Dasmahapatra, S., e Wilks, Y. (2004). Data driven ontology evaluation. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC '04)*, pp. 1–4, Lisbon, Portugal.
- Brickley, D. e Guha, R. V. (2004). RDF vocabulary description language 1.0: RDF Schema, W3C recommendation. Disponível na Internet em <http://www.w3.org/TR/rdf-schema/>. Visitado em 10/03/2006.
- Brickley, D. e Miller, L. (2005). FOAF vocabulary specification. Disponível na Internet em <http://xmlns.com/foaf/0.1/>. Visitado em 10/03/2006.
- Brown, M. (1996). Supporting user mobility. In *Proceedings of the IFIP World Conference on Mobile Communications: Technology, Tools Applications, Authentication and Security*, pp. 69–77, Canberra, Australia. Chapman & Hall.
- Brown, P. J., Bovey, J. D., e Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64.
- Bulcão Neto, R. F., Kudo, T. N., e Pimentel, M. G. C. (2006). POCap: A software process for ontology-based context-aware computing. Relatório Técnico 273, ICMC-USP, São Carlos, Brasil. 16 páginas.
- Bulcão Neto, R. F., Kudo, T. N., e Pimentel, M. G. C. (2006). Using a software process for ontology-based context-aware computing: A case study. In *Anais do*

Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06), Natal-RN, Brasil. ACM Press. 10 páginas.

Bulcão Neto, R. F., Jardim, C. H. O., Camacho-Guerrero, J. A., Lobato, D. C., e Pimentel, M. G. C. (2004). A context-based Web Service approach to communities of practice. In *XXXI Seminário Integrado de Software e Hardware (SEMISH'04)*. 15 páginas. Disponível na Internet em <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianConferencePapers/BulcaoEtAl-SEMISH-2004.pdf>. Visitado em 23/05/2006.

Bulcão Neto, R. F., Jardim, C. H. O., Camacho-Guerrero, J. A., e Pimentel, M. G. C. (2004). A Web service approach for providing context information to CSCW applications. In *Proceedings of 2nd Latin American Web Congress (LA-Web'04)*, pp. 78–85, Ribeirão Preto-SP, Brazil. IEEE CS Press. Disponível na Internet em <http://dx.doi.org/10.1109/WEBMED.2004.1348145>. Visitado em 23/05/2006.

Bulcão Neto, R. F., Macedo, A. A., Camacho-Guerrero, J. A., e Pimentel, M. G. C. (2005). Configurable semantic services leveraging applications context-aware. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'05)*, pp. 1–9, Poços de Caldas-MG, Brasil. ACM Press. Disponível na Internet em <http://doi.acm.org/10.1145/1114223.1114233>. Visitado em 22/05/2006.

Bulcão Neto, R. F. e Pimentel, M. G. C. (2003). Interoperabilidade semântica entre aplicações cientes de contexto. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'03)*, pp. 371–385, Salvador-BA, Brasil. Disponível na Internet em <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianConferencePapers/BulcaoPimentel-Webmedia-2003.pdf>. Visitado em 22/05/2006.

Bulcão Neto, R. F. e Pimentel, M. G. C. (2003). Interoperabilidade semântica entre aplicações cientes de contexto: uma abordagem ontológica. In *Anais do Workshop de Teses e Dissertações como evento integrante do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'03)*, pp. 577–580, Salvador-BA, Brasil. Disponível na Internet em <http://tidia-ae.incubadora.fapesp.br/portal/publications/RefereedBrazilianWorkshopPapers/BulcaoPimentel-WkspTeses-2003.pdf>. Visitado em 22/05/2006.

Bulcão Neto, R. F. e Pimentel, M. G. C. (2004). Explorando conceitos de Web Semântica em Computação ciente de contexto. Workshop de Ontologias, ICMC-USP, São Carlos-SP, Brasil. Disponível na Internet em http://tidia-ae.incubadora.fapesp.br/portal/publications/other_documents/BulcaoPimentel-WkspOnto-2004.pdf. Visitado em 23/05/2006.

- Bulcão Neto, R. F. e Pimentel, M. G. C. (2005). Toward a domain-independent semantic model for context-aware computing. In *Proceedings of the 3rd Latin American Web Congress (LA-Web'05)*, pp. 61–70, Buenos Aires, Argentina. IEEE CS Press. Disponível na Internet em <http://dx.doi.org/10.1109/LAWEB.2005.43>. Visitado em 22/05/2006.
- Bulcão Neto, R. F. e Pimentel, M. G. C. (2006). Performance evaluation of inference services for ubiquitous computing. In *Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06)*, Natal-RN, Brasil. ACM Press. 8 páginas.
- Bulcão Neto, R. F. e Pimentel, M. G. C. (2006). Performance evaluation of ubiquitous inference services: Reasoning-related issues. Relatório Técnico 272, ICMC-USP, São Carlos, Brasil. 17 páginas.
- Bulcão Neto, R. F., Prazeres, C. V. S., e Pimentel, M. G. C. (2006). *Web Semântica: Teoria e Prática*, Cap. 2, pp. 47–86. Tópicos em Sistemas Interativos e Colaborativos. SBC Press. Texto referente a mini-curso proferido no Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'06), Natal-RN, Brasil.
- Bulcão Neto, R. F., Teixeira, C. A. C., e Pimentel, M. G. C. (2005). A Semantic Web-based infrastructure for supporting context-aware applications. In *Proceedings of the IFIP International Conference on Embedded and Ubiquitous Computing (EUC'05)*, number 3824 in Lecture Notes in Computer Science, pp. 900–909, Nagasaki, Japan. Springer. Disponível na Internet em http://dx.doi.org/10.1007/11596356_89. Visitado em 22/05/2006.
- Bulcão Neto, R. F., Udipi, Y. B., e Battle, S. (2004). Agent-based mediation in Semantic Web Service Framework. In *Proceedings of First AKT Workshop on Semantic Web Services (AKT-SWS'04)*, v. 122, pp. 5–8, Milton Keynes, UK. CEUR-WS. Disponível na Internet em <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-122/paper2.pdf>. Visitado em 11/11/2006.
- Burrell, J., Gay, G. K., Kubo, K., e Farina, N. (2002). Context-aware computing: A test case. In *Proceeding of the International Conference on Ubiquitous Computing (UbiComp'02)*, pp. 1–15, Göteborg, Sweden.
- Burton-Jones, A., Storey, V. C., Sugumaran, V., e Ahluwalia, P. (2005). A semiotic metrics suite for assessing the quality of ontologies. *Data Knowledge Engineering*, 55(1):84–102.
- Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., e Wilkinson, K. (2004). Jena: Implementing the Semantic Web recommendations. In *Proceedings of the 13th International World Wide Web Conference (WWW'04)*, pp. 74–83.

- Casati, R. e Varzi, A. C. (1999). *Parts and Places: The Structures of Spatial Representation*. ISBN: 0-262-03266-X. MIT Press.
- Cattelan, R. G., Baldochi Jr, L. A., e Pimentel, M. G. C. (2003). Processing and storage middleware support for capture and access applications. In *Companion Proceedings of the 2003 ACM/IFIP/USENIX International Middleware Conference (Middleware'03)*, page 315, Rio de Janeiro, Brazil.
- Chamberlin, D. D. e Boyce, R. F. (1974). SEQUEL: A structured English query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control (FIDET'74)*, pp. 249–264, Ann Arbor, Michigan. ACM Press.
- Chee, Y.-M., Magaña, J.-A., Franke, K., Froumentin, M., Russell, G., Madhvanath, S., Seni, G., Tremblay, C., e Yaeger, L. (2004). Ink markup language, W3C working draft. Disponível na Internet em <http://www.w3.org/TR/InkML>. Visitado em 15/05/2006.
- Chen, H., Finin, T., e Joshi, A. (2004). Semantic Web in the context broker architecture. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, pp. 277–286, Orlando, Florida, USA. IEEE CS Press.
- Chen, H., Finin, T., Joshi, A., Perich, F., Chakraborty, D., e Kagal, L. (2004). Intelligent agents meet the Semantic Web in smart spaces. *IEEE Internet Computing*, 8(6):69–79.
- Chen, H., Perich, F., Finin, T., e Joshi, A. (2004). SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '04)*, pp. 258–267, Cambridge, MA, USA.
- Chen, P. P. (1976). The entity-relationship model — Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.
- Chen, W., Wang, C.-L., e Lau, F. C. M. (2004). A collaborative and semantic data management framework for ubiquitous computing environment. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC'04)*, pp. 962–971, Aizu-Wakamatsu, Japan.
- Cimiano, P. e Völker, J. (2005). Text2Onto: A framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05)*, number 3513 in Lecture Notes in Computer Science, pp. 227–238, Alicante, Spain. Springer.

- Colmerauer, A. e Roussel, P. (1992). The birth of Prolog. In *Proceedings of the 2nd ACM SIGPLAN Conference on History of Programming Languages*, pp. 37–52. ACM Press.
- ContentGuard (2001). Extensible Rights Markup Language (XrML) 2.0 specification, Part I: Primer. Disponível na Internet em http://www.xrml.org/get_XrML.asp. Visitado em 19/05/2006.
- Cooperstock, J. R., Tanikoshi, K., Beirne, G., Narine, T., e Buxton, W. A. S. (1995). Evolution of a reactive environment. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, pp. 170–177, Denver, Colorado, USA. ACM Press.
- Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., e Reagle, J. (2002). The Platform for Privacy Preferences 1.0 (P3P1.0) specification, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/P3P/>. Visitado em 19/05/2006.
- Cristani, M. e Cuel, R. (2005). A survey on ontology creation methodologies. *International Journal on Semantic Web and Information Systems*, 1(2):49–69.
- Davies, N., Cheverst, K., Mitchell, K., e Efrat, A. (2001). Using and determining location in a context-sensitive tour guide. *Computer*, 34(8):35–41.
- Dawson, F. e Howes, T. (1998). vCard MIME directory profile. Disponível na Internet em <ftp://ftp.isi.edu/in-notes/rfc2426.txt>. Visitado em 10/03/2006.
- Dawson, F. e Stenerson, D. (1998). Internet calendaring and scheduling core object specification (iCalendar). Disponível na Internet em <http://www.ietf.org/rfc/rfc2445.txt>. Visitado em 10/03/2006.
- DCMI (2004). Dublin Core Metadata Element Set, version 1.1: Reference description. Disponível na Internet em <http://dublincore.org/documents/dces/>. Visitado em 10/03/2006.
- Dey, A. K. (2000). *Providing architectural support for building context-aware applications*. PhD thesis, College of Computing, Georgia Institute of Technology. Disponível na Internet em <http://www-static.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf>. Visitado em 08/11/2006.
- Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

- Dey, A. K., Abowd, G., e Salber, D. (1999). A Context-based Infrastructure for Smart Environments. In *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99)*, pp. 114–128.
- Dey, A. K., Abowd, G. D., e Wood, A. (1998). CyberDesk: a framework for providing self-integrating context-aware services. In *Proceedings of the 3rd International Conference on Intelligent User Interfaces (IUI'98)*, pp. 47–54, San Francisco, California, USA. ACM Press.
- Dey, A. K., Salber, D., e Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal*, 16(2-4):97–166.
- Dey, A. K., Salber, D., Abowd, G. D., e Futakawa, M. (1999). The Conference Assistant: Combining context-awareness with wearable computing. In *Proceedings of the 3rd IEEE International Symposium on Wearable Computers (ISWC'99)*, pp. 21–28, San Francisco, California, USA. IEEE CS Press.
- Ding, L., Finin, T. W., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., e Sachs, J. (2004). Swoogle: A search and metadata engine for the Semantic Web. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'04)*, pp. 652–659, Washington, DC, USA.
- Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., e Kolari, P. (2005). Finding and ranking knowledge on the Semantic Web. In *Proceedings of the 4th International Semantic Web Conference (ISWC'05)*, pp. 156–170, Athens, Greece. Springer Verlag.
- Dollin, C. (2006). HOWTO for Jena 2 schemagen. Disponível na Internet em <http://jena.sourceforge.net/how-to/schemagen.html>. Visitado em 13/09/2006.
- Donini, F. M., Lenzerini, M., Nardi, D., e Schaerf, A. (1996). Reasoning in description logics. In Brewka, G., editor, *Foundation of Knowledge Representation*, pp. 191–236. CSLI-Publications.
- Durst, M. e Freytag, A. (2003). Unicode in XML and other markup languages. Technical Report 20, Unicode Consortium. Disponível na Internet em <http://unicode.org/unicode/reports/tr20/>. Visitado em 30/05/2006.
- Eisinger, R., Manzato, M. G., e Goularte, R. (2005). Devices descriptions for context-based content adaptation. In *Proceedings of the 3rd Latin American Web Congress*, pp. 121–129, Buenos Aires, Argentina. IEEE CS Press.
- Elite Care (2006). Senior housing and nursing homes in portland oregon. Disponível na Internet em http://www.elite-care.com/streaming_video/family_portal_full_screen_strmv.wmv. Visitado em 27/04/2006.

- Ellis, A. e Hagino, T., editores (2005). *Proceedings of the 14th International Conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*. ACM Press.
- Elrod, S., Hall, G., Costanza, R., Dixon, M., e Rivières, J. D. (1993). Responsive office environments. *Communications of the ACM*, 36(7):84–85.
- ezOWL (2006). Plugin ezOWL para Protégé. Disponível na Internet em <http://iweb.etri.re.kr/ezowl/index.html>. Visitado em 29/05/2006.
- Fayolle, G., Malyshev, V. A., e Menshikov, M. V. (1995). *Topics in the Constructive Theory of Countable Markov Chains*. ISBN: 0521461979. Cambridge University Press.
- Fensel, D. (2001). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. ISBN: 3540416021. Springer, 1st ed. 138 páginas.
- Fensel, D., Hendler, J. A., Lieberman, H., e Wahlster, W., editores (2005). *Spinning the Semantic Web: bringing the World Wide Web to Its Full Potential*. ISBN: 0-262-06232-1. The MIT Press. 503 páginas.
- Fleck, M., Frid, M., Kindberg, T., O'Brien-Strain, E., Rajani, R., e Spasojevic, M. (2002). From informing to remembering: ubiquitous systems in interactive Museums. *IEEE Pervasive Computing*, 1(2):13–21.
- Flippo, F., Krebs, A., e Marsic, I. (2003). A framework for rapid development of multimodal interfaces. In *Proceedings of the 5th International Conference on Multimodal Interfaces (ICMI'03)*, pp. 109–116, Vancouver, British Columbia, Canada. ACM Press.
- Forstadius, J., Lassila, O., e Seppanen, T. (2005). RDF-Based model for context-aware reasoning in rich service environment. In *Proceedings of the 3rd International Conference on Pervasive Computing and Communications Workshops (PerCom'05 Workshops)*, pp. 15–19, Kauai Island, Hawaii, USA. IEEE CS Press.
- Freed, N. e Borenstein, N. (1996). Multipurpose internet mail extensions (MIME) part one: Format of internet messaged bodies. Network Working Group, Standards Track, The Internet Society.
- Fuggetta, A. (2000). Software process: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE'00)*, pp. 25–34, Limerick, Ireland.
- Gangemi, A., Pisanelli, D. M., e Steve, G. (1998). An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data and Knowledge Engineering*, 31:183–220.

- Gargouri, Y., Lefebvre, B., e Meunier, J.-G. (2005). Ontology maintenance using textual analysis. *Journal of Systemics, Cybernetics and Informatics*, 1(5):1–6.
- Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., e Tu, S. W. (2003). The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123.
- GEONgrid.org (2005). GEON: Cyberinfrastructure for the Geosciences. Disponível na Internet em <http://www.geongrid.org/>. Visitado em 30/05/2006.
- Geyer, W., Richter, H. A., e Abowd, G. D. (2005). Towards a smarter meeting record-capture and access of meetings revisited. *Multimedia Tools and Applications*, 27(3):393–410.
- Gil, Y., Motta, E., Benjamins, V. R., e Musen, M. A., editores (2005). *Proceedings of the 4th International Semantic Web Conference (ISWC'05)*, number 3729 in Lecture Notes in Computer Science, Galway, Ireland. Springer.
- Gómez-Peréz, A. (1996). Towards a framework to verify knowledge sharing technology. *Expert Systems with Application*, 11(4):519–529.
- Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of Intelligent Systems*, 16(3):391–409.
- Gómez-Pérez, A. e Corcho, O. (2002). Ontology languages for the Semantic Web. *IEEE Intelligent Systems*, 17(1):54–60.
- Gómez-Pérez, A., Corcho, O., e Fernandez-Lopez, M. (2004). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. ISBN: 1852335513. Springer, 1st ed. 415 páginas.
- Gómez-Pérez, A. (2004). *Handbook on Ontologies*, Cap. 13, Ontology Evaluation, pp. 251–274. International Handbooks on Information Systems. Springer, 1st ed.
- Goularte, R., Cattelan, R. G., Guerrero, J. A. C., Jr., V. R. I., e da Graça Campos Pimentel, M. (2004). Interactive multimedia annotations: enriching and extending content. In *Proceedings of the ACM Symposium on Document Engineering (DocEng'04)*, pp. 84–86.
- Grant, J. e Beckett, D. (2004). RDF test cases. Disponível na Internet em <http://www.w3.org/TR/rdf-testcases/#ntriples>. Visitado em 10/03/2006.
- Grégoire, B. (2005). protegeDocgen: Protégé plugin for report generation v0.99. Disponível na Internet em <http://protege-docgen.sourceforge.net/>. Visitado em 13/06/2006.

- Grimm, R. (2004). one.world: Experiences with a pervasive computing architecture. *IEEE Pervasive Computing*, 3(3):22–30.
- Grimm, R., Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., e Wetherall, D. (2004). System support for pervasive applications. *ACM Transactions on Computer Systems (TOCS)*, 22(4):421–486.
- Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220.
- Gruninger, M. e Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, Canada. 10 páginas.
- Gu, T., Pung, H. K., e Zhang, D. Q. (2004). Toward an OSGi-based infrastructure for context-aware applications. *IEEE Pervasive Computing*, 3(4):66–74.
- Gu, T., Pung, H. K., e Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18.
- Gu, T., Qian, H. C., Yao, J. K., e Pung, H. K. (2003). An architecture for flexible service discovery in OCTOPUS. In *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN '03)*, pp. 291–296, Dallas, Texas, USA.
- Guarino, N. e Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65.
- Guha, R. V. (2001). rdfDB : An RDF Database. Disponível na Internet em <http://www.guha.com/rdfdb/>. Visitado em 10/03/2006.
- Hansen, T. R. e Bardram, J. E. (2005). ActiveTheatre – A collaborative, event-based capture and access system for the operating theatre. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'05)*, number 3660 in Lecture Notes in Computer Science, pp. 375–392, Tokyo, Japan. Springer.
- Heer, J., Newberger, A., Beckmann, C., e Hong, J. I. (2003). liquid: Context-aware distributed queries. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'03)*, pp. 140–148, Seattle, Washington, USA.
- Heer, J., Newberger, A., Beckmann, C., e Hong, J. I. (2003). liquid: Context-Aware Distributed Queries. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'03)*, number 2864 in Lecture Notes in Computer Science, pp. 140–148, Seattle, Washington, USA. Springer.

- Helal, S. (2005). Programming pervasive spaces. *IEEE Pervasive Computing*, 4(1):84–87.
- Henricksen, K. e Indulska, J. (2004). Modelling and using imperfect context information. In *Proceedings of the Workshop on Context Modelling and Reasoning (CoMoRea'04) in conjunction with the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom'04)*, pp. 33–37, Orlando, FL, USA. IEEE CS Press.
- Henricksen, K. e Indulska, J. (2004). A software engineering framework for context-aware pervasive computing. In *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom'04)*, pp. 77–86, Orlando, FL, USA. IEEE CS Press.
- Henricksen, K. e Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Journal of Pervasive and Mobile Computing*, 2(1):37–64.
- Henricksen, K., Indulska, J., e Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *Proceedings of the International Conference on Pervasive Computing (Pervasive'02)*, number 2414 in Lecture Notes in Computer Science, pp. 167–180, Zürich, Switzerland.
- Hess, C. K. e Campbell, R. H. (2003). A context-aware data management system for ubiquitous computing applications. In *Proceedings of the International Conference on Distributed Computing Systems*, pp. 294–301.
- Hess, C. K., Román, M., e Campbell, R. H. (2002). Building Applications for Ubiquitous Computing Environments. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pp. 16–29, London, UK. Springer-Verlag.
- Hirtle, D., Boley, H., Grosz, B., Kifer, M., Sintek, M., Tabet, S., e Wagner, G. (2005). Schema specification of RuleML 0.9. Disponível na Internet em <http://www.ruleml.org/spec>. Visitado em 07/06/2006.
- Hobbs, J. R. e Pan, F. (2004). An ontology of time for the Semantic Web. *ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing*, 3(1):66–85.
- Hobbs, J. R. e Pan, F. (2006). Time ontology in OWL. Disponível na Internet em <http://www.w3.org/2001/sw/BestPractices/OEP/Time-Ontology>. Visitado em 30/05/2006.
- Hong, J. I. (2002). The Context Fabric: an infrastructure for context-aware computing. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, pp. 554–555, Minneapolis, Minnesota, USA. ACM Press.

- Hong, J. I. e Landay, J. A. (2001). An infrastructure approach to context-aware computing. *Human-Computer Interaction (HCI) Journal*, 16(2-3):287–303.
- Hong, J. I. e Landay, J. A. (2004). An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys'04)*, pp. 177–189, Boston, MA, USA.
- Hong, J. I.-A. (2005). *An Architecture for Privacy-Sensitive Ubiquitous Computing*. PhD thesis, University of California at Berkeley, Computer Science Division.
- Hori, T. e Nishida, Y. (2005). Ultrasonic sensors for the elderly and caregivers in a nursing home. In *Proceedings of the Seventh International Conference on Enterprise Information Systems (ICEIS'05)*, pp. 110–115.
- Horrocks, I., Parsia, B., Patel-Schneider, P., e Hendler, J. (2005). Semantic Web architecture: Stack or two towers? In *Principles and Practice of Semantic Web Reasoning (PPSWR'05)*, number 3703 in Lecture Notes in Computer Science, pp. 37–41. Springer.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., e Dean, M. (2004). SWRL: A Semantic Web rule language combining OWL and RuleML, W3C member submission. Disponível na Internet em <http://www.w3.org/Submission/SWRL/>. Visitado em 07/06/2006.
- Horrocks, I., Patel-Schneider, P. F., e van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a Web ontology language. *Journal of Web Semantics*, 1(1):7–26.
- Hsi, S. e Fait, H. (2005). RFID enhances visitors' museum experience at the Exploratorium. *Communications of the ACM*, 48(9):60–65.
- IDF (2006). *The DOI Handbook*. International DOI Foundation, Oxford, United Kingdom, 4.3.0 ed. Disponível na Internet em http://www.doi.org/handbook_2000/DOIHandbook-v4-3.pdf. Visitado em 19/05/2006.
- IEEE (2006). IEEE 802.11, The working group setting the standards for wireless LANs. Disponível na Internet em <http://grouper.ieee.org/groups/802/11/>. Visitado em 25/04/2006.
- Indulska, J., Robinson, R., Rakotonirainy, A., e Henricksen, K. (2003). Experiences in using CC/PP in context-aware systems. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM'03)*, number 2574 in Lecture Notes in Computer Science, pp. 247–261, Melbourne, Australia. Springer.

- ISO (1995). Information technology – software life cycle processes. ISO/IEC 12207:1995, International Organization for Standardization. Disponível na Internet em <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=21208&ICS1=35&ICS2=80&ICS3=>. Visitado em 07/11/2006.
- ISO (2001). Software engineering, product quality, part 1: Quality model. ISO/IEC 9126-1, International Organization for Standardization. Disponível na Internet em <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22749&ICS1=35&ICS2=80&ICS3=>. Visitado em 22/05/2006.
- Jain, A. K. e Ross, A. (2004). Multibiometric systems. *Communications of the ACM*, 47(1):34–40.
- Jardim, C. H. O., Bulcão Neto, R. F., Godoy, R. P., Ribas, H. M. B., Munson, E. V., Arruda Jr., C. R. E., e Pimentel, M. G. C. (2005). Web Services enabling ubiquitous computing applications: Lessons learned by integrating ubiquitous e-learning applications. *International Journal of Web Services Practices*, 1(1-2):142–152. Disponível na Internet em <http://nwesp.org/ijwsp/2005/vol1/13.pdf>. Visitado em 23/05/2006.
- Jardim, C. H. O., Bulcão Neto, R. F., Ribas, H. M. B., Munson, E. V., e Pimentel, M. G. C. (2005). Web Services Enabling Context-aware Applications: Lessons Learned by Integrating e-Learning Applications. In *International Conference on Next Generation Web Services Practices*, pp. 400–405, Seoul, Korea. IEEE CS Press. ISBN: 0-7695-2452-4.
- JEITA (2002). Exchangeable image file format for digital still cameras: Exif version 2.2. Technical Report CP-3451, Standard of Japan Electronics and Information Technology Industries Association. Disponível na Internet em <http://www.exif.org/Exif2-2.PDF>. Visitado em 09/06/2006.
- Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., e Hendler, J. (2005). SWOOP: A Web ontology editing browser. *Journal of Web Semantics*, 4(2):1–20.
- Kelly, B. e Dodds, L. (2004). Using FOAF to support community-building. In *Proceedings of the IADIS International Conference Web Based Communities (WBC'04)*, Lisbon, Portugal. IADIS Press. 4 páginas.
- Khedr, M. e Karmouch, A. (2004). A context-sensitive middleware for managing embedded pervasive environments. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC'04)*, number 3207 in Lecture Notes in Computer Science, pp. 1085–1095, Aizu-Wakamatsu, Japan. Springer.

- Kim, P., Gargi, U., e Jain, R. (2005). Event-based multimedia chronicling systems. In *Proceedings of the 2nd ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE'05)*, pp. 1–12, Singapore. ACM Press.
- Kindberg, T. e Barton, J. (2001). A web-based nomadic computing system. *Computer Networks*, 35(4):443–456.
- Kindberg, T. e Fox, A. (2002). System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1):70–81.
- Kindberg, T., Morris, H., Schettino, J., Serra, B., Spasojevic, M., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., e Krishnan, V. (2002). People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376.
- Klyne, G. e Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and abstract syntax, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/rdf-concepts/>. Visitado em 10/03/2006.
- Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. H., e Tran, L. (2004). Composite Capability/Preference Profiles (CC/PP): structure and vocabularies 1.0, W3C recommendation. Disponível na Internet em <http://www.w3.org/TR/CCPP-struct-vocab/>. Visitado em 10/03/2006.
- Kortuem, G., Garcia, A., Marshall, I., Mascolo, C., Morrison, R., e Sloman, M. (2006). Proceedings of the workshop on Software Engineering Challenges for Ubiquitous Computing. Disponível na Internet em <http://ubicomp.lancs.ac.uk/workshops/seuc2006/>. Visitado em 24/03/2006.
- Lacy, L. W. (2005). *OWL: Representing Information Using the Web Ontology Language*. ISBN: 1412034485. Trafford Publishing. 302 páginas.
- Lahlou, S., Langheinrich, M., e Röcker, C. (2005). Privacy and trust issues with invisible computers. *Communications of the ACM*, 48(3):59–60.
- Landay, J. A. e Davis, R. C. (1999). Making sharing pervasive: Ubiquitous computing for shared note taking. *IBM Systems Journal*, 38(4):531–550.
- Larson, J. A., Raman, T., e Raggett, D. (2003). Multimodal Interaction Framework. Disponível na Internet em <http://www.w3.org/TR/mmi-framework>. Visitado em 15/05/2006.
- Lee, B.-W. e Yeo, A. W. (2005). Integrating sketch and speech inputs using spatial information. In *Proceedings of the 7th International Conference on Multimodal Interfaces (ICMI'05)*, pp. 2–9, Toronto, Italy. ACM Press.

- López, M. F., Gómez-Pérez, A., Sierra, J. P., e Sierra, A. P. (1999). Building a chemical ontology using Methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1):37–46.
- Macedo, A. A., Bulcão Neto, R. F., Camacho-Guerrero, J. A., Jardim, C. H. O., Cattelan, R. G., Inácio Jr, V. R., e Pimentel, M. G. C. (2005). Linking everyday presentations through context information. In *3rd IW3C2 Latin American Web Conference*, pp. 130–139, Buenos Aires, Argentina. IEEE CS Press.
- Macedo, A. A., Camacho-Guerrero, J. A., Baldochi Jr., L. A., Catellan, R. G., e Pimentel, M. G. C. (2006). Automatically linking live experiences captured with a ubiquitous infrastructure. *Multimedia Tools and Applications*. Aceito em 23/12/2005.
- Macedo, A. A., Truong, K. N., Camacho-Guerrero, J. A., e Pimentel, M. G. C. (2003). Automatically Sharing Web Experiences through a Hyperdocument Recommender System. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pp. 48–56, Nottingham, UK. ACM Press.
- Maedche, A. e Staab, S. (2002). Measuring similarity between ontologies. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW '02)*, number 2473 in Lecture Notes in Computer Science. Springer.
- Manola, F. e Miller, E. (2004). RDF Primer, W3C recommendation. Disponível na Internet em <http://www.w3.org/TR/REC-rdf-syntax/>. Visitado em 10/03/2006.
- Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., e Sycara, K. (2005). OWL-S: Semantic markup for Web Services. Disponível na Internet em <http://www.w3.org/Submission/OWL-S/>. Visitado em 16/03/2006.
- Matuszek, C., Witbrock, M. J., Kahlert, R. C., Cabral, J., Schneider, D., Shah, P., e Lenat, D. B. (2005). Searching for common sense: Populating Cyc from the Web. In *Proceedings of the 20th National Conference on Artificial Intelligence Conference (AAAI'05)*, pp. 1430–1435, Pennsylvania, USA. AAAI Press.
- McCulloch, W. S. e Pitts, W. (1988). A logical calculus of the ideas immanent in nervous activity. pp. 15–27.
- McGlashan, S., Burnett, D. C., Carter, J., Danielsen, P., Ferrans, J., Hunt, A., Lucas, B., Porter, B., Rehor, K., e Tryphonas, S. (2004). Voice extensible markup language (VoiceXML) version 2.0, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/voicexml20>. Visitado em 15/05/2006.

- McGuinness, D. L. (2000). Wine ontology. Disponível na Internet em <http://www.daml.org/ontologies/76>. Visitado em 05/06/2006.
- McGuinness, D. L. e van Harmelen, F. (2004). OWL Web Ontology Language Overview, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/owl-features/>. Visitado em 10/03/2006.
- Miller, G. (1998). *WordNet: An Electronic Lexical Database*. ISBN: 0-262-06197-X. MIT Press. 423 páginas.
- Miller, L., Seaborne, A., e Reggiori, A. (2002). Three implementations of SquishQL: a simple RDF query language. In *Proceedings of the International Semantic Web Conference (ISWC'02)*, number 2342 in Lecture Notes in Computer Science, pp. 423–435. Springer.
- Minsky, M. (1974). A framework for representing knowledge. Technical report, Cambridge, MA, USA.
- Nagel, K., Kidd, C. D., O'Connell, T., Dey, A., e Abowd, G. D. (2001). The Family Intercom: Developing a context-aware audio communication system. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'01)*, number 2201 in Lecture Notes in Computer Science, pp. 176–183, Atlanta, Georgia, USA. Springer-Verlag.
- NASA (2006). Semantic Web for Earth and Environmental Terminology (SWEET). Disponível na Internet em <http://sweet.jpl.nasa.gov/sweet>. Visitado em 31/05/2006.
- Noy, N. F. e McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory. Disponível na Internet em <http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html>. Visitado em 22/05/2006.
- Noy, N. F. e Munsen, M. A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pp. 450–455, Menlo Park, CA, USA. AAAI Press.
- Olofsson, S., Carlsson, V., e Sjolander, J. (2006). The friend locator: supporting visitors at large-scale events. *Personal and Ubiquitous Computing*, 10(2):84–89.
- OMA (2001). User agent profiling specification (uaprof). Disponível na Internet em <http://www.openmobilealliance.org/tech/affiliates/>

LicenseAgreement.asp?DocName=/wap/wap-248-uaprof-20011020-a.pdf.
Visitado em 29/08/2006.

OMG (2005). MOF 2.0 / XMI Mapping Specification, v2.1. <http://www.omg.org/technology/documents/formal/xmi.htm>. Visitado em 30/05/2006.

OMG (2006). Software Process Engineering Metamodel, version 1.1. <http://www.omg.org/technology/documents/formal/spem.htm>. Visitado em 23/03/2006.

OpenCyc.org (2005). OpenCyc: The project. Disponível na Internet em <http://www.opencyc.org/>. Visitado em 10/03/2006.

Oviatt, S., Lunsford, R., e Coulston, R. (2005). Individual differences in multimodal integration patterns: What are they and why do they exist? In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI'05)*, pp. 241–249, Portland, Oregon, USA. ACM Press.

Oviatt, S. L. (2002). *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, Cap. 14, Multimodal Interfaces, pp. 286–304. Lawrence Erlbaum Associates.

Pan, Z. (2005). Benchmarking DL reasoners using realistic ontologies. In *Proceedings of the International Workshop on OWL: Experiences and Directions (OED'05)*. 7 páginas.

Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *Proceedings of the International Symposium on Wearable Computers (ISWC'98)*, pp. 92–99, Pittsburgh, Pennsylvania, USA. IEEE CS Press.

Pease, A. e Niles, I. (2002). IEEE standard upper ontology: A progress report. *Knowledge Engineering Review, Special Issue on Ontologies and Agents*, 17:65–70.

Pessoa, R. M., Calvi, C. Z., Filho, J. G. P., e Andreão, R. V. (2006). Aplicação de um middleware sensível ao contexto em um sistema de telemonitoramento de pacientes cardíacos. In *XXXIII Seminário Integrado de Software e Hardware (SEMISH'06)*. 15 páginas. Disponível na Internet em <http://www.lprm.inf.ufes.br/~camilo/docs/arq0086.pdf>. Visitado em 10/11/2006.

Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., e Hahnel, D. (2004). Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57.

Picard, R. W. (2000). *Affective Computing*. ISBN: 0262661152. The MIT Press. 304 páginas.

- Pieraccini, R., Dayanidhi, K., Bloom, J., Dahan, J., Phillips, M., Goodman, B. R., e Prasad, K. V. (2004). Multimodal conversational systems for automobiles. *Communications of the ACM*, 47(1):47–49.
- Pimentel, M. G. C., Baldochi Jr., L. A., e Catellan, R. G. (2006). Prototyping applications to document the human experience. *IEEE Pervasive Computing*. Aceito em 25/04/2006.
- Pinto, H. S. e Martins, J. (2001). A methodology for ontology integration. In *Proceedings of the 1st International Conference on Knowledge Capture (K-CAP '01)*, pp. 131–138, New York, USA. ACM Press.
- Pinto, H. S. e Martins, J. P. (2002). Evolving ontologies in distributed and dynamic settings. In *Proceedings of the Eight International Conference on Principles and Knowledge Representation and Reasoning (KR'02)*, pp. 365–374, Toulouse, France. Morgan Kaufmann.
- Pinto, H. S. e Martins, J. P. (2004). Ontologies: How can they be built? *Knowledge Information Systems*, 6(4):441–464.
- Porzel, R. e Malaka, R. (2004). A task-based approach for ontology evaluation. In *Proceedings of the ECAI Workshop on Ontology Learning and Population (OLP '04)*, pp. 9–16, Valencia, Spain.
- Prud'hommeaux, E. e Seaborne, A. (2006). SPARQL query language for RDF, W3C candidate recommendation. Disponível na Internet em <http://www.w3.org/TR/rdf-sparql-query/>. Visitado em 09/05/2006.
- Raento, M., Oulasvirta, A., Petit, R., e Toivonen, H. (2005). Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59.
- Raggett, D., Le Hors, A., e Jacobs, I. (1999). Hypertext Markup Language (HTML) 4.01, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/html4/>. Visitado em 10/03/2006.
- Ranganathan, A. e Campbell, R. H. (2003). An infrastructure for context-awareness based on first order logic. *Personal Ubiquitous Computing*, 7(6):353–364.
- Ranganathan, A., Campbell, R. H., Ravi, A., e Mahajan, A. (2002). ConChat: A context-aware chat program. *IEEE Pervasive Computing*, 1(3):51–57.
- Reeves, L. M., Lai, J., Larson, J. A., Oviatt, S., Balaji, T. S., Buisine, S., Collings, P., Cohen, P., Kraal, B., Martin, J.-C., McTear, M., Raman, T., Stanney, K. M.,

- Su, H., e Wang, Q. Y. (2004). Guidelines for multimodal user interface design. *Communications of the ACM, Special issue: Multimodal interfaces that flex, adapt, and persist*, 47(1):57–59.
- Reynolds, D. (2006). Jena 2 inference support. Disponível na Internet em <http://jena.sourceforge.net/inference/>. Visitado em 22/05/2006.
- RMI (2006). The Rule Markup Initiative. Disponível na Internet em <http://www.ruleml.org/>. Visitado em 12/06/2006.
- Rode, J. A., Toye, E. F., e Blackwell, A. F. (2004). The fuzzy felt ethnography – understanding the programming patterns of domestic appliances. *Personal Ubiquitous Computing*, 8(3-4):161–176.
- Román, M. e Campbell, R. H. (2003). A middleware-based application framework for active spaces applications. In *Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware'03)*, pp. 433–454, Rio de Janeiro, Brazil.
- Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., e Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83.
- Rumbaugh, J., Jacobson, I., e Booch, G. (2005). *Unified modeling language user guide*. ISBN: 0321267974. Addison Wesley Professional, 2nd ed. 496 páginas.
- Salber, D., Dey, A. K., e Abowd, G. A. (1999). The Context Toolkit: aiding the development of context-enabled applications. In *Proceedings of the ACM SIGCHI conference on Human factors in computing systems*, pp. 434–441, Pittsburgh, Pennsylvania, United States. ACM Press.
- SALT Forum (2002). Speech Application Language Tags (SALT) 1.0 Specification. Disponível na Internet em <http://www.saltforum.org/saltforum/downloads/SALT1.0.pdf>. Visitado em 16/05/2006.
- Schilit, B. e Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32.
- Schilit, B. N., Adams, N., Gold, R., Tso, M., e Want, R. (1993). The ParcTab mobile computing system. In *Proceedings of the Workshop on Workstation Operating Systems (WWOS-IV)*, pp. 34–39, Napa, California, USA.
- Shakhnarovich, G. e Moghaddam, B. (2005). *Face Recognition in Subspaces, Handbook of Face Recognition*. ISBN: 038740595X. Springer. 398 páginas.

- Sheshagiri, M., Sadeh, N., e Gandon, F. (2004). Using semantic web services for context-aware mobile applications. In *Proceedings of the Workshop on Context Awareness in conjunction with the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, pp. 1–7, Boston, Massachusetts, USA. USENIX.
- Shi, Y., Xie, W., Xu, G., Shi, R., Chen, E., Mao, Y., e Liu, F. (2003). The Smart Classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 2(2):47–55.
- Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., e Katz, Y. (2006). Pellet: A practical OWL-DL reasoner. Submitted for publication to the Journal of Web Semantics. Disponível na Internet em <http://www.mindswap.org/papers/PelletJWS.pdf>. Visitado em 22/05/2006.
- Sleeman, D. e Reul, Q. (2006). CleanONTO: Evaluating taxonomic relationships in ontologies. In *Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON '06) in conjunction with the 15th International World Wide Web Conference (WWW '06)*, Edinburgh, Scotland. 8 páginas.
- Smith, M. K., Welty, C., e McGuinness, D. L. (2004). OWL Web Ontology Language Guide, W3C Recommendation. Disponível na Internet em <http://www.w3.org/TR/owl-guide/>. Visitado em 10/03/2006.
- Sommerville, I. (2006). *Software Engineering*. ISBN: 0321313798. Addison Wesley, 8th ed. 864 páginas.
- Spence, M., Driver, C., e Clarke, S. (2005). Sharing context history in mobile, context-aware trails-based applications. In *Proceedings of the 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE'05) in conjunction with the International Conference on Pervasive Computing*, pp. 1–5, Munich, Germany.
- Stanford, V. (2002). Using pervasive computing to deliver elder care. *IEEE Pervasive Computing*, 1(1):10–13.
- Starner, T., Auxier, J., Ashbrook, D., e Gandy, M. (2000). The Gesture Pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proceedings of the International Symposium on Wearable Computing (ISWC'00)*, pp. 87–94, Atlanta, Georgia, USA. IEEE CS Press.
- Streitz, N. e Nixon, P. (2005). Introduction. *Communications of the ACM, Special issue: The disappearing computer*, 48(3):32–35.

- Swartz, A. (2006). The Semantic Web in breadth. Disponível na Internet em <http://logicerror.com/semanticWeb-long>. Visitado em 12/06/2006.
- Takahashi, M., Ito, S., Sumi, Y., Tsuchikawa, M., Kogure, K., Mase, K., e Nishida, T. (2004). A layered interpretation of human interactions captured by ubiquitous sensors. In *Proceedings of the ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE'04)*, pp. 32–38, New York, USA. ACM Press.
- Tan, J., Zhang, D., Wang, X., e Cheng, H. (2005). Enhancing semantic spaces with event-driven context interpretation. In *Proceedings of the Third International Conference on Pervasive Computing and Communications (PerCom'05)*, pp. 80–97, Kauai Island, Hawaii, USA. IEEE CS Press.
- Tannenbaum, A. (2001). *Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand*. ISBN: 0201719762. Addison-Wesley Professional, 1st ed. 400 páginas.
- Tello, A. L. e Gómez-Pérez, A. (2004). ONTOMETRIC: A method to choose the appropriate ontology. *Journal of Database Management*, 15(2):1–18.
- Tempich, C. e Volz, R. (2003). Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In *Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON'03)*. 12 páginas.
- Tran, Q., Calcaterra, G., e Mynatt, E. (2005). Cook's collage: Déjà Vu display for a home kitchen. In *Proceedings of the IFIP International Conference on Home-Oriented Informatics and Telematics (HOIT'05)*, pp. 15–32, York, United Kingdom. Springer.
- Truong, K. N. e Abowd, G. D. (2004). INCA: A software infrastructure to facilitate the construction and evolution of ubiquitous capture & access applications. In *Proceedings of the International Conference on Pervasive Computing (PerCom'04)*, number 3001 in Lecture Notes in Computer Science, pp. 140–157, Vienna, Austria. Springer-Verlag.
- Truong, K. N., Abowd, G. D., e Brotherton, J. A. (2001). Who, what, when, where, how: Design issues of capture & access applications. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'01)*, number 2201 in Lecture Notes in Computer Science, pp. 209–224, Atlanta, Georgia, USA. Springer.
- Tsetsos, V., Anagnostopoulos, C., Kikiras, P., Hasiotis, T., e Hadjiefthymiades, S. (2005). A human-centered semantic navigation system for indoor environments. In *Proceedings of the IEEE International Conference on Pervasive Services (ICPS'05)*, pp. 146–155, Santorini, Greece. IEEE CS Press.

- Tuffield, M. M., Harris, S., Dupplaw, D. P., Chakravarthy, A., Brewster, C., Gibbins, N., O'Hara, K., Ciravegna, F., Sleeman, D., Shadbolt, N. R., e Wilks, Y. (2006). Image annotation with Photocopain. In *Proceedings of the First International Workshop on Semantic Web Annotations for Multimedia (SWAMM'06) in conjunction with the 15th International World Wide Web Conference (WWW'06)*, pp. 1–14, Edinburgh, Scotland. ACM Press.
- Uschold, M. (1996). Converting an informal ontology into Ontolingua: Some experiences. In *Proceedings of the ECAI '96 Workshop on Ontological Engineering*, pp. 1–17, Budapest.
- Velardi, P., R.Navigli, Cuchiarelli, A., e F.Neri (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*, Cap. 6, Evaluation of Ontolearn, a Methodology for Automatic Population of Domain Ontologies. ISBN: 1-58603-523-1. IOS Press. 180 páginas.
- Völkel, M., Krötzsch, M., Vrandečić, D., Haller, H., e Studer, R. (2006). Semantic Wikipedia. In *Proceedings of the 15th International World Wide Web Conference (WWW'06)*, pp. 1–10, Edinburgh, Scotland. ACM Press.
- Vrandečić, D., Suarez-Figueroa, M. C., Gangemi, A., e Sure, Y., editores (2006). *4th International Workshop on Evaluation of Ontologies for the Web (EON '06) in conjunction with the 15th International World Wide Web Conference (WWW '06)*, v. 179. CEUR Workshop Proceedings. ISSN 1613-0073.
- W3C (2001). Semantic Web Activity. Disponível na Internet em <http://www.w3.org/2001/sw/>. Visitado em 12/02/2006.
- W3C (2006). Web Services Activity. Disponível na Internet em <http://www.w3.org/2002/ws/>. Visitado em 05/09/2006.
- Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S., e Zhang, D. (2004). Semantic Space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39.
- Wang, X., Gu, T., Zhang, D., e Pung, H. (2004). Ontology based context modeling and reasoning using OWL. In *Proceedings of Workshop on Context Modeling and Reasoning (CoMoRea'04) in conjunction with the IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, Orlando, Florida, USA. 5 páginas.
- Want, R., Hopper, A., Falcão, V., e Gibbons, J. (1992). The Active Badge location system. *ACM Transactions on Information Systems*, 10(1):91–102.

- Weal, M. J., Cruikshank, D., Michaelides, D. T., Millard, D. E., De Roure, D. C., Hornecker, E., Halloran, J., e Fitzpatrick, G. (2006). A reusable, extensible infrastructure for augmented field trips. In *Proceedings of the 2nd IEEE Workshop on Pervasive Learning (PerEL'06) in conjunction with the 4th IEEE Conference on Pervasive Computing and Communications (PerCom'06)*, pp. 201–205, Pisa, Italy. IEEE CS Press.
- Weis, T., Handte, M., Knoll, M., e Becker, C. (2006). Customizable pervasive applications. In *Proceedings of 4th IEEE Conference on Pervasive Computing and Communications (PerCom'06)*, pp. 239–244, Pisa, Italy. IEEE CS Press.
- Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.
- Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84.
- Westeyn, T., Brashear, H., Atrash, A., e Starner, T. (2003). Georgia Tech Gesture Toolkit: Supporting experiments in gesture recognition. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI'03)*, pp. 85–92, Vancouver, British Columbia, Canada. ACM Press.
- Wijnalda, G., Pauws, S., Vignoli, F., e Stuckenschmidt, H. (2005). A personalized music system for motivation in sport performance. *IEEE Pervasive Computing*, 4(3):26–32.
- Wilkinson, K., Sayers, C., Kuno, H., e Reynolds, D. (2003). Efficient RDF storage and retrieval in Jena2. Technical Report HPL-2003-266, HP Labs. 18 páginas. Disponível na Internet em <http://www.hpl.hp.com/techreports/2003/HPL-2003-266.pdf>. Visitado em 12/09/2006.
- Wu, H., Siegel, M., e Abhay, S. (2002). Sensor fusion for context understanding. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC'02)*, pp. 13–17, Anchorage, AK, USA. IEEE CS Press.
- Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, 3(8):338–353.
- Zhao, W., Chellappa, R., Phillips, P. J., e Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458.
- Zimmer, T. (2004). Towards a better understanding of context attributes. In *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom'04 Workshops)*, pp. 23–27, Orlando, Florida, USA. IEEE CS Press.