ANA PAULA CHAVES

UM MODELO BASEADO EM *CONTEXT-AWARENESS* PARA DISSEMINAÇÃO DE INFORMAÇÕES EM UM AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

MARINGÁ
2009

ANA PAULA CHAVES

UM MODELO BASEADO EM *CONTEXT-AWARENESS* PARA DISSEMINAÇÃO DE INFORMAÇÕES EM UM AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Elisa Hatsue Moriya Huzita

MARINGÁ
2009

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

C512m

Chaves, Ana Paula
Um modelo baseado em *context-awareness* para disseminação de informações em um ambiente de desenvolvimento distribuído de software / Ana Paula Chaves. -- Maringá, 2009.
149 f. : il.

Orientador : Profª. Drª. Elisa Hatsue Moriya Huzita.
Dissertação (mestrado) - Universidade Estadual de Maringá, Programa de Pós-Graduação em Ciência da Computação, Área de concentração: Ciência da Computação, Linha de Pesquisa: Sistemas de Informação,2009.

1. Softwares. 2. Sistemas de informação. 3. Recuperação da Informação. 4. Tecnologia de comunicação. I. Universidade Estadual de Maringá. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 21.ed.005.1

ANA PAULA CHAVES

UM MODELO BASEADO EM CONTEXT-AWARENESS PARA DISSEMINAÇÃO DE INFORMAÇÕES EM UM AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 31/07/2009.

BANCA EXAMINADORA

Profa. Dra. Elisa Hatsue Moriya Huzita
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Sérgio Roberto Pereira da Silva
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Marcos Roberto da Silva Borges
Universidade Federal do Rio de Janeiro – NCE/UFRJ

Ao meu querido amor, Igor...
Aos meus pais e meus irmãos...
À Olavo Travassos Moreira Netto
(*in memorian*)...

Agradecimentos

Ao meu amor, Igor Fábio Steinmacher, que apareceu na minha vida como um anjo, devolvendo minhas forças e minha vontade de lutar, que fez todos os sacrifícios fazerem sentido...

Aos meus pais Paulo da Silva Chaves e Maria Aparecida de Souza Chaves, aos meus irmãos e cunhado, Maria Helena Chaves e Marcus Vinicius Borges, Amanda Carolina Chaves e André Luiz de Souza Chaves, pelo amor incondicional, o carinho e a força nos momentos difíceis...

À minha professora orientadora Elisa Hatsue Moriya Huzita, pela paciência e confiança com que conduziu os meus passos nessa jornada...

Aos amigos de projeto, César Alberto da Silva, Willian Capato Donegá, Gustavo Sato, Camila Lapasini Leal, Flávio Luiz Schiavoni pelos cafés, lasanhas, *strogonoffs*, cocas-cola e todos os outros momentos culinários que proporcionaram, além da distração, produtivas discussões, idéias e até mesmo consolo nos momentos de dúvida e cansaço...

Aos amigos de turma, em especial Rodrigo Tomaz Pagno e Tiago Lopes Gonçalves, pela companhia, pelos trabalhos, pelas tardes de estudo intermináveis, pelos momentos de diversão e tantas outras coisas que dividimos...

Aos professores do Departamento de Informática, em especial aos professores do grupo de pesquisa GESSD, pela disponibilidade ao esclarecer minhas dúvidas, pelas dicas e conselhos...

À Vaninha Vieira, pelo apoio inestimável, pelas palavras de incentivo, pelas jornadas intensas de trabalho a que nos submetemos, com o único propósito de ver concretizados nossos trabalhos que, por destino ou por graça, se entrelaçaram...

Aos amigos "mouraoenses" Igor Scaliante Wiese e Rafael Liberato Roberto, pelas longas e proveitosas discussões sobre ontologia e contexto...

À família Pallaro dos Reis, especialmente Carolina Pallaro dos Reis e Isadora Pallaro dos Reis, por terem se tornado minha família maringaense, pelas conversas na madrugada, pelas experiências culinárias, pelas dificuldades que enfrentamos juntas, como irmãs...

À Maria Inês Davanço, por assumir os papéis de secretária, amiga, psicóloga e tantos outros, sempre com o mesmo carinho e dedicação...

À Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pelo apoio financeiro...

À Deus, por ter me proporcionado todos esses motivos para agradecer.

Resumo

O Desenvolvimento Distribuído de Software trouxe diversas vantagens competitivas, tais como ganho de produtividade e redução de custos. Entretanto, essas vantagens são acompanhadas de novos desafios, especialmente quando se trata de comunicação entre os indivíduos participantes de trabalho cooperativo. Nesse sentido, técnicas de percepção e gerenciamento de contexto tem sido utilizadas para oferecer aos indivíduos informações que os tornem capazes de perceber o contexto das ações que ocorrem em um ambiente de trabalho. Este trabalho apresenta o modelo DiSEN-CSE (DiSEN-*Context Sensitive Environment*), um modelo baseado em percepção de contexto para disseminação de informações contextuais em um ambiente de desenvolvimento distribuído de software, chamado DiSEN. Esse modelo define de que maneira as informações contextuais existentes no ambiente podem ser capturadas; formalmente representadas para oferecer semântica a essas informações e facilitar a compreensão comum pelos indivíduos dispersos; armazenadas para consultas futuras e, disseminadas para todos os participantes de uma equipe de desenvolvimento trabalhando cooperativamente. Para concretizar o DiSEN-CSE, o foco deste trabalho está em oferecer uma estrutura básica para seu funcionamento, fundamentada em três pilares: (i) identificar quais informações são capazes de compor o contexto das ações e como essas informações influenciam o comportamento do ambiente DiSEN; (ii) desenvolver um modelo de representação, baseado em ontologias, para oferecer semântica às informações; e (iii) desenvolver um gerenciador de notificações, capaz de compartilhar automaticamente essas informações com as entidades interessadas, sem intervenção dos usuários.

Abstract

Distributed Software Development can bring several competitive advantages, such as productivity improvement and cost reduction. However, there are some challenges imposed by this distribution, specially related to communication among users involved on cooperative work. In this sense, awareness techniques and context management are used to offer information to users, allowing them to perceive the context where the actions are occuring within the workspace. This work presents DiSEN-CSE (DiSEN-Context Sensitive Environment) model, a context-awareness based model to share contextual information in a distributed software development workspace, called DiSEN. This model defines how environment contextual information can be captured; formally represented to offer semantics to the information and make it easier to create a common understanding among distributed individuals; stored for future queries; and shared among all participants working cooperatively in a development team. To create DiSEN-CSE the focus of this work is offering a basic structure for it, based on three pillars: (i) to identify which set of information will compose actions context and how this information impacts DiSEN behavior; (ii) to create an ontology-based representation model, offering semantics to the information; (iii) to develop a notification manager able to automatically share this information among stakeholders, without user interference.

Sumário

Lista de Figuras **xiii**

Lista de Tabelas **xv**

1 Introdução **1**
1.1 Considerações Iniciais . . . 1
1.2 Contexto e Motivação . . . 2
1.2.1 Descrição do problema . . . 3
1.2.2 Solução proposta . . . 3
1.3 Objetivos . . . 4
1.4 Metodologia . . . 5
1.5 Organização do Trabalho . . . 7

2 Revisão Bibliográfica **9**
2.1 Desenvolvimento Distribuído de Software . . . 9
2.2 Percepção . . . 10
2.3 Contexto . . . 11
2.3.1 Tipos de contexto . . . 12
2.4 *Context-Awareness* . . . 13
2.5 Técnicas para representar informação contextual . . . 16
2.5.1 Grafos Contextuais . . . 19
2.6 *Context Metamodel* . . . 21
2.7 Considerações Finais . . . 24

3 Trabalhos Relacionados **25**
3.1 Compartilhamento de Informações . . . 25
3.2 Gerenciamento de contexto . . . 31
3.3 Considerações Finais . . . 33

4 Modelo DiSEN-CSE **35**
4.1 Considerações Iniciais . . . 35
4.2 Modelo DiSEN-CSE (DiSEN-*Context Sensitive Environment*) . . . 37
4.2.1 Especificação dos Atores . . . 41
4.2.2 Modelagem e Especificação dos Casos de Uso . . . 42
4.3 Considerações Finais do Capítulo . . . 47

5 Modelagem Contextual **51**
5.1 Considerações Iniciais . . . 51
5.2 Modelagem Contextual do Ambiente DiSEN . . . 54
5.3 Considerações Finais de Capítulo . . . 70

6 Representação de Contexto **73**
6.1 Considerações Iniciais . . . 73
6.2 Ontologias . . . 74
6.3 OntoDiSEN . . . 76
6.4 Considerações Finais de Capítulo . . . 91

7 Disseminação de informações **95**
7.1 Notificações de eventos . . . 96
7.1.1 DiSEN-*Notifier* . . . 97
7.1.2 Exemplo de uso . . . 100
7.2 Considerações Finais . . . 104

8 Conclusões **105**

Referências Bibliográficas **111**

A Documento de Visão **119**
A.1 Introdução . . . 119
A.1.1 Objetivo . . . 119
A.1.2 Escopo . . . 119
A.1.3 Organização do documento . . . 120
A.2 Descrição dos *Stakeholders* e Usuários . . . 120
A.2.1 Resumo dos stakeholders (não usuários). . . . 120
A.2.2 Perfil dos stakeholders (não usuários). . . . 120
A.2.3 Necessidades chave do ambiente sobre o modelo (S/U – Stakeholders Usuários) . . . 121
A.3 Resumo das capacidades . . . 121
A.3.1 Responsabilidades . . . 121
A.3.2 Interfaces externas . . . 121
A.3.3 Características do modelo . . . 122
A.3.4 Precedência e prioridades . . . 122

B Especificação dos Casos de Uso **123**
B.1 Introdução . . . 123
B.1.1 Objetivo . . . 123
B.1.2 Escopo . . . 123
B.1.3 Organização do documento . . . 124
B.2 Especificação dos atores . . . 124
B.3 Diagramas de Casos de Uso . . . 124
B.4 Especificação dos Casos de Uso . . . 127

C Documentação da OntoDiSEN **133**
C.1 Introdução . . . 133
C.1.1 Objetivo . . . 133
C.1.2 Escopo . . . 133
C.1.3 Organização do documento . . . 134
C.2 Questões de competência . . . 134
C.2.1 Questões de competência relacionadas a Usuários . . . 134
C.2.2 Questões de competência relacionadas a Ferramentas . . . 135
C.2.3 Questões de competência relacionadas a Locais . . . 136
C.3 Glossário de termos . . . 137
C.4 Código OWL da ontologia OntoDiSEN . . . 138

Lista de Figuras

1.1 Metodologia de trabalho. 5

2.1 Tipos de contexto. (Brézillon e Pomerol, 1999) 13
2.2 Relação existente entre percepção e contexto. (Adaptada de (Wang et al., 2006)) 14
2.3 Exemplo de grafo contextual (Brézillon, 2007) 20
2.4 Conceitos estruturais do *Context Metamodel*. (Vieira, 2008) 22
2.5 Conceitos comportamentais do *Context Metamodel*. (Vieira, 2008) 23

4.1 Modelo SPC (Pozza, 2005) 36
4.2 Compartilhando informações no DiSEN 38
4.3 Modelo baseado em *context-awareness* para o DiSEN 39
4.4 Representação dos atores do modelo e seus relacionamentos. 41
4.5 Diagrama de Casos de Uso – Visão de Negócio. 42
4.6 Diagrama de Casos de Uso – Capturar informações de contexto. 44
4.7 Diagrama de Casos de Uso – Representar contexto. 45
4.8 Diagrama de Casos de Uso – Processar informações de contexto. 46
4.9 Diagrama de Casos de Uso – Disseminar informações de contexto. 47

5.1 Diagrama de Casos de Uso do ambiente DiSEN. 52
5.2 Processo para especificação de contexto e projeto de sistemas sensíveis ao contexto (Vieira, 2008). 54
5.3 Modelo Conceitual – Foco: Realizar Tarefa 59
5.4 Modelo Comportamental – Foco: Realizar Tarefa 68
5.5 Modelo Comportamental – Atividade: Executar Tarefa 69

6.1 Mapa Conceitual – principais conceitos e relacionamentos da OntoDiSEN. . 81
6.2 Entidades da OntoDiSEN, modeladas na ferramenta Protégé. 82
6.3 Propriedades de objetos da OntoDiSEN, modeladas na ferramenta Protégé. 83
6.4 Propriedades de dados da OntoDiSEN, modeladas na ferramenta Protégé. . 84
6.5 Excerto da OntoDiSEN: Restrições da entidade Usuario em OWL. 92
6.6 OntoDiSEN: Análise de consistência pelo raciocinador Fact++. 93

7.1 Camadas de negócio e infraestrutura do framework FRADE – relação do gerenciador de notificações com os demais componentes do ambiente DiSEN. 98
7.2 Diagrama de Pacotes – DiSEN-*Notifier*. 99
7.3 Acesso do DiSEN-*Notifier* aos suportes da camada de infraestrutura. . . . 100

7.4 Interface de acesso ao DiSEN-*Notifier*. 100
7.5 Inserindo usuária Ana Paula Keys como participante de um projeto. 101
7.6 Modelo Comportamental – Foco: Definir Participante. 102
7.7 DiSEN-*Notifier* comunicando ao usuário sobre inclusão no projeto. 103
7.8 Evento comunicando ao participante inserido no projeto. 103

B.1 Atores do modelo DiSEN-CSE. 124
B.2 Diagrama de Casos de Uso – Visão Geral. 124
B.3 Diagrama de Casos de Uso – Capturar informações de contexto. 125
B.4 Diagrama de Casos de Uso – Representar contexto. 125
B.5 Diagrama de Casos de Uso – Processar informações de contexto. 126
B.6 Diagrama de Casos de Uso – Disseminar informações de contexto. 126

Lista de Tabelas

1.1 Descrição do problema. 4
1.2 Solução proposta. 4

2.1 Técnicas para representação de contexto (Vieira, 2006, 2008). 19

3.1 Intersecção entre percepção e contexto, baseado nos *frameworks* apresentados em (Pinheiro et al., 2001) e (Rosa, 2004). 28

6.1 Subentidades de `ValuePartition`. 90

A.1 Perfil do *stakeholder* Orientadora. 120
A.2 Perfil do *stakeholder* Aluna. 120

B.1 Capturar informações de contexto. 127
B.2 Capturar informações locais. 127
B.3 Recuperar informações do repositório. 127
B.4 Registrar contexto manualmente. 128
B.5 Comunicar gerenciador de notificações. 128
B.6 Representar contexto 128
B.7 Mapear informações para modelo de representação. 129
B.8 Atribuir dependências às informações 129
B.9 Armazenar informações persistentes 129
B.10 Processar informações de contexto 129
B.11 Deduzir novas informações a partir das existentes 130
B.12 Disseminar informações de contexto 130
B.13 Identificar receptores da informação 130
B.14 Identificar mecanismos de percepção 131
B.15 Compartilhar informações 131

Lista de Siglas

API: *Application Programming Interface*
DDS: *Desenvolvimento Distribuído de Software*
GSD: *Global Software Development*
IHC: *Interface Humano-Computador*
IA: *Inteligência Artificial*
CSCW: *Computer Supported Cooperative Work*
ANC: *AulaNet Companion*
NOTICON: *Notification In Context*
NSF: *Notification Specification Language*
CE: *ContextualElement - Elemento Contexual*
SPC: *Sincronização, Percepção e Comunicação*
ADS: *Ambiente de Desenvolvimento de Software*
ADDS: *Ambiente de Desenvolvimento Distribuído de Software*
UML: *Unified Modeling Language*
IDE: *Integrated Development Environment*
DiSEN: *Distributed Software Engineering Environment*
GEESD: *Grupo de Estudos em Engenharia de Software Distribuído*
GUI: *Grafic User Interface*
OWL-DL: *Web Ontology Language - Description Logics*
DiSEN-CSE: *DiSEN-Context Sensitive Environment*
OMG: *Object Management Group, Inc.*
FRADE: *Framework to Infrastructure of Distributed Software Development Environment*

Capítulo

1

Introdução

1.1 Considerações Iniciais

A necessidade cada vez maior de velocidade e precisão na realização de diversas atividades tem incentivado pessoas a trabalhar em conjunto de forma cooperativa. Entretanto, quando a realização de atividades envolve mais de uma pessoa, surgem novas dificuldades referentes ao trabalho em grupo, como capacidade de cooperação e coordenação, exigindo um novo tipo de ferramenta para atender essa demanda (Rosa, 2004).

O Trabalho Cooperativo Apoiado por Computador (CSCW – *Computer Supported Cooperative Work*) busca auxiliar o trabalho cooperativo utilizando tecnologia computacional. Mas existem diversas necessidades em CSCW que precisam ser satisfeitas para garantir que todos os integrantes do grupo consigam compreender as atividades que os demais estão realizando, recuperar informações anteriores e armazenar suas ações e decisões para que possam ser utilizadas no futuro.

Na tentativa de garantir a comunicação clara e não ambígua entre indivíduos envolvidos no trabalho cooperativo, pesquisadores investiram na utilização de técnicas de percepção que, combinadas ao gerenciamento de contexto, favoreceram os estudos acerca de sistemas conhecidos como *context-awareness*. Os sistemas *context-awareness* são aqueles capazes de perceber, de alguma maneira, o contexto das entidades envolvidas no trabalho cooperativo. Segundo Dey e Abowd (1999), entidade representa pessoas, objetos, lugares ou mesmo outros software que interagem com a aplicação.

O desenvolvimento de software com equipes fisicamente distribuídas (DDS – Desenvolvimento Distribuído de Software) surgiu como uma alternativa para se obter vantagens competitivas em termos de ganho de produtividade, melhoria de qualidade e redução de custos (Herbsleb e Moitra, 2001; Prikladnicki e Audy, 2004). Entretanto, a distância física e os fatores dela derivados como diferenças culturais, de idioma, infraestrutura de comunicação, entre diversos outros, dificultam a troca de informações e coordenação entre os membros da equipe, durante todo o ciclo de vida do projeto.

Analisando os trabalhos apresentados no Capítulo 3 e, com base nas características dos sistemas cientes de contexto (*context-aware*), foi possível observar que, embora existam pesquisas voltadas para a caracterização de informações de contexto em sistemas cooperativos, apresentação eficiente das informações para os usuários e compartilhamento de conhecimento, combinar essas técnicas para minimizar os problemas de comunicação em ambientes de desenvolvimento distribuído ainda é um desafio.

De acordo com Kevin C. Desouza (2006), com o aumento dos esforços para possibilitar o desenvolvimento distribuído de software, novos métodos e modelos devem ser desenvolvidos para gerenciar a grande quantidade de conhecimento envolvida nesses projetos. Assim, é oportuno projetar mecanismos que ofereçam meios de aumentar o compartilhamento de informações em um ambiente de desenvolvimento distribuído de software, permitindo que participantes do trabalho cooperativo percebam o contexto dos demais indivíduos envolvidos e garantindo que essas informações possuirão uma representação única, podendo ser compreendidas por todos os membros das equipes, sem ambiguidades. O ambiente utilizado para a realização deste trabalho chama-se DiSEN (*Distributed Software Engineering Environment*) (Huzita et al., 2007). O DiSEN é um ADDS (Ambiente de Desenvolvimento Distribuído de Software) que visa disponibilizar ferramentas de apoio à comunicação, à persistência e à cooperação para equipes de desenvolvimento geograficamente dispersas.

1.2 Contexto e Motivação

Este trabalho corresponde à especificação de um modelo de disseminação de informações de contexto, que será agregado ao ambiente de desenvolvimento distribuído de software DiSEN. O objetivo do ambiente DiSEN é fornecer uma infraestrutura para apoiar o desenvolvimento de software por equipes geograficamente distantes. É necessário, portanto, que seja possível que os indivíduos das equipes dispersas percebam as informações contextuais dos demais participantes da interação. Além disso, as informações devem possuir uma representação única para aumentar a clareza e diminuir as ambiguidades.

1.2.1 Descrição do problema

A utilização de software tem aumentado a cada dia e as empresas de desenvolvimento sentem dificuldades em suprir a demanda crescente com recursos capacitados. As empresas de desenvolvimento têm buscado alternativas para obter custos menores, aumentar a produtividade e diminuir a quantidade de falhas nos projetos.

Nesse contexto, o Desenvolvimento Distribuído de Software (DDS) é uma opção caracterizada pelo desenvolvimento em locais remotos e pela possibilidade de dispor os recursos em âmbito global. No entanto, esse tipo de desenvolvimento traz necessidades, como a gerência do projeto, o processo de desenvolvimento de software, a complexidade e tamanho dos projetos, a tecnologia e a infraestrutura de comunicação disponível, os fatores sociais, culturais, comportamentais, psicológicos, linguísticos e políticos, entre outros. Mais especificamente, analisando os problemas de comunicação, identificou-se que a ausência de elementos como a linguagem das mãos, tão comuns para auxiliar a comunicação face-a-face, dificulta a troca de informações entre pessoas geograficamente dispersas. Dessa forma, para obter uma compreensão clara sobre os objetos de cooperação (que corresponde a cada parte resultante do trabalho cooperativo), é necessário que se conheça também a forma como este objeto foi produzido. Assim, observou-se a necessidade de fornecer informações sobre o contexto das entidades do ambiente no momento da produção dos objetos de interação. Devido às diferenças relacionadas à distância, a representação das informações torna-se uma característica essencial para diminuir as ambiguidades. Dessa forma, a Tabela 1.1 descreve o problema, o impacto e uma possível forma de solução.

1.2.2 Solução proposta

O modelo DiSEN-CSE (DiSEN-*Context Sensitive Environment*), apresentado nesse trabalho, vem ao encontro às necessidades acima descritas, sendo integrado ao ambiente DiSEN. O ambiente DiSEN possui um modelo baseado nas características de Sincronização, Percepção e Comunicação (SPC), que auxilia a cooperação utilizando *workspaces* compartilhados. O modelo baseado em *context-awareness* proposto objetiva agregar à característica de percepção os elementos necessários para capturar informações contextuais, representar, processar e disseminar informações de contexto aos indivíduos interessados, além de prever a existência de um repositório para o armazenamento de informações persistentes. A Tabela 1.2 resume a solução apresentada neste trabalho.

Tabela 1.1: Descrição do problema.

Os problemas são	as barreiras para oferecer percepção de contexto, impostas pela distância geográfica entre as equipes de desenvolvimento.
Afeta	a compreensão das equipes de desenvolvimento geograficamente distantes sobre o contexto das entidades.
O impacto do problema é que	informações sobre a construção dos objetos de cooperação produzidas por equipes distantes podem apresentar ambiguidades; as ambiguidades e falta de clareza podem provocar falhas ou incertezas no desenvolvimento do software; pode haver necessidade de refazer partes do projeto, ou até mesmo, inviabilizá-lo.
Uma solução proposta é	promover a disseminação das informações contextuais para os indivíduos interessados, de forma uniforme e padrão, diminuindo as ambiguidades no desenvolvimento e aumentando a clareza das informações.

Tabela 1.2: Solução proposta.

Para	Ambientes de Desenvolvimento Distribuídos de Software, mais especificamente o ambiente DiSEN.
Com o objetivo de	definir um modelo e especificar seus elementos para possibilitar a captura, representação, armazenamento, processamento e disseminação de informações contextuais.

1.3 Objetivos

Com base nos problemas encontrados e na solução proposta, os objetivos desse trabalho concentram-se na tentativa de minimizar os problemas de disseminação de informações e possibilitar que estas informações sejam compreendidas, sem ambiguidades, por todos os membros das equipes dispersas. Assim, o objetivo geral desta proposta é

"apresentar o modelo DiSEN-CSE, baseado em *context-awareness*, para disseminação de informações contextuais em um Ambiente de Desenvolvimento Distribuído de Software".

Como objetivos específicos, apontam-se:

- especificar os elementos que compõe o DiSEN-CSE, atribuindo responsabilidades a cada parte do modelo;
- identificar as informações do ambiente DiSEN capazes de compor o contexto das interações e especificar a forma como essas informações se relacionam entre si;

- especificar um modelo de representação que ofereça semântica para as informações do ambiente, aumentando a clareza e reduzindo as ambiguidades;
- implementar um gerenciador de notificações de eventos, responsável por disseminar as informações de contexto para as instâncias do ambiente interessadas na informação.

1.4 Metodologia

Para atender aos objetivos desse trabalho, a seguinte metodologia foi utilizada (Figura 1.1):

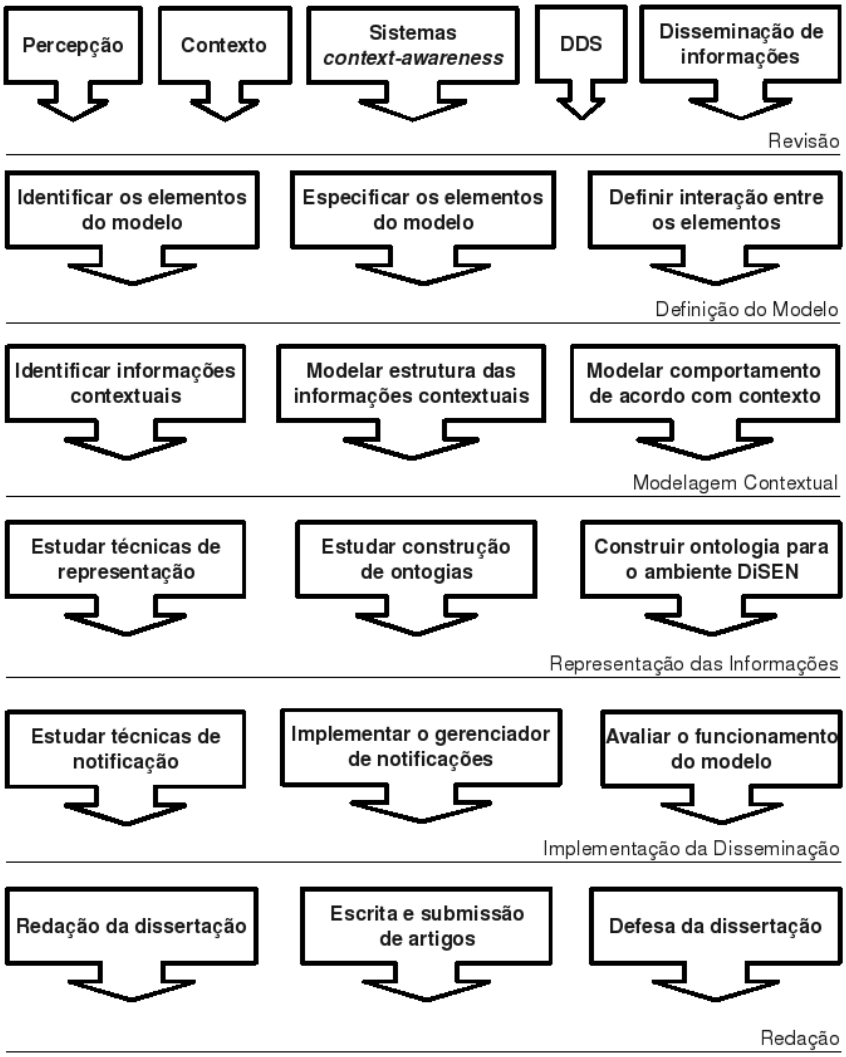

Figura 1.1: Metodologia de trabalho.

Durante a atividade de Revisão, estudos foram realizados sobre diversos pontos específicos da pesquisa: conceitos de percepção, contexto e sistemas *context-awareness*; análise dos trabalhos relacionados nas diversas linhas de investigação científica; questões sobre os problemas existentes na disseminação de informações no desenvolvimento de software com equipes dispersas. Os resultados dessa atividade foram inicialmente discutidos na Seção 1.2, que expôs os problemas levantados e a solução proposta. Os demais resultados podem ser vistos nos Capítulos 2 e 3.

A Definição do Modelo concentrou-se em analisar, com base na Revisão, quais elementos são necessários para tornar os usuários do ambiente DiSEN capazes de perceber mudanças nas informações contextuais e disseminá-las para todos os interessados. Depois de identificados, esses elementos foram detalhados, na forma de casos de uso, para atribuir responsabilidades a cada uma das partes do modelo e as formas de interação. Os resultados dessa atividade estão descritos no Capítulo 4. Os documentos completos estão detalhados no Apêndice A.

Na atividade de Modelagem Contextual, foram identificadas as informações presente no ambiente DiSEN capazes de compor o contexto das interações, como essas informações se relacionam, sua forma de aquisição e como o ambiente se comporta de acordo com as mudanças ocorridas nessas informações. O resultado dessa atividade está detalhado no Capítulo 5 e a documentação completa pode ser encontrada em (Chaves, 2009).

A Representação das Informações contempla a realização de um modelo de representação baseado em ontologias, capaz de definir a semântica das informações do ambiente DiSEN. As informações foram definidas em termos de conceitos ontológicos e os relacionamentos, em termos de propriedades. A descrição dos passos realizados nessa atividade é apresentada no Capítulo 6 e a documentação completa da ontologia gerada pode ser encontrada no Apêndice C.

A atividade de Implementação da Disseminação consiste em especificar e implementar um gerenciador de notificações, capaz de receber informações contextuais (identificadas pela Modelagem Contextual), representá-las na ontologia definida na Representação das Informações, reconhecer os destinatários e disseminá-las. O gerenciador de notificações é o responsável por garantir que as informações contextuais serão efetivamente compartilhadas, compondo a infraestrutura essencial para o funcionamento do modelo. O resultado dessa atividade está descrito no Capítulo 7.

Por fim, a atividade de Redação consiste em produzir este documento de dissertação, na escrita e submissão de artigos relacionados a este trabalho e na defesa da dissertação.

1.5 Organização do Trabalho

Este capítulo apresentou os objetivos deste trabalho, fundamentados nos problemas encontrados no desenvolvimento distribuído de software, mais especificamente, na dificuldade de compartilhamento de informações em ambientes que seguem essa abordagem de desenvolvimento. O restante deste trabalho está organizado da seguinte maneira:

O Capítulo 2 apresenta os conceitos relevantes para o desenvolvimento deste trabalho, entre eles: desenvolvimento distribuído de software, contexto, percepção, percepção de contexto (*context-awareness*), técnicas para representação de informações contextuais e um metamodelo para realizar modelagem conceitual de contexto.

O Capítulo 3 discute trabalhos encontrados na literatura que influenciaram, de alguma maneira, o desenvolvimento do modelo DiSEN-CSE. Esses trabalhos estão divididos entre aqueles que oferecem compartilhamento de informações e os que se preocupam com o gerenciamento de contexto.

O Capítulo 4 descreve o modelo DiSEN-CSE. Cada elemento do modelo é especificado em termos de casos de uso, para detalhar suas funcionalidades.

O Capítulo 5 mostra a modelagem conceitual das informações de contexto e a modelagem do comportamento do ambiente DiSEN, de acordo com as variações de contexto possíveis para uma determinada interação.

O Capítulo 6 apresenta a ontologia desenvolvida para representar a semântica das informações presentes no ambiente DiSEN. Para isso, introduz o conceito de ontologias, a sua importância para o modelo DiSEN-CSE e, então, descreve a metodologia utilizada para o seu desenvolvimento.

O Capítulo 7 contém o detalhamento do gerenciador de notificações de eventos, responsável por disseminar as informações contextuais para os indivíduos interessados.

Por fim, o Capítulo 8 discute os resultados obtidos no desenvolvimento deste trabalho, as contribuições e os trabalhos futuros.

Capítulo

2

Revisão Bibliográfica

Este capítulo apresenta os conceitos relevantes para o desenvolvimento deste trabalho e que fundamentaram o modelo DiSEN-CSE, sendo eles: Desenvolvimento Distribuído de Software, percepção, contexto, *context-awareness*, técnicas para representação de contexto e o metamodelo *Context Metamodel*.

2.1 Desenvolvimento Distribuído de Software

O Desenvolvimento Distribuído de Software (DDS) surgiu acompanhando a necessidade da indústria de software de suprir a crescente demanda por produtos de qualidade. Permitir a interação cooperativa entre centros de desenvolvimento geograficamente distantes tornou-se uma alternativa às organizações que buscavam vantagens competitivas, como proximidade ao mercado, aproveitamento de recursos, ganho de produtividade (proporcionado pelo *round-the-clock*[1]), entre outras. Quando atinge dimensões globais, o DDS pode ser chamado de Desenvolvimento Global de Software (GSD – *Global Software Develpment*).

Entretanto, aliado às vantagens, o GSD trouxe novos desafios, provocados pelas dispersões geográfica e temporal e pelas diferenças cultural, contextual, organizacional e política (Ivcek e Galinac, 2008). Desde então, diversos trabalhos ((Bos et al., 2004; Herbsleb et al., 2001; Herbsleb e Moitra, 2001; Herbsleb, 2007; Sangwan et al., 2006)) foram

[1]Desenvolvimento contínuo, aproveitando a diferença de fuso horário entre países.

realizados com o objetivo de identificar as principais dificuldades do desenvolvimento distribuído.

Dentre os problemas encontrados, um aspecto fundamental é a comunicação. Para que os indivíduos geograficamente distantes trabalhem cooperativamente, é necessário oferecer uma infraestrutura capaz de garantir a eficiente troca de informações e conhecimentos entre os envolvidos.

De acordo com Eckhard (2007), resolver os problemas de comunicação é um desafio, devido à complexidade dos projetos de GSD, causados (i) pela grande quantidade de elementos heterogêneos como pessoas, código fonte, hardware e software, dificultando a integração de ferramentas; (ii) pela grande quantidade de dependência entre os elementos; e (iii) e pelas constantes mudanças nos ambientes de desenvolvimento.

Quando se trata de GSD, as equipes dispersas podem possuir idiomas, vocabulários ou culturas diferentes. Assim, cada equipe pode interpretar de forma distinta as informações disseminadas, o que pode levá-las a erros devido à falta de dados relevantes ou a dúvidas provocadas por possíveis ambiguidades. Além disso, não se pode garantir que os indivíduos obterão as informações necessárias com base na comunicação com outras pessoas, ainda que haja uma infraestrutura de comunicação adequada. Isso porque os indivíduos nem sempre se conhecem pessoalmente, podendo haver falta de confiança em relação aos demais ou ausência de pré-disposição em ajudar outras pessoas envolvidas no trabalho. Em ambos os casos, os custos gerados por essas dificuldades variam desde as necessidades de revisar cronogramas, alocar novos recursos e refazer partes do projeto, podendo até inviabilizar o projeto em questão.

De acordo com Pozza (2005), a intensidade de interação entre usuários geograficamente dispersos ocorre conforme a exigência do trabalho quando existem métodos capazes de oferecer a um indivíduo percepção sobre as atividades realizadas pelos demais membros da equipe. Perceber as ações uns dos outros facilita a compreensão e diminui as barreiras da comunicação, impostas pela distância física. A Seção 2.2 descreve os conceitos de percepção e sua importância para facilitar a comunicação.

2.2 Percepção

Quando um grupo de pessoas realiza uma atividade em conjunto, é necessário garantir que as contribuições individuais serão relevantes para as atividades do grupo como um todo. *Awareness* ou Percepção representa, portanto, a compreensão das atividades dos outros, que oferece um contexto para sua própria atividade (Dourish e Belloti, 1992; Pozza, 2005).

A percepção corresponde ao conhecimento da ocorrência de um evento que compõe o estado mental do usuário (Vieira, 2006). Quando o desenvolvimento de software envolve pessoas fisicamente distantes, compor esse estado mental é um desafio, já que a comunicação e interação entre os indivíduos pode ser bastante reduzida. Dessa forma, é preciso garantir um suporte tecnológico adequado, capaz de prover interação, cooperação e comunicação entre os indivíduos, conhecido como mecanismos de percepção.

Segundo Vieira (2006), dois fatores influenciam a percepção: o modo de interação e os papéis desempenhados pelos participantes do grupo. A interação pode ser síncrona ou assíncrona. A interação síncrona ocorre quando mais de um indivíduo acessa os dados compartilhados de forma concorrente. Exemplos de interações síncronas são as que ocorrem por *chats*, teleapontadores e vídeo conferências. Na interação assíncrona, os indivíduos podem acessar as informações depois de um intervalo de tempo desde a ocorrência do evento. *Emails*, quadro de avisos e grupos de discussão são exemplos desse tipo de interação. Além disso, Molli et al. (2002) descreveram interações multissíncronas, em que os membros têm uma cópia do dado compartilhado e as alteram em paralelo, para então sincronizá-las e estabelecer uma visão comum dos dados.

Quanto aos papéis, cada participante do grupo possui privilégios e responsabilidades e as informações que influenciam seu trabalho dependem dessas características. Conforme Vieira (2006), os papéis dos usuários podem ser: (i) nível operacional – usuários que executam o trabalho; (ii) nível tático – usuários que coordenam o trabalho; e (iii) nível estratégico – usuários que definem metas e objetivos, identificando as práticas que devem ser seguidas ou evitadas.

Quando se trata de desenvolvimento distribuído de software, a distância física impõe que para se conhecer as contribuições de outros membros do grupo torna-se necessário conhecer não apenas o objeto de cooperação, mas também a forma como este foi produzido. Segundo Dourish e Belloti (1992), essas informações refletem a percepção sobre o contexto da equipe, formado pelas circunstâncias que envolvem o desenvolvimento das atividades. Para compreender melhor a relação entre percepção e contexto, que será detalhada na Seção 2.4, a Seção 2.3 descreve o conceito de contexto utilizado nesta proposta e suas características.

2.3 Contexto

A definição de contexto tem sido amplamente discutida nos últimos anos. Diversas pesquisas tem se preocupado em caracterizar sua utilização dentro de uma área de domínio específica, o que inibe a formação de um consenso geral sobre seu significado e forma de

aplicação. As diversas áreas apresentam a definição de contexto baseadas na forma com que a compreendem, de acordo com os interesses específicos de seu trabalho, o que pode ser observado em (Bazire, 2005).

Apesar disso, conforme apontado por Vieira (2008), os pesquisadores concordam em alguns aspectos das definições: (i) contexto existe somente quando relacionado a uma entidade (tarefa, interação ou agente, que pode ser humano ou de software); (ii) contexto é um conjunto de itens associados a uma entidade; e (iii) um item é considerado parte do contexto somente se for útil para apoiar a resolução de um problema.

Com base nessas observações, Vieira (2008) distinguiu os termos contexto e elemento contextual, definindo elemento contextual como "qualquer item de dado ou informação que caracteriza uma entidade em um domínio" e que "o contexto de uma interação entre um agente e uma aplicação, com o objetivo de realizar alguma tarefa, é um conjunto de elementos contextuais instanciados que são necessários para apoiar a execução da tarefa". Além disso, a autora observa que o elemento contextual é estável e pode ser definido em tempo de projeto, enquanto contexto é dinâmico, construído em tempo de execução, durante uma interação. Essas definições foram utilizadas no desenvolvimento deste trabalho por representar claramente a dinamicidade do contexto, relacionando-o com os elementos relevantes para uma determinada interação entre agente e aplicação.

2.3.1 Tipos de contexto

Diversos autores (Brézillon e Pomerol, 1999; Dey e Abowd, 1999; Rosa, 2004) destacam a importância de se filtrar as informações de contexto recebidas, para que seja possível definir a relevância da informação para a atividade a ser realizada. Para classificar as informações de acordo com sua relevância, Brézillon e Pomerol (1999) propuseram uma classificação que divide contexto em dois tipos: (i) conhecimento contextual (*contextual knowledge*) – conhecimento relevante para a tomada de decisão em uma atividade, não sendo explicitamente utilizado, mas influenciando na resolução do problema; e (ii) conhecimento externo (*external knowledge*) – conhecimento que não é relevante à tomada de decisão, embora seja conhecido por um ou mais indivíduos envolvidos na resolução do problema.

Ainda segundo Brézillon e Pomerol (1999), durante a execução da tarefa, o conhecimento contextual é compilado, estruturado e organizado para ser utilizado, direta ou tacitamente, na solução do problema, passando a ser chamado de contexto procedural (*proceduralized context*). Os tipos de contexto são mostrados na Figura 2.1.

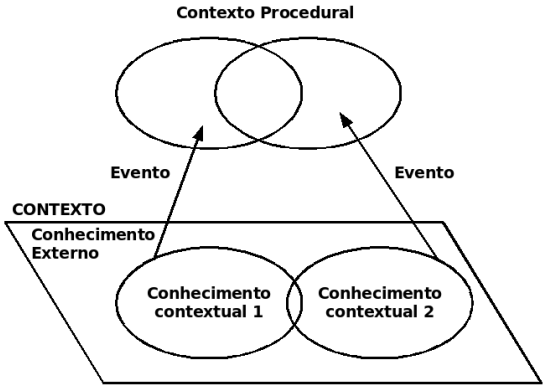

Figura 2.1: Tipos de contexto. (Brézillon e Pomerol, 1999)

Na Figura 2.1, o retângulo envolve o contexto externo e representa todas as informações conhecidas pelos indivíduos ou grupos. Os conhecimentos contextuais 1 e 2 são um subgrupo do conhecimento externo, que contêm as informações relevantes a uma determinada situação. Quando ocorre um evento, o conhecimento contextual é transformado em contexto procedural, sendo utilizado na solução do problema.

É importante observar que o contexto possui um comportamento dinâmico. A construção do contexto procedural é feita com base na junção e organização de elementos ou partes do conhecimento contextual disponível, de acordo com um foco de atenção. O foco de atenção é constituído pelos elementos que devem fazer parte do contexto. Cada vez que o foco muda, os elementos que formavam o contexto procedural voltam ao subconjunto dos conhecimentos contextuais e um novo contexto procedural é formado.

O conhecimento contextual de membros de equipes fisicamente distantes pode ser bastante diferente, já que cada indivíduo possui sua cultura e seus costumes locais. Dessa forma, para realizar uma tarefa em grupo é necessário que os indivíduos possuam um contexto procedural compartilhado (Araujo e Brézillon, 2005), a fim de compreender o contexto uns dos outros e formar seu próprio conhecimento contextual. Quando os indivíduos não possuem em seu conhecimento contextual todas as informações necessárias para a realização da tarefa, podem acrescentar ao seu contexto procedural informações do conhecimento externo, que adquirem com base na comunicação com outros indivíduos ou em dados disponíveis da organização.

2.4 Context-Awareness

As Seções 2.2 e 2.3 definiram percepção e contexto isoladamente. É importante, entretanto, destacar a relação existente entre esses conceitos. Especialmente em ambientes

cooperativos, a necessidade de conhecer as contribuições dos outros membros do grupo torna necessário o conhecimento do objeto de cooperação e da forma como este foi produzido. Segundo Dourish e Belloti (1992), essas informações refletem a percepção de contexto da equipe. A Figura 2.2 foi apresentada em (Wang et al., 2006) e descreve uma maneira de relacionar esses dois conceitos durante uma interação.

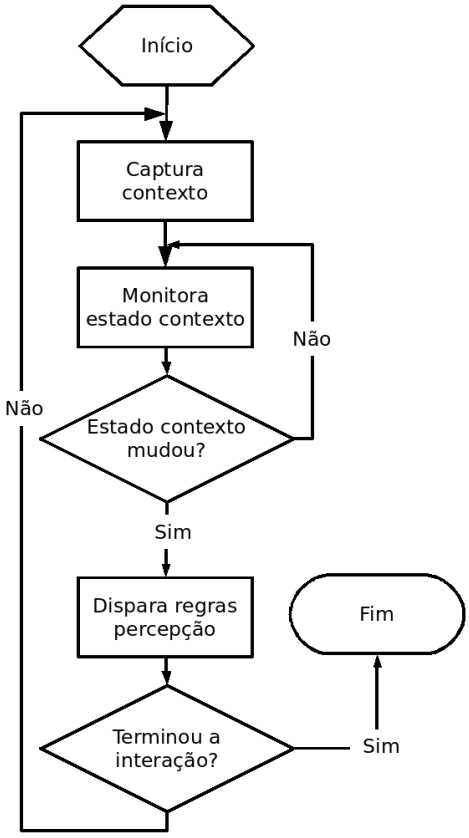

Figura 2.2: Relação existente entre percepção e contexto. (Adaptada de (Wang et al., 2006))

Na Figura 2.2, observa-se que o contexto é capturado e monitorado a partir do início da interação. A cada mudança de contexto, disparam-se as regras de percepção que devem interpretar se as informações serão exibidas instantaneamente, disponibilizadas em segundo plano ou armazenadas para consultas futuras, dependendo do tipo de interação. Esse processo se repete enquanto durar a interação.

Diversos pesquisadores, como os relacionados em (Baldauf et al., 2007), dedicaram seus estudos a aplicações capazes de perceber, de alguma maneira, o contexto das entidades envolvidas no trabalho cooperativo. A essas aplicações, dá-se o nome de sistemas sensíveis ao contexto ou sistemas "*context-aware*".

Quando relacionada à definição de contexto utilizada neste trabalho, sistemas sensíveis ao contexto podem ser definidos como aqueles que gerenciam um conjunto de elementos contextuais relacionados a um domínio de aplicação e utilizam esses elementos para auxiliar um agente na execução de uma tarefa. Esses sistemas são capazes de aumentar a cognição de um agente sobre a tarefa e de fornecer adaptações que facilitem a realização da tarefa em questão (Vieira, 2008).

Para uma aplicação ter a capacidade de oferecer informações de contexto, é necessário seguir alguns passos para transformar os dados, obtidos em sensores, em informações úteis às aplicações. Para isso, vários processos foram propostos para perceber o contexto de uma entidade, como pode ser observado em (Gómez, 2007) e (Dey, 2000). De maneira geral, nota-se que a preocupação está em definir de que maneira o contexto será adquirido, gerenciado, representado e disseminado para as aplicações.

Para oferecer percepção de contexto, as aplicações precisam reconhecer o contexto e gerenciá-lo. Para tanto, é necessário que se desenvolva uma arquitetura capaz de oferecer essas funcionalidades e dar o devido suporte à aplicação que se deseja desenvolver.

Basicamente, as arquiteturas de sistemas *context-aware* encontradas na literatura possuem cinco camadas (Baldauf et al., 2007): sensores – forma de captura dos dados, podendo ser físicos, lógicos ou virtuais; recuperação dos dados – *drivers* para dispositivos físicos e as APIs (*Application Programming Interfaces*) para os sensores lógicos e virtuais; pré-processamento – realiza o raciocínio e a interpretação da informação contextual; armazenamento/gerenciamento – organiza os dados e os oferecem via interface pública para as aplicações-cliente; e aplicação – aplicações-cliente que utilizam as informações contextuais.

De maneira geral, as principais características das aplicações *context-aware* são: apresentar as informações e serviços para um usuário, executar o serviço de um usuário automaticamente e relacionar contexto a informações para apoiar futura recuperação.

Todavia, o desenvolvimento de aplicações *context-aware* apresenta vários desafios. Vieira (2006) destaca a caracterização dos elementos de contexto para uso na aplicação, a aquisição do contexto a partir de fontes heterogêneas, a representação de um modelo semântico formal de contexto, o processamento e interpretação das informações de contexto, a disseminação do contexto a entidades interessadas, o tratamento de questões como segurança e privacidade e o tratamento do desempenho do sistema para apoiar a inclusão de contexto na aplicação.

Observa-se, portanto, que o desenvolvimento de sistemas sensíveis ao contexto envolve alguns fatores, além daqueles relacionados aos sistemas tradicionais, já que se

preocupa com as questões de aquisição, processamento, armazenamento, manipulação e apresentação de informações do contexto de uma tarefa.

Para facilitar a manipulação de informações contextuais, surgiram diversos modelos de representação de contexto ((Baldauf et al., 2007; Gu et al., 2004; Strang e Linnhoff-Popien, 2004; Vieira, 2006)) com o objetivo de oferecer informações a diferentes entidades (pessoas, software ou dispositivos) de maneira que qualquer uma delas tenha a mesma compreensão semântica a respeito do que está sendo informado. A preocupação dos pesquisadores consiste em propor um modelo mais formal e expressivo, que permita que as informações contextuais sejam interoperáveis, podendo ser compreendida independentemente da aplicação.

De acordo com Gu et al. (2004), os modelos para representação de contexto podem ser classificados em três diferentes abordagens: orientada a aplicação, orientada a modelo e orientada a ontologia. Os modelos que seguem a primeira abordagem são, geralmente, proprietários, aplicáveis apenas a determinado sistema ou domínio e possuindo pouca expressividade e formalismo, além de não permitir o compartilhamento do contexto. A segunda abordagem é mais formal que a anterior e utiliza modelagem conceitual, como modelos de ER (Entidade-Relacionamento), UML (*Unified Modeling Language*) ou ORM (*Object-Role Modelling*), para representar contexto. Além disso, pode ser facilmente gerenciada com base de dados relacionais e permite a representação de aspectos temporais das informações, embora também não permita o compartilhamento ou raciocínio sobre o contexto. Por fim, a terceira abordagem utiliza uma ontologia para modelar o contexto tanto de um domínio específico quanto genérico, apoiada pela capacidade potencial de raciocínio e compartilhamento inerentes às tecnologias de Web Semântica.

A seção 2.5 descreve brevemente algumas das principais técnicas existentes para representação de contexto.

2.5 Técnicas para representar informação contextual

Algumas técnicas foram propostas na literatura para oferecer uma melhor representação de contexto, cada qual com suas características particulares. Alguns dos principais modelos são descritos a seguir (Baldauf et al., 2007; Strang e Linnhoff-Popien, 2004; Vieira, 2006):

Modelo baseado em pares de valor-chave: modelo orientado a aplicação que possui a estrutura de dados mais simples para representação de contexto dentro de um domínio específico, em que as capacidades de um serviço são descritas pelo par "valor" e "chave". A "chave" representa o atributo do contexto e possui um "valor"

a ele associado que corresponde à informação propriamente dita. Uma ação é então associada aos pares que formam um determinado contexto e é realizada quando o valor dos atributos "chave" forem correspondentes ao valor.

De acordo com Vieira (2006), a vantagem desse modelo é a simplicidade e facilidade de uso, sendo utilizado pelos sistemas operacionais para armazenar variáveis de ambiente.

Modelo baseado em marcação: também orientado a aplicação, esse modelo utiliza estruturas de dados hierárquicas que consistem de *tags* com atributos e conteúdo, com base no padrão XML (*eXtensible Markup Language*) que facilita a comunicação por diferentes aplicações. É fortemente utilizado para modelagem de perfis, que consiste em oferecer maneiras de reconhecer as preferências dos indivíduos de acordo com suas características (Held et al., 2002).

Embora possa ser utilizada para definir um vocabulário comum e extensível para disponibilizar informações de contexto para diferentes sistemas, essa abordagem é bastante empregada para representar contexto de forma proprietária e limitada a um pequeno conjunto de aspectos (Vieira, 2006).

Modelo baseado em representações gráficas: abordagem orientada a modelos, representa os aspectos contextuais utilizando ER, UML ou ORM, embora Henricksen et al. (2002) afirmem que, comparando ER com UML, a segunda é mais expressiva. De acordo com Strang e Linnhoff-Popien (2004), a estrutura genérica da UML a torna apropriada para modelagem de contexto. Em (Bauer, 2003), são definidos alguns diagramas que expressam os procedimentos e tarefas relacionados ao contexto, conforme segue:

- diagrama de casos de uso – modela os atores envolvidos ou que são influenciados pelo contexto e suas possíveis interações;
- diagrama de componentes – modela os sistemas envolvidos, como bases de dados contendo informações relacionadas ao contexto;
- diagrama de classes – modela as informações estruturais do domínio, indicando quais informações são necessárias e seus aspectos estruturais;
- diagrama de sequência – modela os diferentes cenários de ativação do contexto e o fluxo de informação entre os sistemas envolvidos.

A modelagem de domínio com ORM é baseada em fatos sobre os quais os vários tipos de entidades desempenham algum papel. Em (Henricksen et al., 2002) é apresentada

uma extensão da ORM para modelagem de contexto em dois níveis de abstração: fatos, nível mais baixo, e situações, nível mais alto de abstração.

Modelo baseado em lógica: com alto nível de formalismo, os modelos baseados em lógica são definidos por fatos, expressões e regras manipulados por um sistema de inferência ou raciocínio. Nesses modelos, as informações contextuais são fatos e as regras podem ser utilizadas para derivar novos fatos.

Modelo baseado em mapas de tópicos: mapas de tópicos são definições das relações entre objetos físicos e lógicos localizados dentro ou fora do mundo computacional, pela abordagem de redes semânticas, que facilita a navegação pelos dados organizando os objetos de forma intuitiva (Vieira, 2006). Tópicos são representações de objetos do mundo real que se relacionam por associações, ligados aos respectivos objetos reais por ocorrências (informações relevantes sobre os tópicos) e visíveis dentro de um escopo ou área semântica definida no mapa. Embora existam extensões desse modelo para representação de contexto, Vieira (2006) afirma que seu uso não foi demonstrado em aplicações práticas, necessitando de tecnologias mais maduras para a criação e manipulação de mapas de tópicos.

Modelo baseado em ontologia: com alto nível de formalismo, os modelos baseados em ontologia descrevem conceitos e relacionamentos, possibilitando a aplicação de técnicas de raciocínio oferecidas por linguagens baseada em lógica descritiva, como a OWL. Além da expressividade, esses modelos também apresentam simplicidade, flexibilidade e genericidade, sendo os mais investigados na literatura para modelar contexto independentemente da aplicação. Outra vantagem é permitir o compartilhamento e reutilização das informações de contexto.

Modelo baseado em grafos contextuais: grafo contextual é um formalismo apresentado em (Brézillon, 2003), que consiste na representação baseada em contexto da execução de uma tarefa. Baseia-se nos conceitos de procedimento e prática em que um procedimento corresponde a uma ação a ser realizada e a prática é a contextualização do procedimento, adaptado a uma determinada situação (como os procedimentos são efetivamente realizados pelos operadores, em sua rotina diária). Grafos contextuais são adaptações de árvores de contexto, acíclicos direcionados, contendo uma única entrada, uma única saída, com nós organizados de forma serial-paralela, conectados por arcos orientados. Também é considerado um modelo baseado em representação gráfica e, de acordo com Vieira (2006), concentra-se em

problemas com as características de procedimento e prática ou quando envolve algum processo.

A Tabela 2.1, extraída de (Vieira, 2006) e (Vieira, 2008), resume as principais características das técnicas apresentadas.

Tabela 2.1: Técnicas para representação de contexto (Vieira, 2006, 2008).

Técnica	**Vantagens**	**Desvantagens**	**Processamento/ Recuperação**
Valor-chave	Estrutura simples, facilidades de implementação e uso	Não considera hierarquia, inadequado para estruturas complexas	Busca linear, exigindo exatidão nas strings de nomes
Marcação	Considera hierarquia, facilita o compartilhamento	Não trata ambiguidades e inconsistências, inadequado para estruturas complexas	Linguagem de consulta baseada em XML
Representações gráficas	Facilita a especificação do contexto e o projeto do sistema, além de facilitar a compreensão por humanos	Não utilizado para instanciar a informação	Transformação dos modelos para outro formato (ex. XML)
Lógica	Alto grau de formalismo	Difícil modelagem e compreensão por humanos	Motor de inferência
Mapas de tópicos	Facilita a navegação entre conceitos e a modelagem por humanos	Ferramentas imaturas e falta de formalismo	Navegação por redes semânticas
Ontologia	Aumenta a definição semântica do contexto, permite o uso de ferramentas de inferência, possibilita a compreensão por humanos e computadores	Tecnologia para manipulação imatura, diminui a performance do sistema	Motores de inferência, linguagens baseadas em consultas, frames ou OWL
Grafos contextuais	Adequados para modelar contextos associados a processos	Não utilizado para instanciar a informação	Busca em árvore

Pares valor-chave e linguagens de marcação são fáceis de usar, entretanto, se tornam difíceis de manter e evoluir quando utilizadas em aplicações complexas. Mapas de tópicos são flexíveis, mas não possuem ferramentas de apoio, dificultando sua utilização. Ontologias têm seu foco em formalização e raciocínio, mas apenas modelam a estrutura do modelo de contexto, não definindo seu comportamento. Os modelos gráficos facilitam a visualização das relações existentes entre as informações contextuais, mas precisam ser traduzidos para uma estrutura mais expressiva (ontologias ou modelos orientados a objetos) para serem instanciados e manipulados (Vieira, 2008).

Grafos contextuais são modelos gráficos capazes de modelar o comportamento de um sistema, com base na dinâmica do contexto. Por ser utilizado neste trabalho, os grafos contextuais serão apresentados com mais detalhes na seção a seguir.

2.5.1 Grafos Contextuais

Segundo Brézillon (2003), os Grafos Contextuais (CxG)2 são grafos acíclicos e dirigidos, com exatamente uma entrada e uma saída, já que possuem um único objetivo (realizar o

procedimento) e diversos caminhos que representam as diferentes estratégias dependentes do contexto (práticas) para alcançar esse objetivo. O tamanho da estrutura é facilmente controlado, pois considerar um novo elemento contextual implica em acrescentar elementos no grafo, sem aumentar seu tamanho drasticamente. Além disso, introduzem um modelo dinâmico, comparável ao modelo dinâmico de mudança entre contexto procedural e conhecimento contextual.

Os principais elementos de um grafo contextual são (Brézillon, 2005):

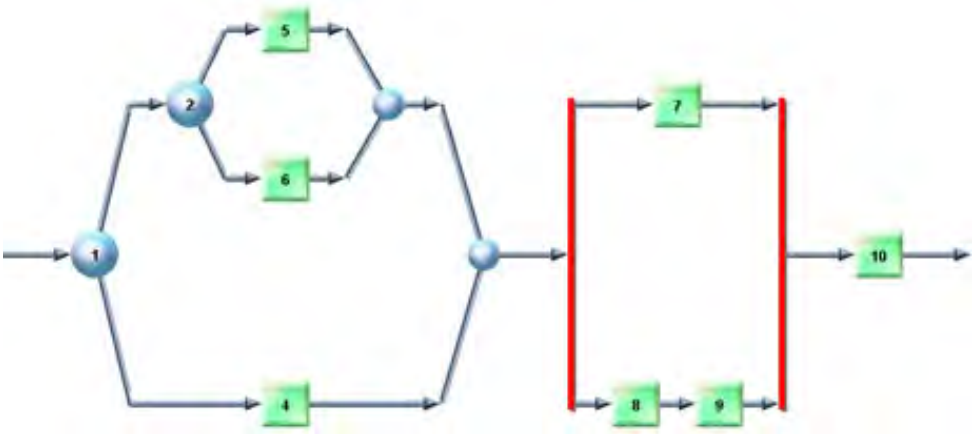

Figura 2.3: Exemplo de grafo contextual (Brézillon, 2007)

***Action*:** representado pelos retângulos enumerados (como pode ser visto na Figura 2.3), indica uma ação (método) a ser executada. Uma ação pode aparecer em diversos caminhos, sendo considerada, a cada ocorrência, como uma instância da ação em um contexto específico.

***Contextual node* e *Recombination node*:** o *contextual node* é uma decisão dependente de contexto. Possui uma única entrada e N saídas (*branches*). Cada *branch* representa uma prática que pode ser realizada de acordo com o contexto. Todas as saídas convergem para um *recombination node*, que possui as N entradas e uma única saída. Esta saída representa a direção a ser seguida depois de verificada a condição do *contextual node*. Na Figura 2.3, os *contextual nodes* são os círculos enumerados e os *recombination nodes* são os círculos pequenos não numerados.

***Sub-graph*:** é um grafo propriamente dito. Entretanto, é uma subdivisão de um grafo complexo. Um grafo contextual pode ser estruturado em subgrafos, diminuindo a complexidade e possibilitando diferentes visões do mesmo.

***Activity*:** é uma ação complexa, que executa um subgrafo. Uma mudança em uma atividade aparece em todos os grafos contextuais onde a atividade foi identificada.

***Paralel action grouping*:** representam grupos de ações que devem ser realizadas cuja ordem de execução não é importante, ou devem acontecer em paralelo, sendo representados pelas barras paralelas na Figura 2.3.

De acordo com Vieira (2008), os grafos contextuais são úteis para modelar o comportamento de sistemas sensíveis ao contexto por descrever as ações que precisam ser executadas pela aplicação e, explicitamente, indicar como o contexto influencia essas ações.

2.6 Context Metamodel

Analisando os diversos modelos de contexto existentes, Vieira (2008) observou que, em geral, não havia uma preocupação em formalizar os conceitos utilizados para descrever o domínio, bem como o seu relacionamento com a dinâmica do contexto. Dessa forma, esses modelos não relacionavam a informação contextual com o seu uso. Com base nessas observações, Vieira (2008) propôs um metamodelo de contexto (*Context Metamodel*) independente de domínio para apoiar a criação de modelos contextuais. Esse metamodelo é uma extensão do UML 2.0 *Metamodel*[2], seguindo a semântica e a notação padrões da UML.

Os objetivos do *Context Metamodel* são (Vieira, 2008):

- oferecer um *framework* conceitual que identifique os principais conceitos relacionados ao uso e manipulação de contexto, independentemente de domínio; e
- apoiar o reuso e extensão dos modelos de contexto existentes ou modelos de aplicação.

O *Context Metamodel* organiza os conceitos em duas categorias: modelos estruturais, que descrevem os conceitos relacionados com os elementos conceituais e estruturais do sistema; e os modelos comportamentais, que contém os conceitos relacionados aos aspectos comportamentais do sistema.

Os conceitos estruturais do *Context Metamodel* são apresentados na Figura 2.4. Com relação a esses conceitos, os principais são descritos a seguir:

***ContextualEntity*:** (entidade contextual) representa as entidades que devem ser consideradas para a manipulação do contexto. Uma *ContextualEntity* se caracteriza por

[2]Unified Modeling Language: Superstructure, Version 2.1.2, In: http://www.omg.org/spec/UML/2.1.2/

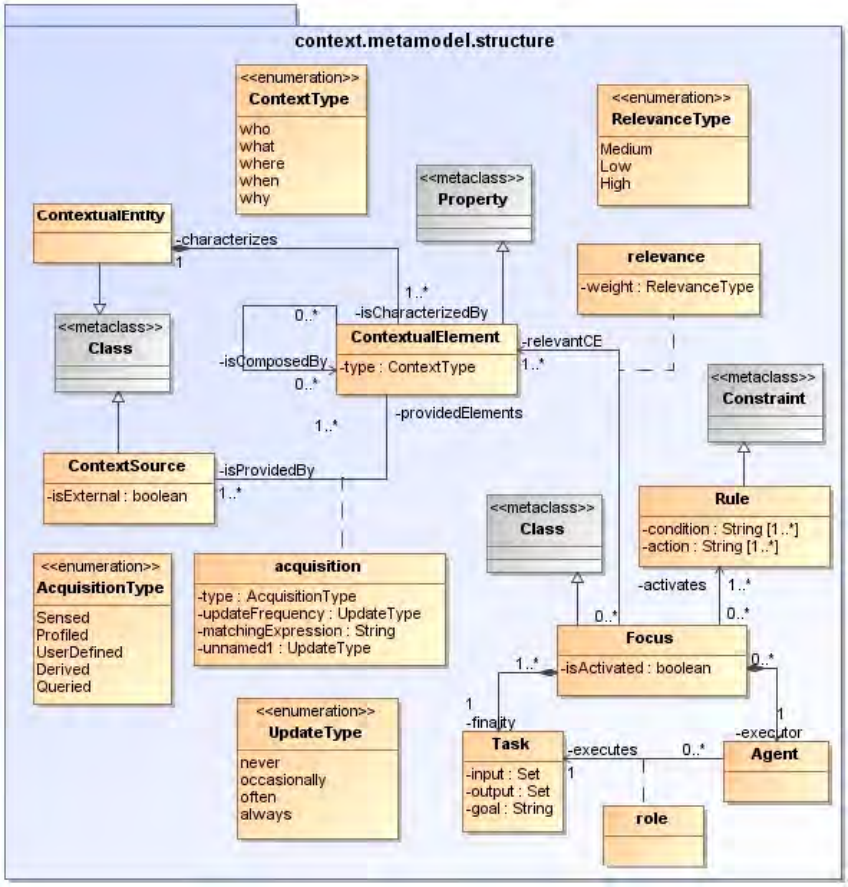

Figura 2.4: Conceitos estruturais do *Context Metamodel*. (Vieira, 2008)

possuir pelo menos um *ContextualElement*. Por entidade, entende-se uma representação concreta de um objeto do mundo real que pode ser identificado unicamente e que é relevante para descrever um domínio;

***ContextualElement* (CE):** (elemento contextual) representa uma propriedade utilizada para caracterizar uma entidade contextual. Pode ser identificado a partir de um conjunto de atributos e relacionamentos associados a uma entidade. Uma propriedade é uma relação binária entre duas entidades ou uma entidade e um valor de dado. É importante observar que nem todas as propriedades de uma entidade são CEs. O critério para identificar se uma propriedade é um CE é subjetivo e dependente dos requisitos do contexto definidos pelo projetista de contexto para o sistema;

***ContextSource*:** (fonte de contexto) identifica as diversas fontes heterogêneas que oferecem informações correspondentes aos elementos contextuais (CEs). Uma fonte de contexto possui uma relação com um CE por uma associação do tipo *Acquisition*

(aquisição). Essa associação indica e parametriza o relacionamento entre um CE e um *ContextSource*, especificando o tipo de aquisição (*Type*), a frequência de atualização (*updateFrequency*) e, se necessário, uma função de transformação para derivar informações a partir dos dados oriundos da fonte de contexto (*matchingExpression*);

***Focus*:** (foco) permite determinar qual CE deve ser instanciado e utilizado para compor o contexto. O *Focus* é definido como a composição de uma Tarefa (*Task*) e um Agente (*Agent*), em que o Agente executa a Tarefa desempenhando um papel;

***Rule*:** (regra) representa um conjunto de uma ou mais condições (*conditions*) e um conjunto de uma ou mais ações (*actions*), em que cada condição representa uma expressão que retorna um valor verdadeiro, falso ou desconhecido quando os dados estão disponíveis e cada ação representa um procedimento que deve ser executado quando uma condição da regra for satisfeita.

Quanto aos conceitos comportamentais, o *Context Metamodel* utiliza os conceitos definidos no formalismo dos grafos contextuais (Brézillon, 2003), apresentados na Seção 2.5.1: *contextual node*, *action*, *recombination node*, *activity* e *parallel action grouping*. O metamodelo segue a semântica e as restrições estabelecida por este formalismo.

A Figura 2.5 apresenta, graficamente, os conceitos e restrições dos grafos contextuais, utilizando um Diagrama de Classes UML.

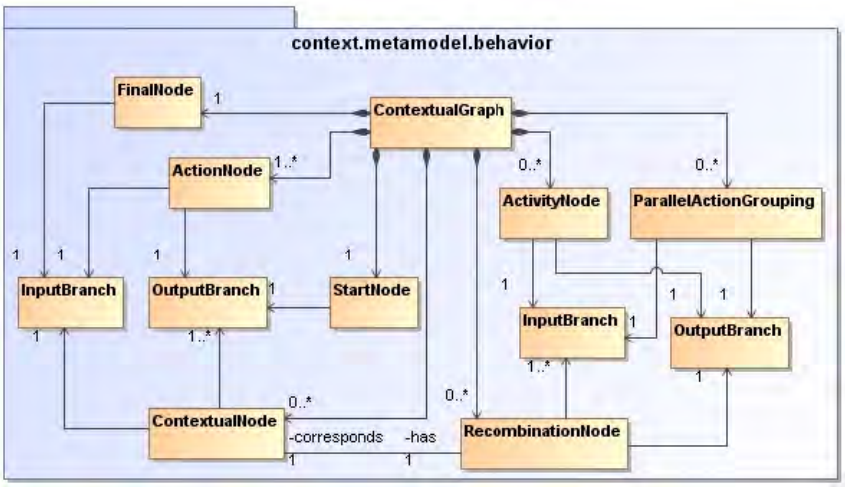

Figura 2.5: Conceitos comportamentais do *Context Metamodel*. (Vieira, 2008)

O *Context Metamodel* foi utilizado para modelar as informações contextuais definidas neste trabalho por ser, entre as pesquisas realizadas, o único metamodelo independente

de domínio encontrado na literatura capaz de expressar graficamente a relação existente entre os elementos contextuais e a sua aplicação em um contexto dinâmico.

2.7 Considerações Finais

O objetivo deste capítulo foi apresentar os principais conceitos utilizados no desenvolvimento deste trabalho. Depois de compreendidos esses conceitos, pesquisas foram realizadas para identificar trabalhos relacionados a esses assuntos, nas diversas áreas de pesquisa. O Capítulo 3 discute os trabalhos encontrados e sua relevância para o desesnvolvimento do modelo DiSEN-CSE.

Capítulo

3

Trabalhos Relacionados

Nos últimos anos, muitos trabalhos foram desenvolvidos no âmbito de sistemas sensíveis ao contexto, nas mais diversas linhas de pesquisa, com objetivos que variam de acordo com o domínio das aplicações. Neste capítulo serão apresentadas as pesquisas que estimularam ou influenciaram o desenvolvimento deste trabalho.

3.1 Compartilhamento de Informações

Conforme discutido na Seção 2.1, quando se trata de GSD um dos fatores de maior relevância para o sucesso dos projetos é a comunicação. De acordo com Herbsleb (2007), pouca comunicação e comunicação pouco eficiente, falta de percepção sobre as informações contextuais e as incompatibilidades de ferramentas, processos, entre outros, estão entre as dificuldades mais comuns dos mecanismos de coordenação do trabalho distribuído.

Para tentar minimizar o problema da falta de comunicação, trabalhos foram propostos para garantir o compartilhamento de informações para usuários interessados. Pesquisas na área de CSCW (*Computer Supported Cooperative Work*) foram desenvolvidas para aumentar a percepção de indivíduos participantes do trabalho cooperativo e contextualizar suas ações, oferecendo relevância às suas contribuições individuais e ao significado dessas contribuições para o grupo como um todo (Pinheiro et al., 2001).

Com o objetivo de facilitar a percepção de indivíduos que trabalham cooperativamente, Pinheiro et al. (2001) propuseram um *framework* para *awareness* em ferramentas de

groupware que classifica a percepção de acordo com seis questões, considerando ambientes síncronos ou assíncronos:

o que: se atenta para quais informações devem ser disponibilizadas para os usuários considerando atividades e papéis;

quando: especifica em que momento as informações devem ser exibidas, sendo dividida em eventos – momento em que a informação é gerada, apresentação – momento em que a informação deve se tornar disponível, e tempo de utilidade ou persistência – por quanto tempo as informações precisam estar disponíveis;

onde: indica o espaço de trabalho em que as informações devem ser geradas e apresentadas;

como: aponta para a forma como as informações serão apresentadas na interface do usuário;

quem: detecta a presença de indivíduos que participam do trabalho ou estão atentos a ele;

quanto: identifica qual quantidade de informação é suficiente para os indivíduos, evitando tanto a sobrecarga quanto a omissão.

A proposta de Pinheiro et al. (2001) se concentra na maneira como as informações serão compartilhadas entre os participantes do trabalho cooperativo. Entretanto, o *framework* refere-se estritamente à questão da percepção sobre o objeto de cooperação, não se preocupando com a forma como foi produzido.

Oferecer percepção a uma determinada atividade do trabalho cooperativo depende da relação existente entre percepção e contexto, como destacado na Seção 2.4. Entretanto, essa relação nem sempre é clara. As informações de contexto, muitas vezes, não são consideradas e, quando são, normalmente ocorre de forma *ad hoc* e intuitiva.

Devido à importância do conhecimento a respeito do contexto para apoiar as interações dos grupos de trabalho, um *framework* conceitual foi proposto por Rosa (2004) para identificar e classificar as informações contextuais mais comuns em ferramentas de *groupware*. O *framework* divide as informações em cinco tipos:

Pessoas: dividido em indivíduos e grupos, corresponde às informações que os caracterizam e identificam dentro de um determinado ambiente;

Tarefas: corresponde às características das tarefas, que devem ser identificadas no momento em que estas são definidas.

Relação entre pessoas e tarefas: refere-se às ações que ocorrem durante a execução das tarefas e o plano de ação, indicando as responsabilidades de cada indivíduo ou grupo;

Ambiente: identifica as informações que caracterizam o ambiente em que as ações ocorrem e que podem influenciar de alguma forma na execução;

Tarefas realizadas: equivalente a um histórico de tarefas realizadas, apontam para os motivos que levaram ao seu sucesso ou fracasso, servindo como subsídios aos grupos para tarefas futuras.

Com base nessas informações, esse *framework* busca manter informações importantes para que os indivíduos que utilizam sistemas *groupware* percebam e compreendam a influência de suas interações. Todavia, não define como as informações contextuais serão apresentadas aos indivíduos do grupo.

Embora possuam focos diferentes, os dois trabalhos são altamente relacionados. A identificação e classificação das informações contextuais podem ser apoiadas pelo *framework* de Rosa (2004), que sugere alguns elementos de contexto genéricos, relevantes a diversos domínios de aplicação, além de permitir que esses elementos sejam adaptados, dentro da classificação sugerida, para ser mais específico a um domínio determinado. A proposta de Pinheiro et al. (2001) está focada na maneira como essas informações serão exibidas para os participantes do trabalho cooperativo, de forma a aumentar a percepção do indivíduo, considerando interações síncronas ou assíncronas. Além disso, o *framework* de Pinheiro et al. (2001) está intrínseco ao *framework* de Rosa (2004), devido à relação existente entre percepção e contexto. A Tabela 3.1 objetiva ressaltar o relacionamento entre esses dois *frameworks*.

Algumas observações podem ser feitas com relação à Tabela 3.1. Em primeiro lugar, ela não aborda as questões como e quanto. Isso porque essas questões têm origem na percepção, muito mais que no contexto. No *framework* de Pinheiro et al. (2001), o "como" está relacionado à maneira como as informações são exibidas e não como elas são realizadas. A questão "como" referente à realização das tarefas ou atividades está dissolvida entre os elementos contextuais presentes no *framework* conceitual de Rosa (2004), nas mensagens trocadas durante as interações ou nas políticas estabelecidas no contexto do ambiente, por exemplo. A questão "quanto", entretanto, não é mapeada para o *framework* conceitual.

Tabela 3.1: Intersecção entre percepção e contexto, baseado nos *frameworks* apresentados em (Pinheiro et al., 2001) e (Rosa, 2004).

		O que	**Quando**	**Onde**	**Quem**
Pessoas	**Indivíduo**	Quais tarefas cada indivíduo está executando ou é responsável.	Relacionado à agenda dos indivíduos, períodos em que está envolvido com o projeto. Ex: horário que conecta no ambiente ou que disponibiliza um artefato.	Local em que o indivíduo está conectado, local a que ele pertence.	Informações pessoais de cada indivíduo e suas afinidades com outros indivíduos do sistema.
	Grupo	Relacionado aos papéis. É preciso identificar que papéis os indivíduos exercem nas equipes para saber que informações precisam receber.	Horas em que o grupo trabalha em uma determinada tarefa, Quantidade de horas trabalhadas pelo grupo.	Sede geográfica do grupo, ou local sob o qual a equipe está sendo regida, fuso horário, legislação vigente, etc.	Quem são os participantes de um determinado grupo de trabalho, com base nas experiências, nos conhecimentos e nas afinidades.
Tarefas a executar	**Tarefas**	Representa todas as informações sobre uma determinada tarefa, seus estados e artefatos produzidos.	-	-	Que participantes ou grupos realizam as tarefas, seus conhecimentos e suas afinidades.
Pessoas x Tarefas	**Contexto de interação**	O que foi realizado durante as interações. Caracterizam a realização da tarefa. Em ambientes síncronos, é importante a noção de presença e saber que mensagens foram trocadas. Em ambientes assíncronos, que versão de artefato foi produzida, p.ex.	Momento em que as tarefas iniciam, em que as versões são disponibilizadas, ou até mesmo quando os usuários estão efetivamente trabalhando nelas.	-	Que indivíduos estão interagindo. Em ambientes síncronos, deve haver uma forma de identificar, p.ex., a quem pertence o mouse no caso de um teleapontador, ou quem está digitando em um editor cooperativo. Em ambientes síncronos, é preciso saber quem realizou a ação.
	Contexto do planejamento	Plano de execução da tarefa, contendo o que deve ser feito para que a tarefa seja realizada: prazos, metas, estratégias, etc.	Relacionado a prazos das tarefas. Previsão de quando a atividade deve ser executada, se está atrasada, se o prazo precisa ser revisto.	-	Quem planejou as ações e quem está apto a realizar cada atividade de acordo com os grupos, conhecimentos e afinidades.
Informações sobre o ambiente	**Contexto do ambiente**	Quais os padrões que devem ser seguidos para a realização da tarefa.	Relacionado aos prazos. Devem respeitar os limites impostos pelo contexto do ambiente.	Culturas, políticas, leis de propriedade intelectual, estrutura da organização.	-
Tarefas concluídas	**Contexto histórico**	"O quê" passado. Informações sobre tarefas já realizadas devem estar disponíveis para automatização de tarefas ou para auxiliar decisões futuras.	Quando cada ação relacionada àquela tarefa, atividade ou projeto foi realizada.	-	Quem realizou cada ação. É importante para armazenar a experiência de cada componente do grupo, bem como a experiência do grupo como um todo.

Em segundo lugar, as questões abordadas sugerem quatro tipos diferentes de percepção: percepção de tarefa (o que), reportando-se aos elementos de contexto que especificam as ações; a percepção temporal (quando), especialmente importante em ambientes assíncronos, já que em ambientes síncronos o "quando" referencia o momento presente; percepção de localidade (onde) que pode fazer referência tanto a sedes geográficas como ao local no qual o indivíduo está trabalhando, como um artefato compartilhado ou o seu próprio *desktop*; e, por fim, a percepção de identidade (quem), que referencia não apenas informações pessoais como um catálogo de endereços, mas o que um determinado indivíduo representa dentro do grupo ou para os outros indivíduos que se relacionam com ele.

Essa classificação de tipos de percepção pode ser encontrada em (Brooks, 2003). Além das questões abordadas, o autor também identifica a importância de se responder à questão "por quê", para oferecer contexto sobre o porquê da realização de determinadas ações, de uma determinada maneira, criando um histórico de tomadas de decisões. Essa questão, entretanto, ainda é um desafio, já que envolve diretamente a expressão de pensamentos e raciocínios próprios de um indivíduo, que muitas vezes não consegue exprimi-los de maneira clara e não ambígua.

Além de obter as informações contextuais relevantes e possuir mecanismos de percepção para apoiar seu compartilhamento, outro aspecto importante para o GSD é garantir que essas informações serão recebidas pelo usuário certo, no momento certo. De maneira geral, os trabalhos encontrados na literatura focam em notificações baseadas em eventos para realizar esse tipo de compartilhamento.

Para um ambiente de ensino e aprendizagem a distância, conhecido como AulaNet[1], Filippo et al. (2008) desenvolveram um serviço de notificações, chamado ANC (*AulaNet Companion*) com o objetivo de informar os participantes de um curso a distância sobre as últimas movimentações no ambiente, mesmo que estes não estejam conectados. O ANC é um serviço cliente/servidor, iniciado no momento do *logon* do sistema operacional Windows. Todas notificações são apresentadas na mesma interface gráfica, independente das preferências ou perfis dos usuários. A apresentação ocorre em janelas *pop-up* durante dez segundos, e contém a quantidade de participantes que se conectaram no curso ou acessaram um determinado serviço no minuto anterior. A janela pode conter *links* e botões, que permitem que o participante acesse o sistema a partir das notificações. As notificações do AulaNet não levam em conta o contexto dos participantes, enviando a mensagem a todos os indivíduos que participam do curso. Além disso, possui um conjunto restrito e bem definido de tipos de eventos que podem ser notificados. Assim, o

[1] www.eduweb.com.br – groupware.les.inf.puc-rio.br

ANC corresponde a uma maneira de aumentar a coordenação entre os indivíduos, apenas informando-os igualmente sobre determinadas ações realizadas no ambiente.

Mais especificamente para apoiar o desenvolvimento distribuído de software, Kobylinski et al. (2002) propuseram um sistema de percepção em que os participantes podem se subscrever para serem notificados quando algumas ações ocorrem no ambiente de trabalho virtual. Essas ações são o início de uma determinada atividade, o início de uma atividade relacionada a um determinado assunto ou quando um determinado artefato do sistema é modificado. Após se registrarem, quando o evento acontece, uma mensagem é disparada para todos os subscritos. A mensagem de evento inclui quem originou a ação, a ferramenta utilizada, o artefato manipulado e o comando da ferramenta utilizado para modificar o artefato. Quando os receptores não estão disponíveis, as notificações são persistidas e entregues posteriormente. Esse sistema, assim com o AulaNet, também possui um conjunto restrito e bem definido de tipos de eventos que podem ser notificados. Todas as mensagens são exibidas em uma interface padrão, sem considerar o perfil e as preferências dos usuários. Além disso, não considera o contexto para decidir quem deve receber as mensagens em um dado momento, exigindo que os usuários se subscrevam, aumentando o esforço necessário para sua utilização.

Uma ferramenta de notificação, chamada NOTICON (*Notification In Context*), foi desenvolvida por Eckhard (2007) para um ambiente GSD típico. Essa ferramenta permite que membros dos projetos especifiquem seus requisitos de notificações, levando em conta seu contexto atual. NOTICON interpreta uma linguagem denominada NSL (*Notification Specification Language*), que permite a modelagem precisa e amigável ao usuário dos requisitos de notificação. O objetivo é maximizar o número de notificações relevantes entregues aos usuários e diminuir as irrelevantes. Além disso, NOTICON pode ser integrada ao ambiente de trabalho, apresentando as notificações na ferramenta que o usuário está utilizando, reduzindo, assim, as interrupções causadas pelas notificações.

A ferramenta NOTICON considera o contexto dos indivíduos à medida que permite filtrar as notificações de acordo com sua relevância aos usuários, analisando as dependências das atividades e artefatos, reduzindo a necessidade de se subscrever manualmente para os eventos que devem ser de seu interesse. Além disso, o conteúdo da mensagem pode ser adaptado, de acordo com o contexto do projeto. Por outro lado, a ferramenta não permite o armazenamento de notificações, impossibilitando os usuários de realizar consultas sobre eventos passados. Outra dificuldade em sua utilização é a necessidade de que um usuário especialista defina as notificações utilizando a linguagem NSL, para serem interpretadas.

3.2 Gerenciamento de contexto

Nas áreas de computação ubíqua, pervasiva e móvel, cujas pesquisas se encontram em um nível mais avançado de implementação e aplicação, diversas arquiteturas e infraestruturas foram propostas para auxiliar o gerenciamento de contexto, com o intuito de capturar as informações contextuais por utilização de sensores, representá-las, armazená-las e compartilhá-las entre as entidades envolvidas. Dentre esses trabalhos, citam-se:

***Context Managing Framework* (Korpipää et al., 2003):** o contexto semântico é reconhecido em tempo real, considerando as mudanças rápidas de informações e incertezas e a disseminação do contexto para aplicações móveis com um gerenciador baseado em eventos. Utiliza uma abordagem baseada em *blackboard* para comunicação entre as entidades do *framework* e um modelo baseado em ontologias para representação do contexto;

MoCA (Filho et al., 2006): arquitetura para oferecer recursos para o desenvolvimento e execução de aplicações *context-aware* cooperativas, envolvendo usuários de dispositivos móveis. Entre outros componentes, o MoCA possui um sensor que coleta as informações de contexto, um componente que armazena e processa os dados, tornando-as disponíveis para outras aplicações, e um componente responsável por oferecer a localização dos dispositivos móveis, além de um serviço de controle de privacidade;

LOTUS (Bublitz, 2006): *middleware* baseado em *plugins* para o desenvolvimento de aplicações em ambientes pervasivos com apoio à obtenção, representação e inferência de informações de contexto, tornando-as disponíveis para os diversos domínios de aplicação. Possui três camadas: (i) aquisição, que possui os vários tipos de sensores, (ii) representação, que possui um modelo baseado em ontologias e (iii) raciocínio, contendo os módulos de inferência sobre as informações do ambiente;

SOCAM (Gu et al., 2005): arquitetura distribuída e baseada em *middleware*, oferece uma infraestrutura eficiente para apoiar a construção de serviços sensíveis ao contexto em ambientes de computação pervasiva. As informações são abstraídas de diversas fontes e convertidas para um modelo baseado em ontologias, podendo ser processadas por um interpretador e armazenadas em bases lógicas, específicas para cada domínio. Além disso, possui serviços para adaptar o comportamento de acordo com o contexto e localização de serviços remotos;

CoBrA (Chen, 2004): arquitetura baseada em agentes para apoiar sistemas sensíveis ao contexto em ambientes inteligentes. Um agente inteligente oferece meios para centralizar o contexto que dispositivos, serviços e agentes podem compartilhar. A estrutura do conhecimento é descrita por metadados e ontologias e as informações podem ser adquiridas por sensores, agentes e pela Web. Além disso, possui um componente para gerenciar políticas de privacidade;

Gaia (Ranganathan e Campbell, 2003): é uma infraestrutura para ambientes computacionais ubíquos que tem como objetivo tornar inteligentes espaços como salas, casas e aeroportos, para auxiliar as pessoas que agem neles. Possui diferentes tipos de agentes que executam diferentes tarefas, podendo estar distribuídos, utilizando CORBA para a comunicação.

Os mecanismos citados acima, embora possuam arquiteturas distintas, são semelhantes em diversos pontos. Primeiramente, todos eles utilizam representação de contexto baseado em ontologias. Foram encontradas diversas arquiteturas que não utilizam esse modelo de representação, entre elas *Context Toolkit* (Dey et al., 2001), que utiliza pares valor-chave; SOPHIE (Belotti, 2004), que utiliza gráficos ORM; CORTEX (Biegel e Cahill, 2004), utilizando DER; Hydrogen (Hofer et al., 2002), com modelos baseados em objetos. Entretanto, a representação baseada em ontologias tem se consolidado na grande maioria dos trabalhos recentes encontrados na literatura.

Além da representação, é possível observar que, em geral, as arquiteturas se preocupam com a forma com que as informações de contexto serão adquiridas, quanto é possível inferir sobre essas informações automaticamente, de que maneira esse contexto será disseminado e como poderá ser persistido para manter um contexto histórico.

Outro ponto importante é que, por pertencerem ao domínio da computação ubíqua ou pervasiva, os contextos abordados são amplamente baseados em sensores físicos, mais relevantes para esse tipo de ambientes.

Mais especificamente, quando se trata de gestão de conhecimento, um trabalho foi proposto em (Nunes et al., 2007) para oferecer um modelo que define como tratar o ciclo de criação, armazenamento e reutilização contextual. O modelo foca em três aspectos relevantes, que são: (i) representação da informação de contexto, que utiliza uma ontologia para definir quais as informações de contexto relevantes para uma determinada situação, (ii) captura e armazenamento, que coleta as informações de contexto de acordo com o modelo de representação e as armazena, relacionando-as às atividades, para alimentar a memória organizacional e (iii) recuperação e apresentação do contexto, que recupera os contexto existentes, relacionados a uma determinada atividade, utilizando mecanismos de

inferência capazes de decidir a relevância da informação para o usuário, apresentando-as por mecanismos de percepção. Esse modelo concentra-se na construção de uma memória organizacional que torne os indivíduos cientes do contexto em que determinadas atividades ocorreram no passado, permitindo que, quando necessário, as pessoas envolvidas nas atividades possam recuperar informações pelos mecanismos de percepção. Por essa razão, não possui mecanismos de disseminação automática da informação que ofereça meios pelos quais as pessoas sejam comunicadas a respeito de mudanças ocorridas nesta base de informações.

3.3 Considerações Finais

Durante os últimos anos, diversas pesquisas nas mais variadas áreas de conhecimento, tem se preocupado em tornar os sistemas sensíveis ao contexto, seja para aumentar a percepção, facilitar a comunicação ou realizar ações automaticamente para o usuário.

Dentre essas pesquisas, este trabalho destaca aquelas relacionada à computação ubíqua, pervasiva e móvel, que possuem maiores resultados implementados e aplicados, os que tratam compartilhamento de informação no trabalho cooperativo e que tentam, de alguma maneira, diminuir as barreiras de comunicação entre indivíduos geograficamente dispersos.

Além desses, pesquisadores das áreas de IHC (Interface Humano-Computador) e IA (Inteligência Artificial) propuseram trabalhos especialmente interessados na apresentação personalizada de informações para o usuário e na formalização/representação do contexto, como pode ser visto em (Vieira, 2008).

Baseado nos problemas existentes no GSD a respeito de comunicação e coordenação de atividades e nas observações feitas com relação aos trabalhos descritos neste capítulo, foi possível elaborar um modelo baseado em *context-awareness* para disseminação de informações em ambientes de desenvolvimento distribuído de software, que será apresentado nos próximos capítulos.

Capítulo

4

Modelo DiSEN-CSE

4.1 Considerações Iniciais

De acordo com Pozza (2005), o desenvolvimento distribuído de software se torna mais produtivo se existe um ambiente estruturado para oferecer recursos (ferramentas, técnicas e metodologias) que auxiliem na execução de atividades, minimizando as dificuldades causadas pela distância física. Além disso, é necessário que os indivíduos troquem informações, organizem-se e operem em conjunto. Para tanto, os indivíduos precisam estar cientes das atividades dos demais, para poder adquirir informações sobre o seu próprio trabalho (Fuks et al., 2003).

O DiSEN (*Distributed Software Engineering Environment*) é um ADDS (Ambiente de Desenvolvimento Distribuído de Software) que visa oferecer uma infraestrutura para auxiliar o desenvolvimento distribuído de software. Sua arquitetura inicial foi proposta em (Pascutti, 2002) e, posteriormente, um *framework*, denominado FRADE (*Framework to Infrastructure of Distributed Software Development Environment*), foi definido em (Schiavoni, 2007) para especificar a infraestrutura que capacita o ambiente a gerenciar a comunicação entre os diversos participantes.

O ambiente DiSEN possui três gerenciadores (Pascutti, 2002): (i) Gerenciador de Objetos – adaptado em (Schiavoni, 2007), compreende o conjunto de entidades do ambiente, envolvendo os Gerenciadores de Artefatos, de Processos, de Recursos, de Projetos, de Ferramentas, de Usuários e de Locais; (ii) Gerenciador de Agentes – responsável pela criação, registro, localização, migração e destruição de agentes de software; e (iii) Gerenciador de

Workspace – detalhado em (Pozza, 2005), fornece *workspaces* compartilhados para que os usuários possam cooperar no desenvolvimento de suas atividades.

Para gerenciar a interação entre os *workspaces* compartilhados, Pozza (2005) propôs um modelo que oferece as características de Sincronização, Percepção e Comunicação (SPC) para o Gerenciador de *Workspace* (Figura 4.1). Neste modelo, a Comunicação recebe e envia os dados das tarefas em execução, via canal de comunicação, para os diversos *workspaces* envolvidos no trabalho cooperativo. Quando ocorre uma interação, os dados são enviados para a Percepção, que os coordena com a tarefa em execução (respondendo às questões "o que", "quando", "onde", "como", "quem" e "quanto", como visto no Capítulo 3). A Sincronização garante que os *workspaces* possam trabalhar concorrentemente de forma coesa.

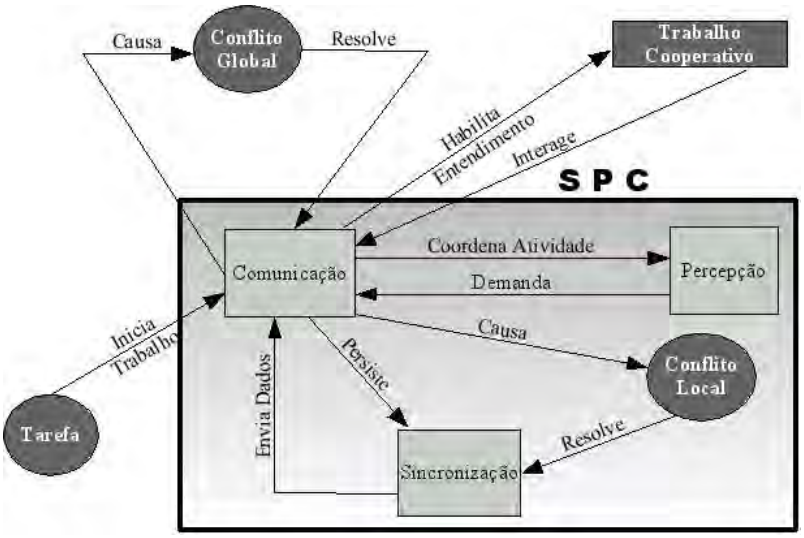

Figura 4.1: Modelo SPC (Pozza, 2005)

Entretanto, em ambientes distribuídos, a distância física evidencia as dificuldades de oferecer percepção. Araujo e Brézillon (2005) afirmam que, em um ambiente cooperativo, além de organizar, armazenar e reutilizar os conhecimentos disponíveis, fatores como a facilidade de contato, a comunicação, a compreensão mútua e o compartilhamento são necessários.

Analisando os problemas discutidos na Seção 2.1, no âmbito do DDS, se as informações disseminadas pelo ambiente forem contextualizadas e possuírem uma representação única, reconhecida pelos participantes, então os indivíduos terão um entendimento comum, claro e não ambíguo das informações geradas pelas interações.

Para realizar suas tarefas, as pessoas dependem da disponibilidade de recursos para completá-las. Assim, contextualizar as atividades no DiSEN envolve não apenas saber

em quais tarefas de um projeto os indivíduos estão trabalhando e como são realizadas, mas também conhecer as ferramentas e serviços que o ambiente deve oferecer durante as interações.

Conforme definido em (Pascutti, 2002; Schiavoni, 2007), o Gerenciador de Objetos do DiSEN possui um Gerenciador de Recursos. Além dos recursos materiais como equipamentos, papéis, mídias, entre outros, consideram-se como recursos Locais, Usuários e Ferramentas, que se constituem nas três entidades necessárias e capazes de gerar informações para o ambiente que precisam ser contextualizadas. Os usuários, que podem ser uma única pessoa ou um grupo, são as entidades mais ativas do ambiente que, por meio de ferramentas e dos serviços disponíveis, alteram o estado e geram informações para o ambiente como um todo.

Para que as instâncias do ambiente tornem-se sensíveis ao contexto, é necessário que conheçam e compreendam as ações realizadas por cada uma dessas entidades, o que é particularmente difícil em ambientes distribuídos. Assim, para compartilhar informações contextuais em DDS, é necessário transformá-las de forma que os participantes de um trabalho cooperativo possam compreendê-las e utilizá-las para realizar suas tarefas, independente de sua localização geográfica.

4.2 Modelo DiSEN-CSE (DiSEN-Context Sensitive Environment)

De maneira geral, os trabalhos encontrados na literatura (Dey, 2000; Gómez, 2007; Vieira, 2006) destacam quatro elementos essenciais à *context-awareness*: captura, representação, armazenamento e disseminação das informações contextuais. Seguindo essa linha, o modelo definido para o ambiente DiSEN visa integrar esses elementos à característica de Percepção do modelo SPC (Sincronização, Percepção, Comunicação) (Pozza, 2005), para permitir o compartilhamento de informações contextuais entre os diversos *workpaces* envolvidos em um trabalho cooperativo. Para alcançar esse objetivo, cada vez que uma informação contextual é gerada por uma entidade do Gerenciador de Objetos (Local, Usuário ou Ferramentas) em um *workspace* do Ambiente de Desenvolvimento Distribuído de Software (ADDS), esta deverá ser capturada pelo Gerenciador de *Workspace*, como pode ser visto na Figura 4.2. Nesse gerenciador, o modelo DiSEN-CSE a transforma em uma informação que pode ser compreendida por todos os membros da equipe fisicamente dispersos e as compartilha com os demais *workspaces* existentes, incluindo o próprio *workspace* que gerou a informação.

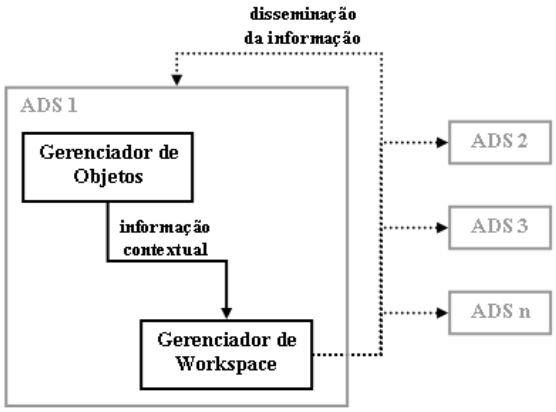

Figura 4.2: Compartilhando informações no DiSEN

A Figura 4.3 detalha o modelo DiSEN-CSE. O modelo possui quatro elementos essenciais: `Suporte à captura`, `Representação do contexto`, `Suporte ao processamento` e `Mecanismos de apresentação`, além de um repositório para armazenamento das informações contextuais, que estão descritos a seguir.

a. Suporte à captura: responsável por reconhecer as mudanças que ocorrem no contexto do trabalho. Essas mudanças poderão ser capturadas por agentes humanos ou de software. Um agente humano corresponde a qualquer usuário capaz de fornecer as informações referentes ao seu contexto atual. Um agente de software pode ser qualquer entidade autônoma, capaz de reconhecer informações de contexto sem a interferência de um sistema ou de outra entidade, podendo ser pró-ativos ou despertados por evento. Quanto à origem das informações, o ambiente DiSEN possui três tipos de fontes de informação: **manuais**, quando os agentes humanos registram seu próprio contexto a partir de uma determinada interface; **locais**, quando estão armazenadas em arquivos de configuração, interfaces gráficas ou derivados de outros sensores locais (podendo ser, por exemplo, um dispositivo físico); e **repositórios**, quando estão armazenadas em algum repositório de dados comum a todos os *workspaces*, como um servidor de banco de dados central. É importante notar que ambas as fontes manuais e locais podem ser recuperadas a partir de interfaces gráficas do *workspace*. A diferença está na voluntariedade. Quando as fontes de contexto são locais, a captura deve ser transparente, o que significa que o usuário não terá ciência de que suas ações estarão oferecendo informações contextuais. Nas fontes manuais, o contexto é solicitado ao usuário que o informa voluntariamente. As informações de fontes locais e repositórios devem ser capturadas por agentes de software, enquanto as fontes manuais sempre são alimentadas por agentes humanos.

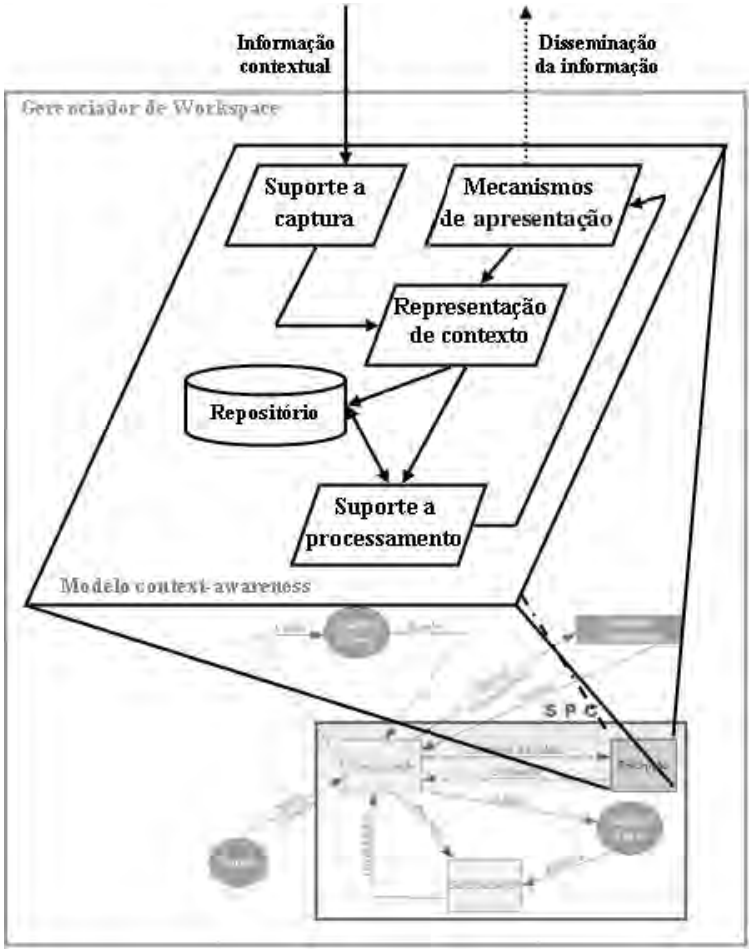

Figura 4.3: Modelo baseado em *context-awareness* para o DiSEN

b. Representação do contexto: responsável por receber as informações contextuais capturadas e representá-las formalmente. A importância da representação formal do contexto é destacada em diversos trabalhos encontrados na literatura (Baldauf et al., 2007; Nunes et al., 2007; Vieira, 2006), que apontam para a necessidade de que as diferentes entidades que participam das interações (pessoas, software, dispositivos) possuam a mesma compreensão semântica das informações de contexto. Dessa forma, o objetivo do elemento `Representação do contexto` é mapear as informações provenientes do `Suporte à captura` para o modelo de representação formal e relacioná-las com as demais informações contextuais já existentes no repositório, com base no modelo de representação definido. Existem na literatura diversas propostas de modelos de representação (Baldauf et al., 2007; Nunes et al., 2007; Vieira, 2006), entre eles, pares valor-chave, modelos baseados em marcação, baseados em lógica, baseados em representação gráfica, entre outros. Entretanto,

observa-se uma convergência para modelos baseados em ontologias, conforme discutido na Seção 2.5. Seguindo essa tendência, esse modelo foi adotado para o ambiente DiSEN devido ao seu alto nível de formalismo e possibilidade de inferência sobre as informações representadas. Além disso, o uso de ontologias favorece o reuso dos conceitos modelados, aumentando a interoperabilidade entre aplicações que poderão ser desenvolvidas no futuro como, por exemplo, ferramentas para o gerenciamento de memória organizacional e gestão de conhecimento.

c. Armazenamento: para o modelo proposto, as informações contextuais capturadas e representadas poderão ser persistentes ou transientes. As informações transientes são importantes para um determinado momento, mas não serão utilizadas no futuro, podendo ser disparadas para os mecanismos de apresentação, sem a necessidade de armazenamento. Ao contrário, as informações persistentes são aquelas que deverão permanecer em um repositório de dados para consultas futuras, formando um histórico de informações de contexto. Quando necessário, esse repositório, representado pelo Armazenamento, poderá ser acessado pelo `Suporte ao processamento` para a recuperação das informações nele contidos.

d. Suporte ao processamento: mecanismo de raciocínio capaz de deduzir contextos implícitos nas informações recebidas e corrigir possíveis inconsistências, com base nos relacionamentos existentes entre os conjuntos de informação, criados na `Representação do contexto`. As informações persistentes são adquiridas a partir do Armazenamento, enquanto as informações transientes são recebidas diretamente da `Representação do contexto`. Entretanto, todas as informações precisam ser processadas antes de serem enviadas para os `Mecanismos de apresentação`, a fim de possibilitar que as informações disseminadas sejam interpretadas e possuam, assim, maior relevância.

e. Mecanismos de apresentação: depois de serem representadas e processadas, as informações contextuais estão prontas para serem disponibilizadas para os `Mecanismos de apresentação`. Esses mecanismos serão responsáveis por identificar, automaticamente, quais *workspaces* estão interessados na informação, quais métodos serão utilizados para exibi-las (um *chat*, um *email*, uma ferramenta de geração de artefatos, um *pop-up* no *workspace* do usuário, entre outros métodos definidos em (Pozza, 2005)) e, por fim, se encarregarão de disseminá-las para as instâncias do ambiente, independente de sua localização geográfica, por meio da infraestrutura de comunicação disponível no DiSEN.

Para detalhar as principais funcionalidades do modelo, foram especificados os casos de uso que representam cada uma de suas quatro partes fundamentais. Para essa modelagem, foi utilizada a ferramenta MagicDraw 15.5 Enterprise[1]. A MagicDraw é uma ferramenta proprietária, entretanto, possui uma licensa *demo* que pode ser obtida no site em que está disponível. As seções 4.2.1 e 4.2.2 apresentam os modelos dos atores e dos casos de uso do DiSEN-CSE, respectivamente e suas especificações.

4.2.1 Especificação dos Atores

O modelo DiSEN-CSE possui cinco atores principais, como pode ser visto na Figura 4.4. O `agente` é uma generalização dos agentes de captura do ambiente e representam quaisquer entidades capazes de produzir informações contextuais ou capturar essas informações quando produzidas por outras entidades. Os agentes são especializados em `agente humano`, `agente de software pró-ativo` e `agente de software reativo`.

Figura 4.4: Representação dos atores do modelo e seus relacionamentos.

Um `agente de software pró-ativo` pode ser qualquer entidade autônoma, capaz de reconhecer informações de contexto sem a interferência de um sistema ou de outra entidade. Um `agente de software reativo` é qualquer entidade autônoma de software que, quando despertada por alguma outra entidade ou ação de um agente humano, é capaz de reconhecer informações de contexto.

Um `agente humano` é qualquer pessoa que utiliza o ambiente (usuário) capaz de oferecer, voluntária ou involuntariamente, informações sobre o contexto. Neste trabalho, os termos usuário e agente humano serão tratados como sinônimos.

E, por fim, o `gerenciador de notificações` é um componente de software responsável por receber as informações obtidas pelos agentes de captura e repassar para as

[1] www.magicdraw.com

entidades responsáveis pela representação, armazenamento e processamento. Depois do processamento, o gerenciador de notificações identifica quem são os usuários interessados e dissemina as informações. O gerenciador de notificações funciona, portanto, como um "delegador", que recebe a informação e reconhece quem deve tratá-la em cada estágio, até que esta chegue ao usuário final. Detalhes sobre o gerenciador de notificações serão apresentados no Capítulo 7.

4.2.2 Modelagem e Especificação dos Casos de Uso

O DiSEN-CSE possui quatro elementos principais, além do repositório para armazenamento das informações contextuais. Dessa forma, a visão de negócio dos casos de uso pode ser definido como apresentado na Figura 4.5

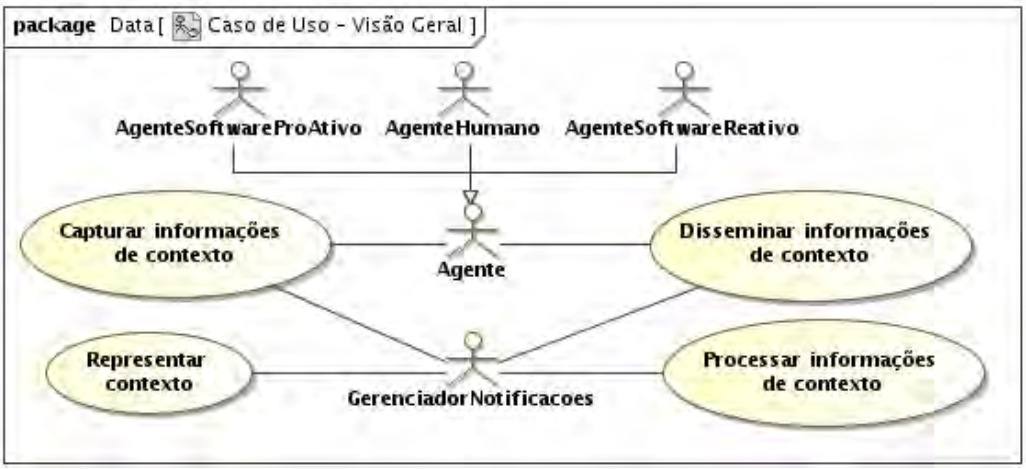

Figura 4.5: Diagrama de Casos de Uso – Visão de Negócio.

Neste diagrama, é possível observar que os agentes (humanos ou de software) atuam sobre a captura e disseminação, enquanto o gerenciador de notificações atua sobre os quatro elementos do modelo. Isso porque os agentes humanos podem oferecer informações sobre o contexto de suas ações e estão interessados nas informações contextuais de outros usuários para a realização do seu trabalho. Os agentes de software, por sua vez, possuem, explicitamente, a responsabilidade de captura automática das informações contextuais. Além disso, podem ser responsáveis por receber as informações prontas para disseminação e entregá-las aos usuários interessados, de acordo com o método de apresentação mais adequado (os métodos de apresentação propostos para o ambiente DiSEN podem ser encontrados em (Pozza, 2005)).

Por outro lado, o gerenciador de notificações não se preocupa com a forma com que as informações são capturadas ou disseminadas. Ele recebe o resultado da captura, delega à representação e ao processamento e entrega o resultado dessas operações à disseminação. Dessa forma, quer seja recebendo ou enviando informações, o gerenciador de notificações age sobre todos os elementos do modelo.

O diagrama de visão de negócio possui quatro casos de uso:

Capturar informações de contexto: corresponde ao elemento `Suporte à captura`. Seu objetivo é definir como ocorre a captura das informações de contexto, quer sejam informações percebidas por agentes de software, quer sejam providas por agentes humanos, voluntária ou involuntariamente;

Representar contexto: corresponde ao elemento `Representação de contexto` e é responsável por definir a representação formal das informações contextuais, respeitando as depedências e relacionamentos com informações já existentes. Além disso, as informações persistentes precisam ser armazenadas, enquanto as transientes precisam ser enviadas para processamento e disseminação;

Processar informações de contexto: permite a extração de novas informações a partir daquelas recebidas da representação, correspondendo ao elemento `Suporte ao processamento`. O raciocínio sobre essas informações é possível devido à utilização do modelo de representação baseado em ontologias, que oferece um modelo semântico que possibilita a realização de inferência sobre as informações representadas;

Disseminar informações de contexto: define a escolha do melhor método de apresentação para cada indivíduo interessado naquele contexto e disseminar as informações, atendendo a relevância da informação, o perfil e as preferências dos usuários (elemento `Mecanismos de apresentação`).

Cada um desses quatro casos foram detalhados em casos de uso mais específicos, que serão descritos a seguir. A documentação completa dos casos de uso pode ser encontrada no Apêndice B.

Capturar informações de contexto

O caso de uso `Capturar informações de contexto` pode ser detalhado como na Figura 4.6.

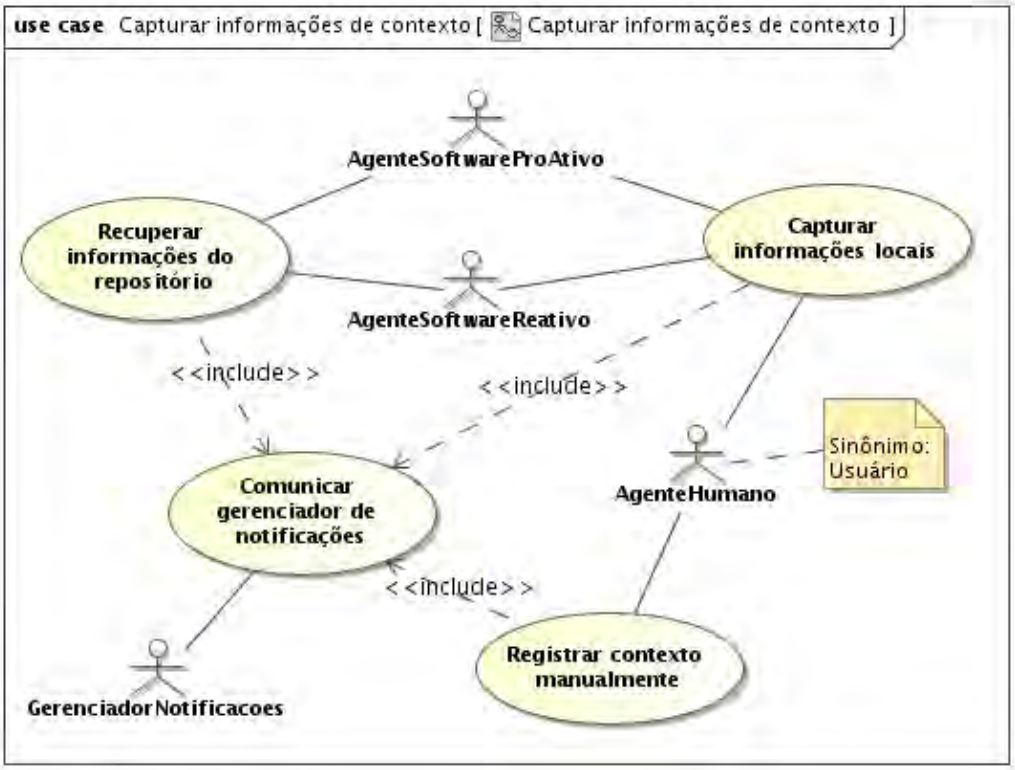

Figura 4.6: Diagrama de Casos de Uso – Capturar informações de contexto.

No caso de uso `Capturar informações locais`, as informações contextuais podem ser obtidas de arquivos de configuração, interfaces gráficas ou por interpretação feita a partir de outros sensores (por exemplo, um sensor físico). Nesse caso, a captura das informações é transparente ao usuário, mesmo que ele as forneça por uma interface gráfica (ação involuntária).

As informações de contexto podem ser persistentes ou transientes. Quando são persistentes, são armazenadas em um repositório e podem ser recuperadas quando necessário. Para isso, os agentes de software pró-ativos (quando necessita de informações históricas) ou reativos (quando o usuário solicita uma consulta a um determinado mecanismo de recuperação) podem realizar o caso de uso `Recuperar informações do repositório`, que obtém uma solicitação de consulta e busca a informação no repositório de dados.

O caso de uso `Registrar contexto manualmente` consiste em um agente humano informar seu contexto a partir de uma interface gráfica, por exemplo, preenchendo um formulário. Diferentemente do caso de uso `Capturar informações locais`, o usuário está ciente de que sua ação fornecerá informações contextuais (ação voluntária).

O caso de uso `Comunicar o gerenciador de notificações` é incluído por todos os casos de uso de captura, para comunicar que uma informação foi capturada. Nesse

caso de uso, o gerenciador de notificações recebe as informações de contexto capturadas pelos agentes e decide o tratamento que será dado à informação, de acordo com suas características (persistente, transiente ou resultado de consulta).

Representar contexto

O caso de uso `Representar contexto` pode ser especificado em três casos de uso correspondentes, apresentados na Figura 4.7.

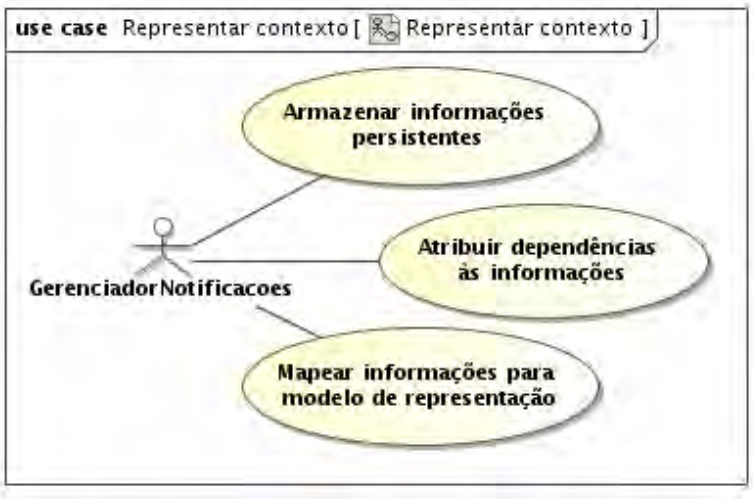

Figura 4.7: Diagrama de Casos de Uso – Representar contexto.

No caso de uso `Mapear informações para modelo de representação`, o gerenciador de notificações, ao receber informações dos agentes de captura, deverá representá-las formalmente, com base na ontologia predefinida. Assim, esse caso de uso é responsável por mapear essas informações para as classes ontológicas, respeitando as relações preestabelecidas.

Algumas classes de informações podem depender de outras informações relacionadas. Por essa razão, o caso de uso `Atribuir dependências às informações` é responsável por mapear as dependências da informação atual, relacionando-as com as informações já existentes no repositório de informações contextuais, para facilitar o processamento, posteriormente.

E, por fim, no caso de uso `Armazenar informações persistentes`, o gerenciador de notificações fica responsável por, após a representação, enviá-las para o armazenamento, a fim de garantir que esta informação possa ser recuperada em um momento futuro.

Processar informações de contexto

O gerenciador de notificações possui acesso ao elemento de `Suporte ao processamento`, que consiste em um motor de inferência capaz de deduzir informações a partir daquelas já existentes. O caso de uso `Deduzir novas informações a partir das existentes` (Figura 4.8) consiste em submeter as informações que serão disseminadas a esse motor de inferência, obtendo novas informações por raciocínio ou dedução.

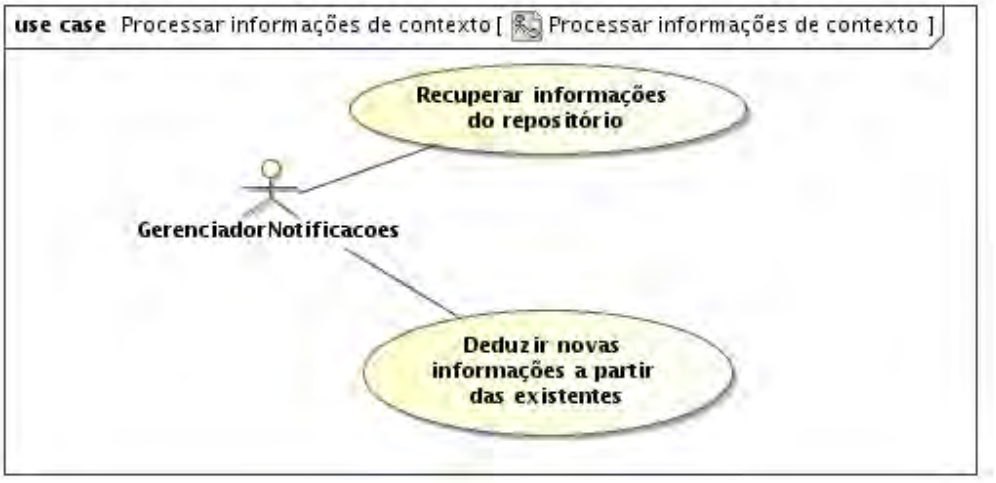

Figura 4.8: Diagrama de Casos de Uso – Processar informações de contexto.

Disseminar informações de contexto

O caso de uso `Disseminar informações de contexto` pode ser descrito nos casos de uso apresentados na Figura 4.9.

O gerenciador de notificações mantém uma lista de receptores da informação contextual. Um receptor pode ser um local (todos os *workspaces* conectados a um determinado servidor), uma ferramenta (todos os usuários que utilizam determinada ferramenta) ou um usuário (um indivíduo ou um grupo). No caso de uso `Identificar receptores da informação`, quando o gerenciador recebe uma informação representada e processada, ele deve se encarregar de verificar quem está interessado naquele evento e formar uma lista de receptores para uma informação determinada, para então enviar essa informação às entidades que a utilizam.

As informações, quando são disseminadas, são exibidas por mecanismos de apresentação ou ferramentas de comunicação que sejam capazes de oferecer a informação ao usuário. O caso de uso `Identificar mecanismos de apresentação` seleciona os mecanismos que serão utilizados para cada entidade da lista de receptores, de acordo

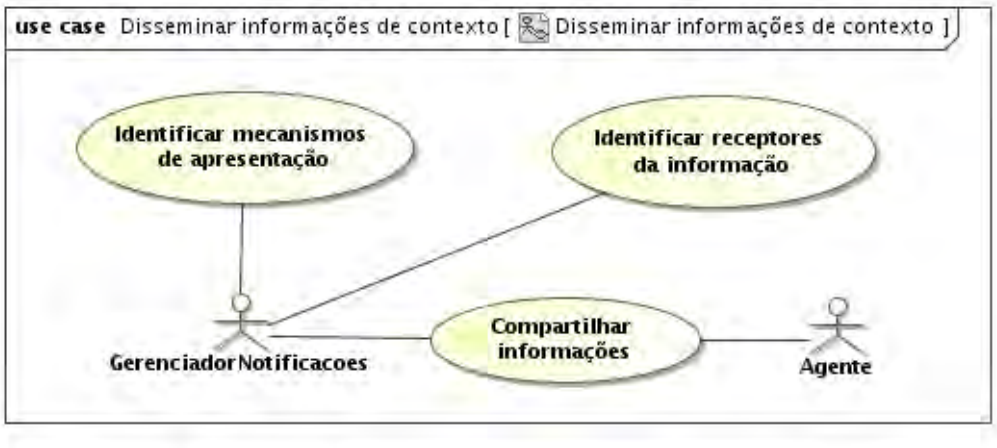

Figura 4.9: Diagrama de Casos de Uso – Disseminar informações de contexto.

com a relevância da informação para a entidade receptora. Para isso, o gerenciador de notificações mantém um resgistro de como as informações precisam ser exibidas (qual método utilizar para a disseminação).

De posse da informação contextual, da lista dos receptores e da lista dos mecanismos de apresentação, o gerenciador de notificações realiza o caso de uso `Compartilhar informações`, que se encarrega de enviar as informações para os respectivos mecanismos de apresentação, para serem exibidas na forma de mensagem para as entidades receptoras.

4.3 Considerações Finais do Capítulo

O desenvolvimento distribuído de software é uma abordagem caracterizada pelo desenvolvimento em locais remotos e pela possibilidade de dispor os recursos em âmbito global. Mas, aliado às vantagens desse tipo de desenvolvimento, surgem diversas dificuldades, entre elas, as barreiras impostas pela comunicação.

Analisando mais especificamente os problemas de comunicação, identificou-se que para obter uma compreensão clara sobre os objetos de cooperação, é necessário que se conheça também a forma como este objeto foi produzido. Assim, observou-se a necessidade de fornecer informações sobre o contexto das entidades do ambiente no momento em que ocorre a interação. Devido às diferenças relacionadas à distância, a representação das informações torna-se uma característica essencial para diminuir as ambiguidades. A representação formal aumenta a capacidade de processamento automatizado, permitindo a dedução de novas informações a partir daquelas já existentes. Além disso, ponderou-se que, com base em um gerenciador de notificações de eventos, a disseminação de informações

relevantes a cada usuário poderia ser feita automaticamente, sem que estes precisassem solicitá-las a um repositório de informações contextuais. Atentou-se ainda que, separando os mecanismos de apresentação do gerenciador de notificações, que produz as mensagens que serão enviadas para os usuários, seria possível gerar mensagens que pudessem ser destinadas a diversos métodos de apresentação diferentes. Essa característica favorece a disseminação de acordo com a relevância das informações, o perfil dos usuários que as recebem e suas preferências, facilitando a percepção dos indivíduos sobre as informações produzidas no ambiente.

Diante dos pontos apresentados, as principais contribuições do modelo DiSEN-CSE são:

- prover mecanismos capazes de capturar as informações contextuais, incluindo circunstâncias físicas, sociais ou históricas;
- representar adequadamente essas informações contextuais por meio de uma ontologia, definida sobre os conceitos existentes no ambiente DiSEN, oferecendo semântica às informações existentes no ambiente, diminuindo as ambiguidades na comunicação;
- utilizar mecanismos de apresentação para disseminar automaticamente informações contextuais a todos os participantes dos diferentes projetos, independente de sua localização física. Com isso, as pessoas não precisam, necessariamente, consultar o repositório para estar ciente das modificações, sendo automaticamente notificadas, permitindo a disseminação rápida e eficiente das informações e facilitando a tomada de decisões, de acordo com o referido contexto;
- separar o notificador dos mecanismos de apresentação, permitindo que diferentes indivíduos recebam as informações em uma interface adequada ao seu perfil, preferências e de acordo com a relevância da informação para o seu contexto.

Em virtude da amplitude do modelo DiSEN-CSE, este trabalho não contempla a implementação do modelo completo. O escopo está em definir as estruturas básicas para que cada elemento do modelo trabalhe corretamente. Trabalhos do grupo de pesquisa GEESD (Grupo de Estudos em Engenharia de Software Distribuído), responsável pelo desenvolvimento do ambiente DiSEN, já foram iniciados com o objetivo de desenvolver outras partes do modelo. Um destes trabalhos tem seu foco na construção de agentes de software para atuar como sensores de mudanças, capturando as informações contextuais que serão manipuladas pelo modelo. Outro trabalho em andamento concentra-se em

extrair conhecimento das informações representadas na ontologia, realizando o processamento dessas informações e deduzindo novas, com base em motores de inferência. Outras partes do modelo serão construídos em trabalhos futuros, discutidos no Capítulo 8.

Para que todos os elementos do modelo se integrem de maneira eficaz, é necessário desenvolver uma estrutura capaz de garantir que as informações contextuais obtidas pelo `Suporte à captura` poderão ser representadas na ontologia e que haverá um gerenciador capaz de compartilhar essas informações com as entidades interessadas. Os Capítulos 5, 6 e 7 descrevem, portanto, as partes necessárias a essa estrutura fundamental do modelo, que são o foco deste trabalho.

Capítulo

5

Modelagem Contextual

5.1 Considerações Iniciais

O modelo DiSEN-CSE, apresentado no Capítulo 4 tem por principais objetivos capturar, manipular e compartilhar as informações contextuais presentes no ambiente DiSEN. Entretanto, para que esse objetivo seja atingido, é necessário saber quais são as informações contextuais existentes no ambiente e as relações existentes entre elas.

Conforme visto no Capítulo 2, Vieira (2008) afirma que o contexto consiste em um conjunto de elementos sobre uma determinada entidade que, para ser útil, precisa apoiar a resolução de um problema. Para identificar quais elementos possuem essas características no domínio do ambiente DiSEN, foi utilizado o metamodelo proposto em (Vieira, 2008) (*Context Metamodel* – apresentado na Seção 2.6), para especificar as principais ações realizadas no ambiente DiSEN, definindo os elementos contextuais e a relações existentes entre esses elementos, bem como a forma como esses elementos influenciam o comportamento do ambiente.

Inicialmente, com base no conhecimento do domínio oferecido pelos *stakeholders* não usuários (Apêndice A), foram definidas as principais funcionalidades do ambiente DiSEN, sendo representadas no diagrama de casos de uso da Figura 5.1.

O DiSEN possui basicamente três atores principais, que podem realizar ações sobre o ambiente:

Usuário: qualquer indivíduo que possui um *login* e senha, podendo se conectar ao ambiente a acessar suas funcionalidades. O usuário é o ator com menor nível

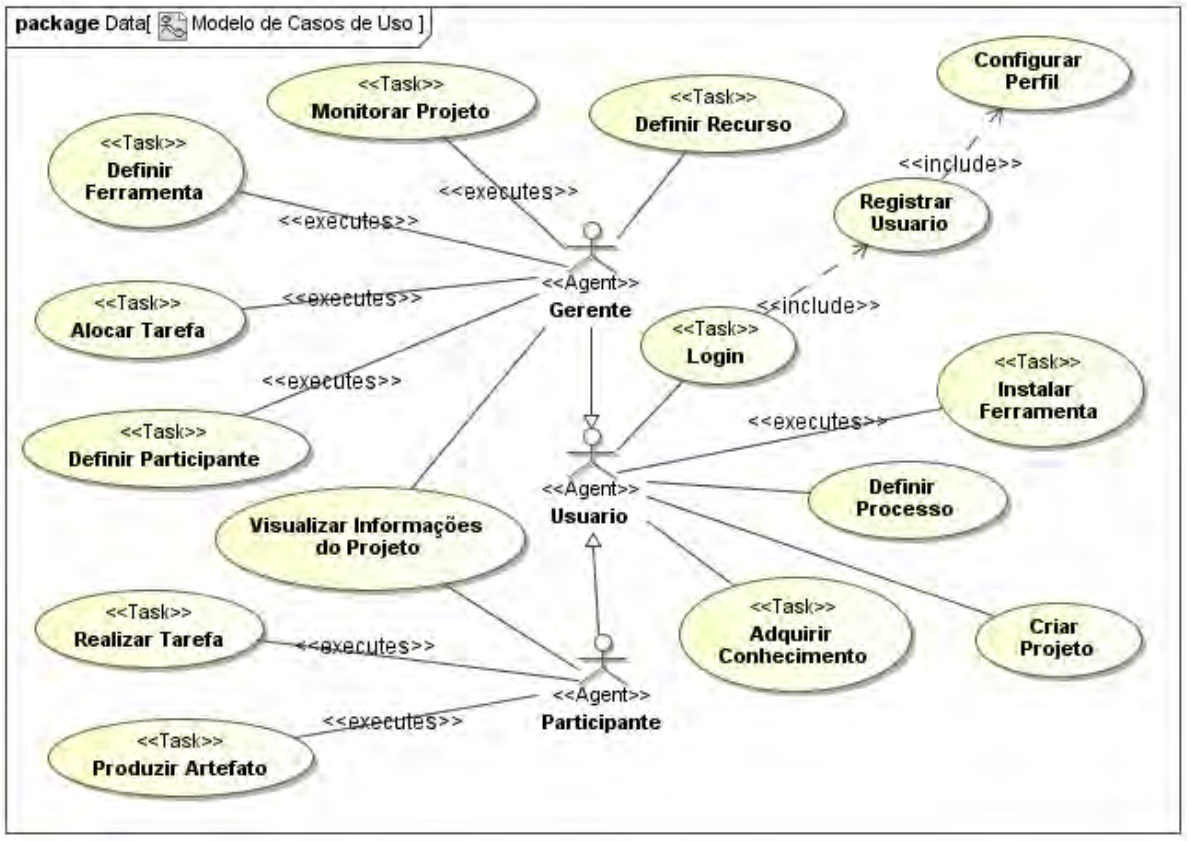

Figura 5.1: Diagrama de Casos de Uso do ambiente DiSEN.

de permissão e todos os demais atores são usuários, herdando, portanto, suas características.

Participante: usuário que participa de um projeto. Um usuário pode participar de mais de um projeto ao mesmo tempo. Os usuários são participantes apenas no contexto do projeto em que possui um papel definido. Por exemplo, um usuário pode ser participante de um projeto X e não ser participante de um projeto Y. Assim, no contexto do projeto X, o usuário possui as permissões de um participante. Já no projeto Y, possui apenas as permissões de um usuário.

Gerente: usuário responsável por monitorar e controlar a execução de um projeto. Assim como os participantes, o gerente pode possuir mais de um projeto sob sua responsabilidade. Da mesma forma, um usuário só possui as permissões de gerente no contexto dos projetos em que possui esse papel. Por exemplo, um usuário pode ser gerente do projeto X e não ser gerente do projeto Y. Assim, no contexto do projeto

X, o usuário possui as permissões de um gerente, entretanto, no projeto Y, possui as permissões de um usuário, apenas, a menos que seja um participante.

Um usuário precisa estar registrado para possuir um *login* e uma senha (`Registrar usuário`). De posse desses dados, pode se conectar em um *workspace* (`Login`) e, a partir de então, realizar ações no ambiente. Um usuário pode alterar suas configurações de perfil (`Configurar Perfil`), definir um processo, indicando quais fases e atividades deverão conter (`Definir Processo`), criar um projeto, informando os objetivos e as datas previstas para cada atividade/tarefa. Além disso, um usuário pode instalar ferramentas em seu *workspace* (`Instalar Ferramenta`) e, quando participa de projetos ou realiza treinamentos, pode adquirir novos conhecimentos (`Adquirir Conhecimento`)

Um participante, além das funcionalidades que um usuário realiza, pode visualizar as informações de projetos do qual participa (`Visualizar informações do Projeto`), realizar as tarefas para as quais foi alocado (`Realizar Tarefa`) e criar ou modificar os artefatos de sua responsabilidade (`Produzir Artefatos`).

O gerente pode visualizar as informações sobre os projetos que gerencia (`Visualizar informações do Projeto`), além de receber informações sobre o andamento destes projetos (`Monitorar Projeto`), para que possa tomar decisões sobre as ações realizadas. O gerente é também responsável por definir quais ferramentas devem ser usadas para gerar os artefatos do projeto (`Definir Ferramentas`), quem serão os participantes do projeto (`Definir Participantes`) e quais participantes serão responsáveis por cada tarefa (`Alocar Participantes`). Além disso, pode definir quais recursos serão alocados para cada projeto (`Definir Recurso`).

Analisando as características gerais do desenvolvimento de software, destacam-se, facilmente, ações que não foram consideradas nesse diagrama de casos de uso, como planejamento dos projetos, gerenciamento de riscos, gerenciamento de qualidade, entre outros. Entretanto, para este trabalho, o escopo foi definido para envolver as informações contextuais produzidas pelas ações que os indivíduos realizam pelo ambiente (software), não envolvendo as ações de negócio. A contextualização das informações de nível organizacional serão tratadas em trabalhos futuros.

A Seção 5.2 apresenta as principais atividades do processo utilizado para a especificação do modelo contextual do ambiente DiSEN. O caso de uso `Realizar Tarefa` será utilizado para demonstrar a modelagem de contexto realizada. A documentação completa dos demais casos de uso pode ser encontrada em (Chaves, 2009). Os modelos foram gerados na ferramenta MagicDraw 15.5 Enterprise. A escolha de uma ferramenta

proprietária ocorreu porque conjunto de *UML profile*[1], criado por Vieira (2008), que possui os estereótipos para aplicação do metamodelo *Context Metamodel*, foi desenvolvido para ser utilizado nesta ferramenta.

5.2 Modelagem Contextual do Ambiente DiSEN

O processo *Context Process*, utilizado para modelar as informações contextuais, foi proposto por Vieira (2008) e seus artefatos são produzidos utilizando o metamodelo *Context Metamodel*. As principais fases estão descritas na Figura 5.2.

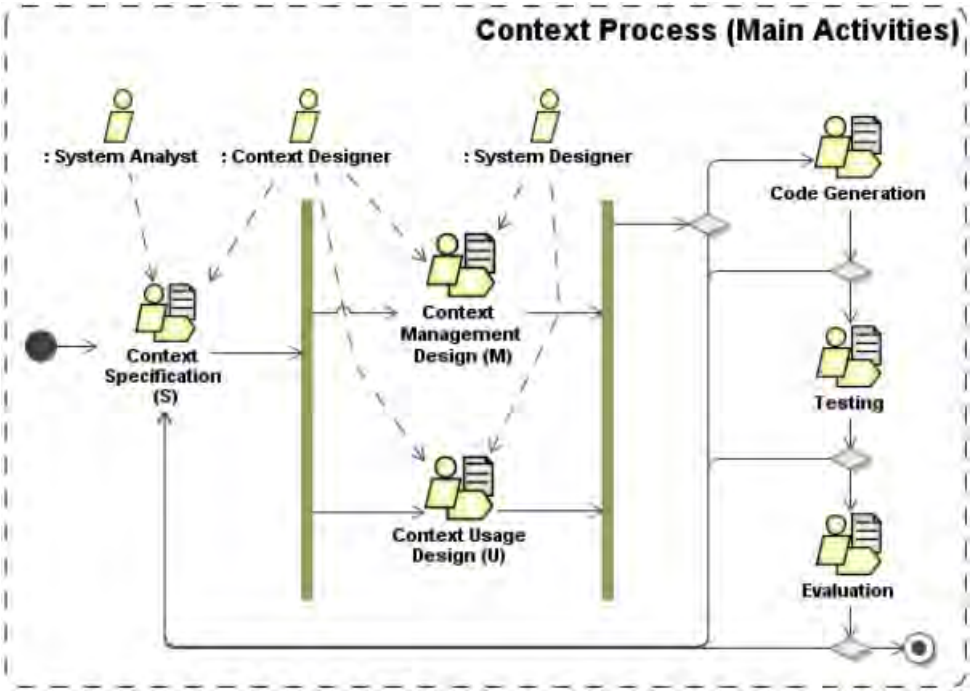

Figura 5.2: Processo para especificação de contexto e projeto de sistemas sensíveis ao contexto (Vieira, 2008).

O processo *Context Processs* está dividido em quatro fases principais, que correspondem às fases do desenvolvimento de software: *Context Specification* (análise), *Context Management Design* e *Context Usage Design* (projeto), *Code Generation*, *Testing* e *Evaluation* (manutenção) (Vieira, 2008).

A fase de análise (*Context Specification*) concentra-se na criação do modelo conceitual do contexto, baseado nos requisitos do negócio. Possui quatro atividades: `Identificar`

[1] *UML profile* são extensões customizadas do Metamodelo UML (UML *Metamodel*) para domínios de aplicação particulares, sendo compostos de estereótipos (*stereotypes*), definições de *tags* (*tag definitions*) e restrições (*constraints*) (Vieira, 2008)

`o Foco`, `Identificar as Variações de Comportamento`, `Identificar as Entidades e os Elementos Contextuais` e `Verificar a Relevância do Elemento Contextual`.

Considerando o diagrama de casos de uso do ambiente DiSEN (Figura 5.1), observa-se que cada caso de uso corresponde a uma associação (*executes*) entre um agente (*Agent*) e uma tarefa que deve ser realizada (*Task*). Dessa forma, cada caso de uso corresponde a um foco, cujos elementos contextuais precisam ser identificados. É importante salientar que apenas os casos de uso mais relevantes foram especificados. O critério para estabelecer a relevância é o quão variável é o contexto daquele caso de uso. Aqueles casos que podem ser executados sem variações de comportamento não foram identificados como focos e, portanto, não foram modelados.

Para cada foco, é necessário identificar que variações são esperadas no comportamento do ambiente e quais fatores influenciam para a ocorrência dessas variações. Assim, para o foco

$$\langle\langle Agent\rangle\rangle \text{ Participante } \langle\langle executes\rangle\rangle \ \langle\langle Task\rangle\rangle \text{ Realizar Tarefa,}$$

as seguintes variações de comportamento foram identificadas:

- Uma tarefa T1 deve produzir um artefato A1.
- O artefato deve ser gerado usando uma ferramenta F1.
- A ferramenta a ser utilizada para produzir o artefato é definida pelo gerente do projeto.
- Caso a ferramenta não exista no *workspace* do participante, esta deve ser instalada. Se a ferramenta estiver instalada, mas não for a última versão, ela deve ser atualizada.
- Uma tarefa T1 pode ser dependente de outra tarefa T2. Nesse caso, T1 só pode ser executada após o término de T2.
- O participante responsável por T1 deve ser constantemente notificado sobre o estado do artefato associado a T2.
- Artefatos têm arquivos.
- Arquivos possuem versões.
- Versões de arquivos têm estados:

1. Conflito: o participante detectou algum conflito nas versões.
2. Bloqueada: versão que não pode ser utilizada para modificações até que um determinado problema seja resolvido.
3. Obsoleta: versões que anteriormente foram versões finais, mas foram substituídas por novas versões.
4. Inutilizada: versão com erros que não deve ser utilizada.
5. Final: versão válida do artefato (só existe uma única versão válida).

- Arquivos possuem estados:

1. Indisponível: não existe versão alguma do arquivo.
2. Intermediário: existe pelo menos uma versão do arquivo, mas nenhuma delas em estado final.
3. Final: tem uma versão pronta (em estado final).

- Artefatos possuem estados:

1. Indisponível: nenhum dos seus arquivos está na versão final.
2. Intermediário: pelo menos um dos seus arquivos está na versão final, mas não todos (ainda não está pronto).
3. Final: todos os arquivos daquele artefato estão na versão final.

- Tarefa possui estados:

1. Aguardando Recursos: a tarefa ainda não iniciou, portanto não existem versões de arquivos.
2. Em Andamento: a tarefa iniciou, mas o artefato que esta deve gerar ainda é indisponível ou intermediário.
3. Cancelado: a tarefa foi cancelada e não será mais realizada.
4. Suspenso: a tarefa foi suspensa por algum motivo e poderá ser posteriormente retomada.
5. Concluído: o artefato que a tarefa deve gerar foi concluído (estado Final).

- A atividade à qual a tarefa faz parte possui estados:

1. Planejamento: nenhuma tarefa daquela atividade foi iniciada.

2. Em andamento: pelo menos uma tarefa daquela atividade foi iniciada.
3. Cancelado: a atividade foi cancelada e nenhuma de suas tarefas serão realizadas.
4. Suspenso: a atividade foi suspensa por algum motivo e poderá ser posteriormente retomada. Enquanto a atividade está suspensa, as tarefas não são realizadas, até a regularização da situação.
5. Concluído: todas as tarefas da atividade foram concluídas.

- A fase ao qual a atividade pertence possui estados:

1. Planejamento: nenhuma atividade daquela fase foi iniciada.
2. Em andamento: pelo menos uma atividade daquela fase foi iniciada.
3. Cancelado: a fase foi cancelada e nenhuma de suas atividades serão realizadas.
4. Suspenso: a fase foi suspensa por algum motivo e poderá ser posteriormente retomada. Enquanto a fase está suspensa, as atividades não são realizadas, até a regularização da situação.
5. Concluído: todas as atividades da fase foram concluídas.

- O projeto ao qual a fase pertence possui estados:

1. Planejamento: nenhuma fase daquele projeto foi iniciada.
2. Em andamento: pelo menos uma fase daquele projeto foi iniciada.
3. Cancelado: o projeto foi cancelado e nenhuma de suas fases serão realizadas.
4. Suspenso: o projeto foi suspenso por algum motivo e poderá ser posteriormente retomado. Enquanto o projeto está suspenso, as fases não são realizadas, até a regularização da situação.
5. Concluído: todas as fases do projeto foram concluídas.

- Se o artefato que o participante está gerando na tarefa T1 tiver alguma versão marcada como em conflito a tarefa deve ficar suspensa, até que o gerente resolva o estado do artefato.

- O sistema pode notificar o gerente que um participante está aguardando a resolução de conflito de um artefato.

- Quando o participante informar uma data de fim para sua tarefa, avisar aos participantes das tarefas dependentes que a tarefa está pronta.

- Se o gerente alterar o estado da versão final para bloqueada ou inutilizada, o participante que gerou a versão deve ser avisado que esta foi alterada pelo gerente (ele provavelmente precisará refazer). Os participantes das tarefas dependentes também precisam ser avisados.

Depois de identificar o foco e as variações de comportamento, o próximo passo definido pelo processo *Context Process* é identificar as entidades e os elementos contextuais que influenciam em cada variação.

Na Figura 5.3, as classes marcadas com o estereótipo $\langle\langle ContextualEntity \rangle\rangle$ representam as entidades contextuais, conforme definido no metamodelo *Context Metamodel* (Seção 2.6). Os elementos contextuais (CE) possuem o estereótipo $\langle\langle ContextualElement \rangle\rangle$.

Como exemplo, `Tarefa` é uma entidade contextual do foco `Realizar Tarefa`. Como elementos contextuais, destaca-se, entre outros:

- `Tarefa.dataInicio`
- `Tarefa.estado`
- `Tarefa.dependeDeTarefa`
- `Tarefa.temTarefaDependente`
- `Tarefa.temArtefato`
- `Tarefa.realizadaPorParticipante`

Esses elementos são aqueles que influenciam o foco em questão. Outros atributos da classe foram marcados como $\langle\langle ContextualElement \rangle\rangle$, como `Tarefa.dataInicioPrevista` e `Tarefa.dataFimPrevista`. A razão de não serem listados é que esses CEs têm influência sobre outros focos. A ferramenta de modelagem utilizada não permite que um elemento seja marcado com o estereótipo em apenas um diagrama. A partir do momento que um elemento é definido como CE, em todos os lugares onde aparecer, possuirá o estereótipo definido.

Depois da definição das entidades e elementos contextuais, a próxima atividade do processo é a definição da relevância de cada elemento contextual para o foco. Essa atividade não foi realizada por dois motivos. Primeiro, o processo define que, para cada elemento contextual exista um relacionamento com uma classe `Realizar Tarefa`, que representa, conceitualmente, o foco (marcada com o estereótipo $\langle\langle Focus \rangle\rangle$. O foco `Realizar Tarefa` possui 28 CEs definidos. Isso significa que haveriam até 28 relacionamentos entre entidades contextuais e essa classe, poluindo a visibilidade do diagrama e tornando inviável a

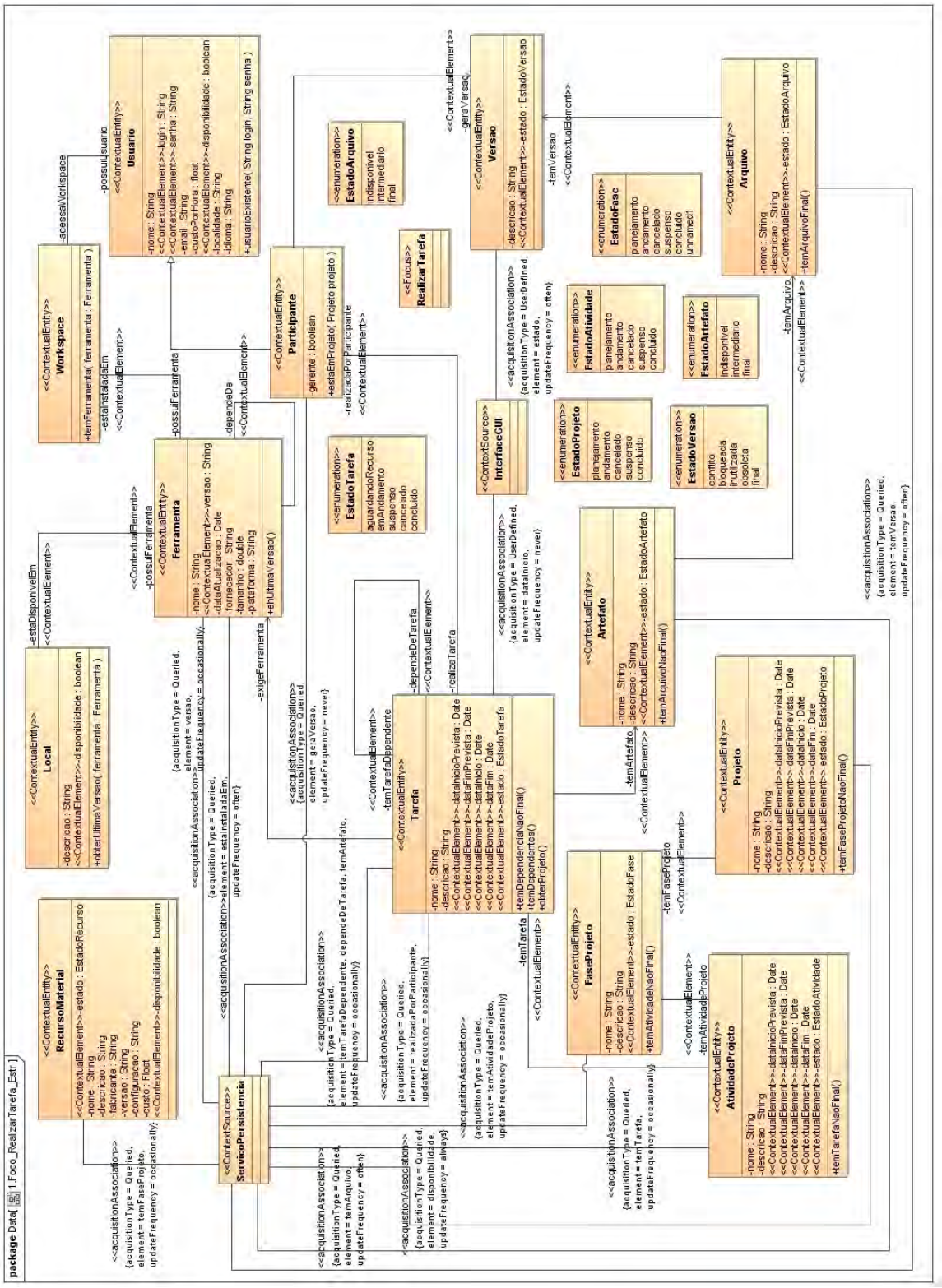

Figura 5.3: Modelo Conceitual – Foco: Realizar Tarefa

representação. Assim, novos estudos deverão ser realizados e alternativas serão discutidas para uma melhor representação.

Segundo, definir a relevância de uma informação é uma ação complexa e depende de diversos fatores, além de saber qual é o foco. Por exemplo, para o foco `Realizar Tarefa`, se a data de início da tarefa está atrasada, essa informação pode ser analisada de maneiras diferentes para o participante que realiza a tarefa, para o participante que realiza uma tarefa dependente e para o gerente do projeto. Para cada perfil, portanto, a informação pode ter uma relevância diferente. Por essas razões, a atividade de verificar a relevância dos elementos contextuais será tratada, mais cuidadosamente, em trabalhos futuros. Para este trabalho, assume-se que todas as informações tenham a mesma relevância e devem ser tratadas com a mesma prioridade.

Concluída a fase de análise do *Context Process*, iniciou-se as fases de projeto, que correspondem às fases *Context Management Design* (projeto de gerenciamento do contexto) e *Context Usage Design* (projeto de utilização do contexto).

A fase de gerenciamento do contexto concentra-se na forma como as informações são adquiridas e processadas. O primeiro passo consiste em definir as propriedades de aquisição de todos os elementos contextuais, o que significa dizer, para cada CE, de que maneira devem ser capturados.

No metamodelo *Context Metamodel*, as informações necessárias sobre a aquisição de cada CE são: a fonte de contexto, o tipo de aquisição, a frequência de atualização e, opcionalmente, uma expressão de formação, indicando uma regra para transformar uma informação obtida para o formato esperado.

De acordo com Vieira (2008), o tipo de aquisição de um CE pode ser:

***Sensed*:** (informação sentida) obtida por um sensor físico ou virtual. Em geral, essas informações mudam frequentemente;

***Profiled*:** (informações extraídas de perfis) extraídas de perfis de pessoas ou dispositivos. Para mudar, depende de alguém manter seu valor atualizado;

***UserDefined*:** (informações definidas pelo usuário) informada por um agente em tempo de execução ou sob demanda,por uma interface gráfica, por exemplo. Como não pode ser inferida automaticamente, deve ser verificado se existem alterações todas as vezes em que a informação for necessária;

***Queried*:** (informações consultadas) extraídas de repositórios externos ao sistema. Geralmente relacionadas a dados transacionais, essas informações mudam com pouca frequência;

***Derived*:** (derivadas) inferidas por funções de transformações ou regras de inferência. Assim como as informações definidas pelos usuários, mudanças devem ser verificadas sempre que forem necessárias.

Quanto à frequência de atualização de um CE, o metamodelo *Context Metamodel* possui a seguinte classificação:

***Never*:** (nunca) o valor do CE é estável e nunca muda;

***Occasionally*:** (ocasionalmente) o valor do CE pode mudar, mas não regularmente;

***Often*:** (frequentemente) o valor do CE é dinâmico e muda regularmente;

***Always*:** (sempre) o valor do CE é volátil, muda constantemente e pode estar desatualizado logo após a sua aquisição;

Dessa forma, para o foco `Realizar Tarefa`, gerou-se o seguinte documento de configuração de aquisição:

1. `Ferramenta.estaInstaladaEm`
    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Often*
2. `Ferramenta.versao`
    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*
3. `Tarefa.dependeDeTarefa`
    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*
4. `Tarefa.temTarefaDependente`
    - Fonte de contexto: Repositório de Dados

- Modo de aquisicao: *Queried*
- Frequencia de atualizacao: *Occasionally*

5. `RecursoMaterial.disponibilidade`
   - Fonte de contexto: Repositório de Dados
   - Modo de aquisicao: *Queried*
   - Frequencia de atualizacao: *Always*
6. `Tarefa.dataInicio`
   - Fonte de contexto: Formulário de Entrada de Dados
   - Modo de aquisicao: *UserDefined*
   - Frequencia de atualizacao: *Never*
7. `Participante.geraVersao`
   - Fonte de contexto: Repositório de Dados
   - Modo de aquisicao: *Queried*
   - Frequencia de atualizacao: *Never*
8. `Versao.estado`
   - Fonte de contexto: Formulário de Entrada de Dados
   - Modo de aquisicao: *UserDefined*
   - Frequencia de atualizacao: *Often*
9. `Arquivo.temVersao`
   - Fonte de contexto: Repositório de Dados
   - Modo de aquisicao: *Queried*
   - Frequencia de atualizacao: *Often*
10. `Arquivo.estado`
    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*
11. `Artefato.temArquivo`

- Fonte de contexto: Repositório de Dados
- Modo de aquisicao: *Queried*
- Frequencia de atualizacao: *Often*

12. `Artefato.estado`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*

13. `Tarefa.temArtefato`

    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*

14. `Tarefa.estado`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*

15. `Tarefa.dataFinal`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Never*

16. `AtividadeProjeto.temTarefa`

    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*

17. `AtividadeProjeto.estado`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*

18. `AtividadeProjeto.dataInicio`

    - Modo de aquisicao: *Derived*

- Frequencia de atualizacao: *Never*

19. `AtividadeProjeto.dataFinal`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Never*

20. `FaseProjeto.temAtividadeProjeto`

    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*

21. `FaseProjeto.estado`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*

22. `FaseProjeto.dataInicio`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Never*

23. `FaseProjeto.dataFinal`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Never*

24. `Projeto.temFaseProjeto`

    - Fonte de contexto: Repositório de Dados
    - Modo de aquisicao: *Queried*
    - Frequencia de atualizacao: *Occasionally*

25. `Projeto.estado`

    - Modo de aquisicao: *Derived*
    - Frequencia de atualizacao: *Often*

26. `Projeto.dataInicio`

- Modo de aquisicao: *Derived*
- Frequencia de atualizacao: *Never*

27. `Projeto.dataFinal`

- Modo de aquisicao: *Derived*
- Frequencia de atualizacao: *Never*

28. `Tarefa.realizadaPorParticipante`

- Fonte de contexto: Repositório de Dados
- Modo de aquisicao: *Queried*
- Frequencia de atualizacao: *Occasionally*

Analisando as configurações de aquisição dos CEs do foco `Realizar Tarefa` é possível observar que os elementos cujo modo de aquisição é *Derived* não possuem fonte de dados. Nesses casos, os dados são fornecidos a partir da ocorrência de algum evento sobre outro elemento contextual. Por exemplo, se todas as fases de um projeto foram concluídas no dia 19 de março de 2009, então, é possível inferir que a data final do projeto é 19 de março de 2009. Essa informação não possui uma fonte específica, mas é derivada de uma regra existente na organização que diz que um projeto concluído é aquele cujas fases foram concluídas.

O Repositório de Dados faz referência ao Serviço de Persistência, definido no *framework* FRADE. O Serviço de Persistência é responsável por persistir as informações no repositório central de dados do ambiente DiSEN, realizando as consultas, inserções, alterações e remoções dos objetos de negócio e metadados envolvidos no desenvolvimento de software (Schiavoni, 2007).

Os Formulários de Entrada de Dados correspondem a interfaces gráficas com o usuário (GUI – *Graphic User Interface*), utilizados para solicitar informações dos usuários. Utilizando essa fonte de dados, o usuário informa, manualmente, o contexto, seja voluntária ou involuntariamente.

Na Figura 5.3, observam-se relações de aquisição. As entidades `ServicoPersitencia` e `IntefaceGUI` foram marcadas com o estereótipo $\langle\langle ContextualSource \rangle\rangle$, representando as fontes de contexto. Para cada elemento contextual, existe uma relação com a sua respectiva fonte, indicando as configurações de aquisição. Essas relações são marcadas com o estereótipo $\langle\langle acquisitionAssociation \rangle\rangle$.

Depois de definidas as fontes de contexto, o processo *Context Process* define que se determine como as fontes de contexto serão implementadas. A implementação dos mecanismos de captura de informações está fora do escopo deste trabalho. Como já citado anteriormente (Seção 4.3), pesquisas já foram iniciadas com o objetivo de definir como as fontes de contexto serão implementadas. Cada fonte de contexto possui uma particularidade, dependendo do tipo de informação que deve prover, o que torna, portanto, a tarefa de projetar a forma de implementação de cada uma delas um trabalho não trivial. Dessa forma, este trabalho limita-se a informar quais serão as fontes, sem se preocupar com a forma como serão implementadas. Esta atividade será realizada, mais detalhadamente, em trabalhos futuros.

Em seguida, a atividade de projetar o processamento consiste em determinar, para cada CE, se é necessário realizar qualquer inferência para determinar o seu valor. Mais uma vez, definir as regras para processar informações contextuais não é um trabalho trivial. Inicialmente, foram definidas as regras e ações para os principais elementos contextuais. Entretanto, este trabalho não tem a pretensão de esgotar a capacidade de processamento e inferência sobre as informações contextuais. Um dos trabalhos em andamento no grupo de pesquisa GEESD foca em explorar essa capacidade mais profundamente, aproveitando a semântica oferecida pela técnica de representação baseada em ontologias. Para este trabalho, as regras foram definidas conforme a seguir, para o foco `Realizar Tarefa`:

Regra 1

- **Condições:** não (Ferramenta.estaInstaladaEm)
- **Ações:** Chamar Foco Instalar Ferramenta

Regra 2

- **Condições:** Ferramenta.estaInstaladaEm `AND` não (Ferramenta.ehUltimaVersao())
- **Ações:** Chamar Foco Instalar Ferramenta

Regra 3

- **Condições:** não (Tarefa.dependeDe(Tarefa))
- **Ações:** Chamar Atividade Executar Tarefa

Regra 4

- **Condições:** Tarefa.dependeDe(Tarefa) `AND` não (Tarefa.temDependenciaNaoFinal()) `AND` RecursoMaterial.disponibilidade

- **Ações:** Chamar Atividade Executar Tarefa

Regra 5

- **Condições:** Tarefa.dependeDe(Tarefa) `AND` não (Tarefa.temDependenciaNaoFinal()) `AND` não (RecursoMaterial.disponibilidade)
- **Ações:** Aguardar Notificação

Regra 6

- **Condições:** Tarefa.dependeDe(Tarefa) `AND` Tarefa.temDependenciaNaoFinal()
- **Ações:** Aguardar Notificação

Depois de definidas as regras gerais para o processamento das informações contextuais, o próximo passo é definir a forma de disseminação dessas informações. Para este trabalho, a disseminação será realizada via gerenciador de notificações. Esse gerenciador será detalhado no Capítulo 7.

A fase de projeto de utilização do contexto (*Context Usage*) consiste em definir como o contexto será efetivamente utilizado. As atividades são realizadas para cada um dos focos identificados no modelo conceitual, na fase de análise (Vieira, 2008).

A primeira atividade consiste em, para cada foco, produzir o modelo comportamental do contexto, relacionando cada CE com as variações de comportamento. Para isso, são utilizadas representações baseadas em grafos contextuais, apresentados na Seção 2.5.1. A Figura 5.4 representa o modelo comportamental para o foco `Realizar Tarefa`. Os demais modelos podem ser encontrados em (Chaves, 2009).

Na Figura 5.4, as variações de comportamento estão representadas por nós de decisão, marcados com o estereótipo $\langle\langle ContextualNode \rangle\rangle$. Quando as ações que ocorrem no ambiente (marcadas com o estereótipo $\langle\langle Action \rangle\rangle$) atingem um $\langle\langle ContextualNode \rangle\rangle$, o ambiente deve decidir, de acordo com o contexto, qual caminho seguir. Os diversos caminhos possíveis estão marcados com o estereótipo $\langle\langle ContextualBranch \rangle\rangle$. Ao final, realizadas todas as ações correspondentes ao caminho escolhido, existe um nó de recombinação ($\langle\langle RecombinationNode \rangle\rangle$), responsável por garantir a consistência do grafo. O nó de recombinação é o ponto em que, independente do caminho (ou *branch*) escolhido, quando o sistema atinge esse ponto, as ações correspondentes àquele contexto foram concluídas.

Conforme visto na Seção 2.5.1, uma ação pode invocar uma ação complexa, modelada em um subgrafo. Para exemplificar, a $\langle\langle Action \rangle\rangle$ `Atividade_Executar Tarefa`, invoca a atividade `Executar Tarefa`, que pode ser observada na Figura 5.5.

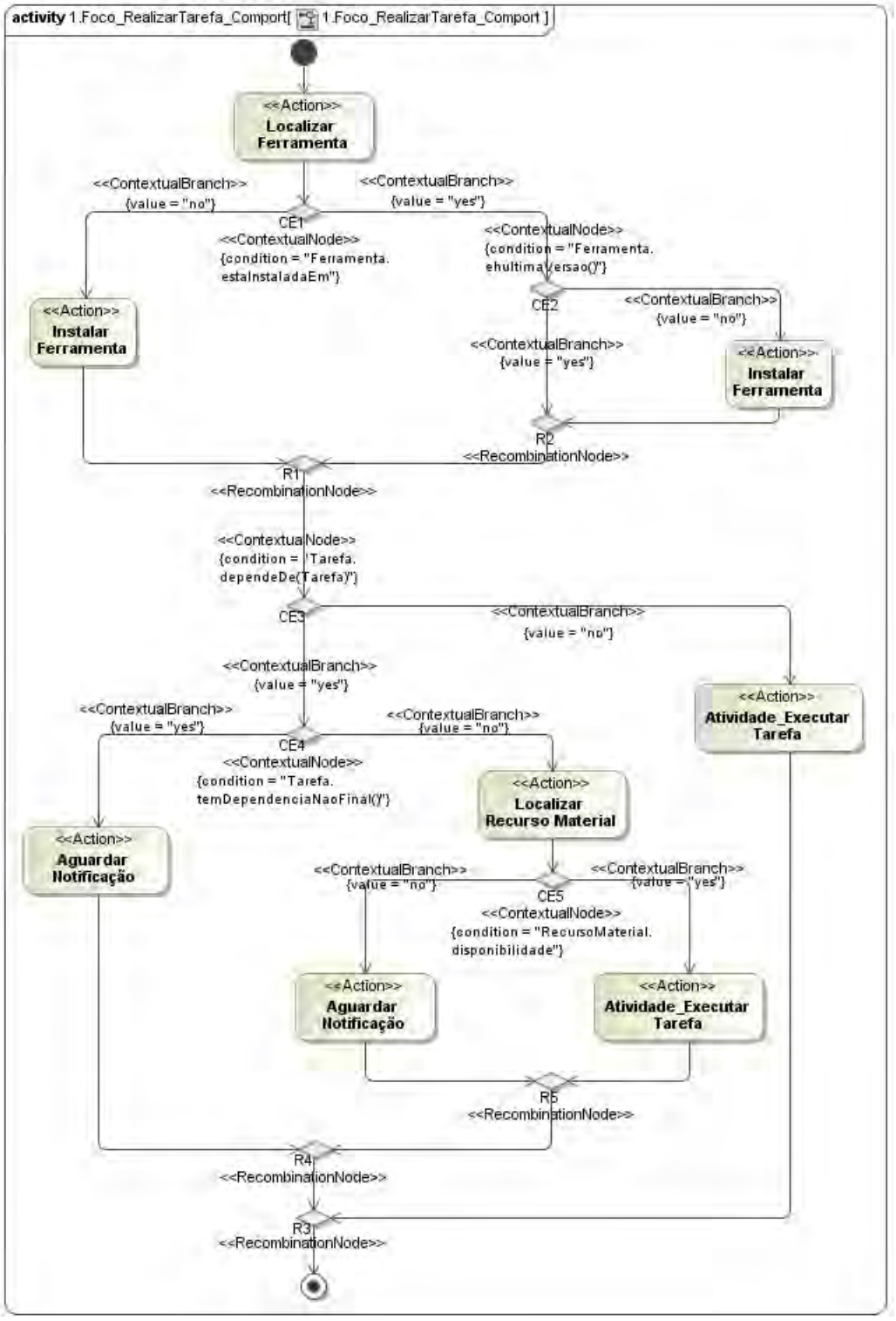

Figura 5.4: Modelo Comportamental – Foco: Realizar Tarefa

As duas últimas atividades do processo consiste em projetar a adaptação e apresentação do contexto. A adaptação considera fatores como privacidade e segurança da

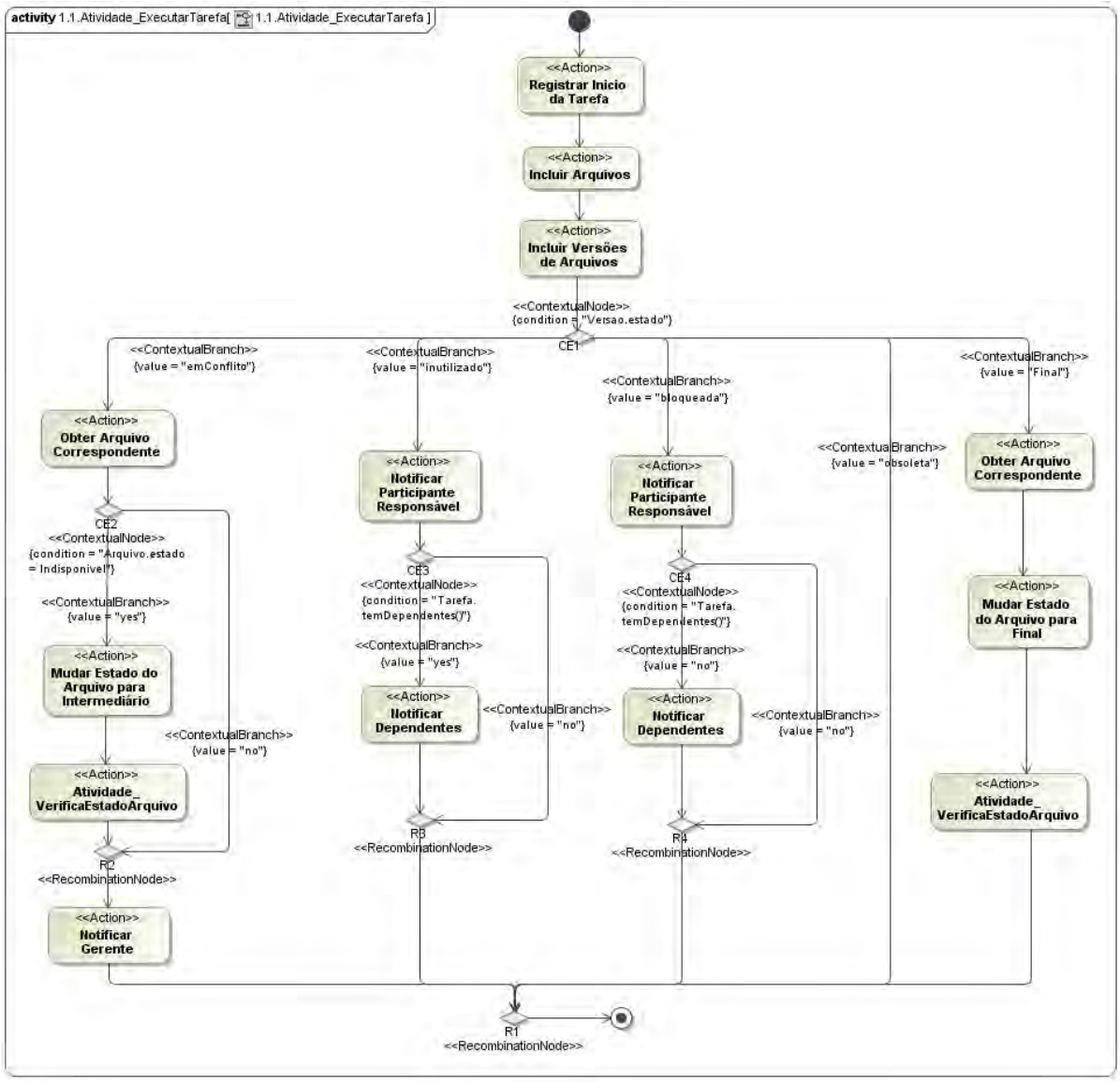

Figura 5.5: Modelo Comportamental – Atividade: Executar Tarefa

informação, intrusão, entre outros, aumentando a confiança dos usuários no sistema e a facilidade cognitiva para compreender as informações recebidas. A apresentação consiste em definir quais métodos de percepção devem ser utilizados para cada informação, a cada momento. Essas atividades exigem esforços no sentido de definir o perfil de cada usuário do sistema e encontrar maneiras de oferecer as informações respeitando as individualidades, não apenas de acordo com os papéis que desempenham. Isso significa conhecer as ações individuais, preferências, facilidade cognitiva, interesse e confiança nas informações que recebem e diversas outras características de cada usuário. Quando se

trata de GSD, torna-se ainda mais complexo definir essas características, já que cada indivíduo é influenciado pela sua cultura e costumes locais.

Devido ao seu alto nível de complexidade, para este trabalho, essas atividades não foram realizadas. Entretanto, o modelo proposto prevê sua futura realização, pelo elemento Mecanismos de apresentação. Quando enviadas pelo gerenciador de notificações para a disseminação, as informações estão prontas para serem disseminadas. Quando alcança o mecanismo de apresentação, uma informação poderá ser oferecida para os usuários utilizando qualquer dos diversos métodos de percepção que podem existir no ambiente. Como as informações serão disseminadas na forma de mensagens, essas mensagens podem ser oferecidas por ferramentas de comunicação como *chat*, *email*, *post-it* anexados ao *workspace* dos usuários, janela *pop-up*, entre outros métodos definidos em (Pozza, 2005), para o ambiente DiSEN. Nos modelos comportamentais apresentados em (Chaves, 2009), é possível observar ações denominadas `Notificar gerente`, `Notificar participante`, `Notificar dependentes`. Cada ação de notificação corresponde à disseminação de informações contextuais, indicando quem deverá receber essa informação (gerente, participante, usuário) naquele contexto. Como pode ser facilmente observado, essa definição baseia-se no papel desempenhado pelos indivíduos para indicar quando a informação será recebida. Em trabalhos futuros, essa definição pode ser alterada para tratar perfis individuais de usuários.

5.3 Considerações Finais de Capítulo

Este capítulo apresentou os passos seguidos para especificar quais elementos e entidades são úteis para identificar o contexto das ações realizadas no ambiente DiSEN, além de suas características e relacionamentos. O objetivo foi modelar o contexto para esse ambiente e definir quais informações são relevantes em cada momento. O processo utilizado foi definido por Vieira (2008). O foco `Realizar Tarefa` foi utilizado para exemplificar os passos do processo. Os demais focos gerados podem ser encontrados em (Chaves, 2009).

A primeira dificuldade nesse ponto foi compreender a instanciação do processo e como as atividades deveriam ser desenvolvidas. A utilização do processo *Context Process*, do metamodelo *Context Metamodel* com o *profile* UML para MagicDraw, propostos por Vieira (2008), facilitaram o desenvolvimento dos modelos por três principais razões: (i) o metamodelo ajudou a compreender a dinâmica do contexto, a diferença entre contexto (dinâmico) e elementos contextuais (estáticos) e como o contexto pode ser utilizado pelos sistemas; (ii) a existência do *profile* para uma ferramenta, ainda que proprietária, ofereceu aparato tecnológico para reduzir o esforço para a construção dos modelos; e

(iii) a existência de um processo permite a construção dos modelos de forma metódica e organizada, conduzindo a especificação em uma ordem lógica e oferecendo padrões de documentação.

Entretanto, algumas atividades do processo não foram realizadas. Isso se deve à complexidade de tratar informações contextuais, agravada pelo fato de o sistema em questão ser um ambiente de desenvolvimento distribuído de software, onde questões culturais e organizacionais, que interferem no contexto dos indivíduos, são bastante críticas. Dessa forma, essa etapa do trabalho concentrou-se em definir o contexto das ações realizadas no software (ambiente DiSEN), como essas informações são adquiridas e de que forma influenciam no contexto (variações de comportamento). A disseminação de informações (ações de notificação) são baseadas nos papéis que os indivíduos desempenham no ambiente. As demais atividades serão realizadas, mais detalhadamente, em trabalhos futuros.

Sabendo-se quais informações são relevantes para o contexto, iniciou-se a definição de como essas informações seriam representadas. O Capítulo 6 descreve a técnica de representação e a forma como foi especificada.

Capítulo

6

Representação de Contexto

6.1 Considerações Iniciais

Um dos elementos do modelo DiSEN-CSE é a Representação de Contexto. Esse elemento tem por função identificar as informações contextuais capturadas e representá-las de forma que, mesmo com entidades receptoras diferentes, a compreensão semântica sobre o que está sendo informado seja a mesma.

Conforme visto no Capítulo 2, diferentes técnicas de representação foram estudadas, com o objetivo de definir qual ofereceria maior eficácia para os propósitos do modelo DiSEN-CSE. Analisando características como expressividade, formalismo, capacidade de inferência e ferramentas disponíveis, a técnica que se destacou foi a representação baseada em ontologias.

De acordo com Nunes et al. (2007), a motivação para usar ontologias na formalização de modelos de contexto está em:

- compartilhar um entendimento comum sobre a estrutura da informação, entre pessoas ou agentes computacionais;
- permitir reutilização dentro do domínio de conhecimento;
- tornar explícitas as concepções acerca do domínio;
- analisar o conhecimento do domínio;

- permitir o compartilhamento e a reutilização do conhecimento do domínio; e
- permitir o uso de mecanismos de inferência para raciocinar sobre vários contextos.

Outra vantagem do uso de ontologias, destacada em (Neto, 2006), é descrever um domínio modelando o conhecimento sem qualquer compromisso com a implementação de um sistema de software. Além disso, permite a interoperabilidade entre os sistemas que a utilizam.

Por essas razões, esse capítulo descreve o projeto de uma ontologia para o domínio do desenvolvimento distribuído de software, mais especificamente voltada para o ambiente DiSEN. O objetivo é descrever as entidades e os elementos contextuais que devem ser representados e disseminados pelo modelo DiSEN-CSE. Antes de iniciar os detalhes sobre o projeto da ontologia, uma breve seção apresenta as definições de ontologia e algumas características relevantes para a compreensão dessa etapa do trabalho.

6.2 Ontologias

O termo ontologias surgiu na Filosofia, para descrever as coisas do mundo real. Nos últimos anos, conquistou espaço no domínio da computação devido ao papel que desempenha, especialmente, nas áreas de Inteligência Artificial e Linguística Computacional. Nessas áreas, a definição mais aceita para o termo foi cunhada por Gruber (1993), que afirma ser ontologias uma especificação formal e explícita de uma conceitualização. Para o autor, uma conceitualização corresponde a uma visão abstrata e simplificada do mundo que se deseja representar.

De acordo com Noy e McGuiness (2001), uma ontologia define um vocabulário comum para indivíduos ou sistemas que precisam compartilhar informações sobre um domínio. Entretanto, de acordo com Theresa Edgington e Vinze (2004), uma ontologia não é simplesmente um vocabulário ou uma taxonomia[1]. Falbo (1998) afirma que, idealmente, uma ontologia não deve ser uma simples hierarquia de termos, mas uma infraestrutura teórica que verse sobre o domínio. Ontologias mais complexas incluem axiomas que aumentam a complexidade das relações, conceitos e restrições, oferecendo a interpretação desejada àquele domínio. Para esse trabalho, a definição utilizada é a proposta em (Gruber, 1993), já que o objetivo da ontologia desse trabalho consiste em representar formal e explicitamente os conceitos existentes no ambiente DiSEN, oferecendo semântica a esses conceitos e, consequentemente, diminuindo ambiguidades e incompreensões.

[1]classificação, baseada em similaridades de estrutura, origem, entre outros (Fonte: http://wordnet.princeton.edu)

Uma ontologia é composta por um conjunto de entidades[2], que representam os conceitos do domínio (Noy e McGuiness, 2001) e podem ser organizadas hierarquicamente. As entidades possuem propriedades, que correspondem às características e atributos que as identificam. Além disso, essas propriedades podem ter restrições, para aumentar a precisão da especificação. Cada entidade possui uma população de indivíduos, que representam as instâncias dos conceitos. Cada indivíduo, portanto, possui as mesmas propriedades e respeita as mesmas restrições da entidade a que pertence.

Segundo Guarino (1997), as ontologias podem ser classificadas quanto ao nível de dependência sobre um ponto de vista ou tarefa em particular, sendo distinguidas em:

ontologia de alto-nível: descreve conceitos muito gerais como espaço, tempo, objetos, entre outros, que são independentes de um problema ou domínio em particular;

ontologia de domínio: descrevem o vocabulário relacionado a um domínio específico, como medicina, comércio eletrônico, entre outros;

ontologia de tarefa: descreve o vocabulário relacionado a uma tarefa ou atividade específica, como diagnóstico para a medicina, vendas para um comércio eletrônico;

ontologia de aplicação: descreve os conceitos dependendo do domínio e da tarefa ao mesmo tempo.

A ontologia desenvolvida nesse trabalho se encaixa na classificação de ontologias de domínio, por descrever os conceitos relacionados ao desenvolvimento distribuído de software. Entretanto, desenvolver uma ontologia de domínio pode ter um alto custo (Falbo, 1998; Martimiano, 2006). Assim, surgiram diversas metodologias, conforme discutidas em (Simperl e Tempich, 2006), que tem como objetivo conduzir e auxiliar a construção de ontologias. Contudo, não existe uma metodologia considerada padrão para o desenvolvimento (Martimiano, 2006; Simperl e Tempich, 2006).

Dessa forma, a metodologia utilizada para a construção da ontologia para o ambiente DiSEN corresponde a uma intersecção entre os passos seguidos em trabalhos encontrados na literatura (Fernández et al., 1997; Martimiano, 2006; Noy e McGuiness, 2001; Roberto, 2006), que serviram como base teórica para o desenvolvimento desta etapa do trabalho.

A Seção 6.3 apresenta os passos da metodologia utilizada para o desenvolvimento da ontologia e um exemplo dos artefatos gerados em cada etapa. A tecnologia utilizada

[2]Também denominadas conceitos ou classes.

para a construção da ontologia foi a linguagem OWL-DL[3] (*Web Ontology Language - Description Logics*) e a ferramenta Protégé 4.0(beta)[4].

A linguagem OWL possui três sublinguagem (OWL *Lite*, OWL DL e OWL *Full*), diferentes entre si no nível de formalismo e capacidade de expressão. A OWL-DL foi escolhida por possuir o mesmo conjunto de construtores oferecidos pela OWL *Full*, embora com algumas restrições de uso, e por oferecer expressividade e decidibilidade, favorecidas pelo uso da lógica descritiva.

A Protégé é uma ferramenta *free* e *opensource* para desenvolvimento de ontologias. A versão 4.0(beta) oferece apoio para a criação de ontologias utilizando a linguagem OWL e possui uma interface direta com os raciocinadores Pellet[5] e FaCT++[6].

6.3 OntoDiSEN

O desenvolvimento da ontologia OntoDiSEN iniciou com a definição do planejamento, conforme abaixo:

Definição do domínio: ambiente de desenvolvimento distribuído de software – DiSEN;

Definição do objetivo principal: representar de forma não ambígua as informações relacionadas ao contexto das ações de indivíduos, locais e ferramentas de um ambiente de desenvolvimento distribuído de software, mais especificamente, o DiSEN;

Definição dos usuários: os usuários da ontologia são os usuários do ambiente DiSEN e as ferramentas disponíveis no próprio ambiente;

Definição dos recursos: ferramenta para modelagem (Protégé 4.0) e linguagem para modelagem (OWL-DL);

Definido o planejamento, o próximo passo foi a aquisição de conhecimento. Para isso, foi necessário pesquisar nas diversas fontes de conhecimento, informações relevantes sobre o domínio. As técnicas utilizadas para essa aquisição foram: *brainstorming*, entrevistas com especialistas do domínio (*stakeholder* não usuário, definido no Apêndice A) e análise de textos formais contendo informações sobre o ambiente DiSEN, como (Huzita et al., 2007; Pascutti, 2002; Pozza, 2005; Schiavoni, 2007), entre outros.

[3]http://www.w3.org/TR/owl-features
[4]http://protege.stanford.edu
[5]http://clarkparsia.com/pellet
[6]http://owl.man.ac.uk/factplusplus

Com base no conhecimento adquirido por essas fontes, definiu-se uma série de questões de competência, que representam exemplos de questões que a ontologia precisa responder, facilitando a fase de definição de conceitos. Essas questões não tem a pretensão de esgotar a capacidade de raciocínio da OntoDiSEN, mas apenas auxilia a definição de quais conceitos serão mais importantes para representar o conhecimento do domínio. Para definir as questões de competência, o estudo concentrou-se nos elementos de percepção (quem, o que, quando, onde, porque e como) e nos elementos que auxiliam a definição do contexto (localidade, identidade, atividade, tempo e presença). Além disso, as questões estão centradas nas três entidades ativas do ambiente: usuários, locais e ferramentas, conforme descrito no Capítulo 4. O conjunto completo das questões de competência foram definidas incrementalmente, durante todo o desenvolvimento, e podem ser encontradas no Apêndice C. Algumas das principais questões são:

Questões de competência para a entidade `Usuario`:

- Em que local o usuário está logado? (localidade)
- Quem é o usuário? (identidade)
- Que papéis o usuário desempenha? (identidade)
- Quais tarefas o usuário realiza? (atividade)
- De quais projetos o usuário participa? (atividade)
- Qual o estado do projeto do usuário? (atividade)
- Quais artefatos o usuário produz? (atividade)
- Quais ferramentas um usuário utiliza? (atividade)
- Quando iniciou/concluiu um artefato? (tempo)
- Quais usuários do mesmo projeto estão logados? (presença)

Questões de competência para a entidade `Ferramenta`:

- Em quais locais está disponível? (localidade)
- Em quais *workspaces* está instalada? (localidade)
- Qual é a versão? (Identidade)
- Que tipo de artefatos gera? (identidade)

- Qual projeto inclui a utilização dessa ferramenta? (atividade)
- Quem está utilizando a ferramenta agora? (presença)
- Qual a data da sua última atualização? (tempo)

Questões de competência para a entidade `Locais`:

- Quem está conectado? (identidade)
- Quem é o administrador? (identidade)
- Há quanto tempo o local está disponível? (tempo)
- Qual dos locais permanece mais tempo disponível? (tempo)

Definidas as questões de competência, foi elaborado um texto que descreve, de maneira geral, a semântica do ambiente DiSEN. O objetivo era descrever a idéia geral do domínio e destacar os principais conceitos relacionados às questões de competência, sendo como segue:

*"Os **recursos** no ambiente DiSEN estão divididos em três especializações: **usuários**, **ferramentas** e **locais**. Esses três tipos de recursos possuem os atributos genéricos **nome**, **descrição** e **estado**, sendo que o estado pode ser definido como **disponível** ou **indisponível**.*

*Um usuário é todo indivíduo, disponível em uma **localidade**, que pode acessar um **workspace** (está conectado a um local) utilizando um **login** e **senha** definidos e respeitando algumas permissões. Os usuários têm afinidades com outros usuários, realizam **treinamentos** e possuem/adquirem **habilidades** que são importantes para a realização de **tarefas** nos **projetos**. Seu estado (disponível ou indisponível) está diretamente relacionado à sua agenda. Quando um usuário está disponível em um determinado período, pode ser alocado por um gerente para atuar em um projeto, tornando-se um **participante**. Quando é participante de um projeto, o usuário pode ser alocado para realizar uma tarefa, se possuir as habilidades e **conhecimentos** necessários e se o seu período de disponibilidade for coincidente com o **período** de realização da tarefa. Caso seja alocado para a tarefa, o usuário se torna indisponível para aquele período e adquire um **papel** (função) dentro do projeto. Os participantes podem ser excluídos do projeto. Quando isso ocorre, é necessário que o gerente de projetos aloque outro participante para a tarefa e registre os motivos da exclusão.*

*O produto de uma tarefa é um **artefato**, do tipo especificado pela **atividade** (uma atividade possui um **tipo de artefato**). Os artefatos podem ser expressos em diversos*

***arquivos**, cada qual podendo conter diversas **versões**. Um usuário pode ser responsável por uma ou mais versões de arquivos, sendo que nem sempre o usuário responsável por uma versão é responsável pelas outras do mesmo arquivo. Os usuários produzem seus artefatos, utilizando **ferramentas**. Para construir seus artefatos, o usuário pode precisar de artefatos construídos em outras tarefas, por outros usuários. Nesse caso, diz-se que a tarefa possui uma restrição ou depende de outra tarefa. Os usuários precisam informar quando iniciam e terminam a execução das suas tarefas. Além disso, precisam conhecer o **estado das versões, dos arquivos e dos artefatos** que produzem, bem como os estados dos artefatos dependentes.*

Quando participam de um projeto, o usuário faz parte de uma equipe (grupo). As equipes possuem interesses comuns como informações sobre o projeto, sobre a tecnologia utilizada, sobre a presença de outros membros da equipe no ambiente. Os usuários podem trocar mensagens entre si para resolver conflitos em artefatos, obter informações sobre os projetos, entre outros.

*Um projeto possui um **objetivo**, **datas de início e término (previstas e efetivas)**, um **estado** atual e segue o **fuso horário** de uma determinada localidade. Além disso, um projeto instancia um **processo**. Um processo é uma seqüência de passos definidos para o desenvolvimento de um produto de software. Possui um conjunto de **fases**, que por sua vez são divididas em **atividades**. O projeto é guiado por um processo, mas pode ser customizado para adicionar ou desabilitar fases e atividades, dependendo do problema a ser resolvido. Deve-se registrar os motivos pelos quais uma fase ou atividade não será realizada.*

Um processo que já foi instanciado não pode ser excluído, mesmo que os projetos que o utilizam não estejam mais em andamento. Caso o gerente decida pela não utilização de um processo em projetos futuros, este poderá ser desativado e deve-se registrar os motivos da desativação do processo.

*O **projeto, as fases e as atividades** também possuem **estados**. Os usuários precisam conhecer o estado do projeto em que é participante, bem como das fases e atividades nas quais desempenham algum papel, ou em fases e atividades dependentes destas.*

*Uma ferramenta está disponível em um local. Os usuários podem acessar as ferramentas e instalá-las em seu workspace. As ferramentas possuem **versão**, **tamanho** em bytes, **plataforma**, **fornecedor** e uma listagem dos tipos de artefatos que é capaz de gerar. Uma ferramenta pode depender de outras ferramentas, módulos ou bibliotecas. Nesse caso, quando instalada, as partes de que dependem também devem ser instaladas. Os projetos podem indicar quais ferramentas utilizar para construir os artefatos, de acordo*

com os tipos de artefatos que é capaz de produzir. Por questões de compatibilidade, é necessário saber em quais ferramentas e versões foram construídos os artefatos, por quais usuários.

As ferramentas podem ser atualizadas no repositório central (um local). Quando isso ocorre, é necessário saber quem são os usuários que utilizam essa ferramenta e comunicá-los da existência de uma nova versão.

O recurso local está disponível em uma localidade e possui uma ***configuração local****, expressa por seu* ***endereço****,* ***porta****,* ***login****,* ***senha****,* ***driver*** *e* ***fornecedor****. Possui também um usuário responsável (****administrador****), a lista de serviços que oferece e a lista de usuários conectados nele. Os locais podem também ser repositório de ferramentas. Nesse caso, deve indicar quais são as ferramentas disponíveis. O estado de um recurso do tipo local é modificado pelo administrador. Um local está sempre disponível, a menos que ocorra algum evento que o desconecte do ambiente. Nesse caso, deve haver uma forma de comunicar ao sistema que aquele local (com todos os serviços, ferramentas e usuários) tornou-se indisponível.*

Os recursos como papel, mídias, equipamentos, entre outros, são ***recursos materiais****, contendo nome, descrição, custo e, quando se trata de equipamentos, versão, fabricante e configuração. Os recursos materiais são de uma localidade ou podem estar disponíveis em um local. Possuem um* ***tipo de recurso*** *e são utilizados nos projetos, mais especificamente, nas tarefas."*

Com base nesse texto, um conjunto de entidades e seus relacionamentos foram definidos e um mapa conceitual foi elaborado, conforme a Figura 6.1. O mapa conceitual foi construído na ferramenta CmapTools[7] v4.18.

A partir de então, a ontologia foi formalizada utilizando a linguagem OWL DL e a ferramenta Protégé. Os conceitos e relacionamentos foram transcritos na forma de entidades e propriedades, como representadas nas Figuras 6.2 e 6.3, repectivamente.

Algumas entidades possuem atributos. Em OWL, as propriedades são divididas em propriedades de objetos (relacionamentos) e propriedades de dados (atributos). Nas propriedades de objeto, estabelece-se uma relação entre duas entidades, sendo uma imagem (entidade que possui a propriedade) e um domínio (faixa de valores possíveis para a propriedade). Nas propriedades de dados, o domínio não é uma entidade, mas um tipo de dado, como *string, char, integer, date, float*, entre outros. A Figura 6.4 apresenta as propriedades de dados definidas para esta ontologia.

A seguir, são apresentadas as entidades, seus atributos (propriedades de dados) e as propriedades de objetos na qual cada entidade é domínio:

[7]http://cmap.ihmc.us

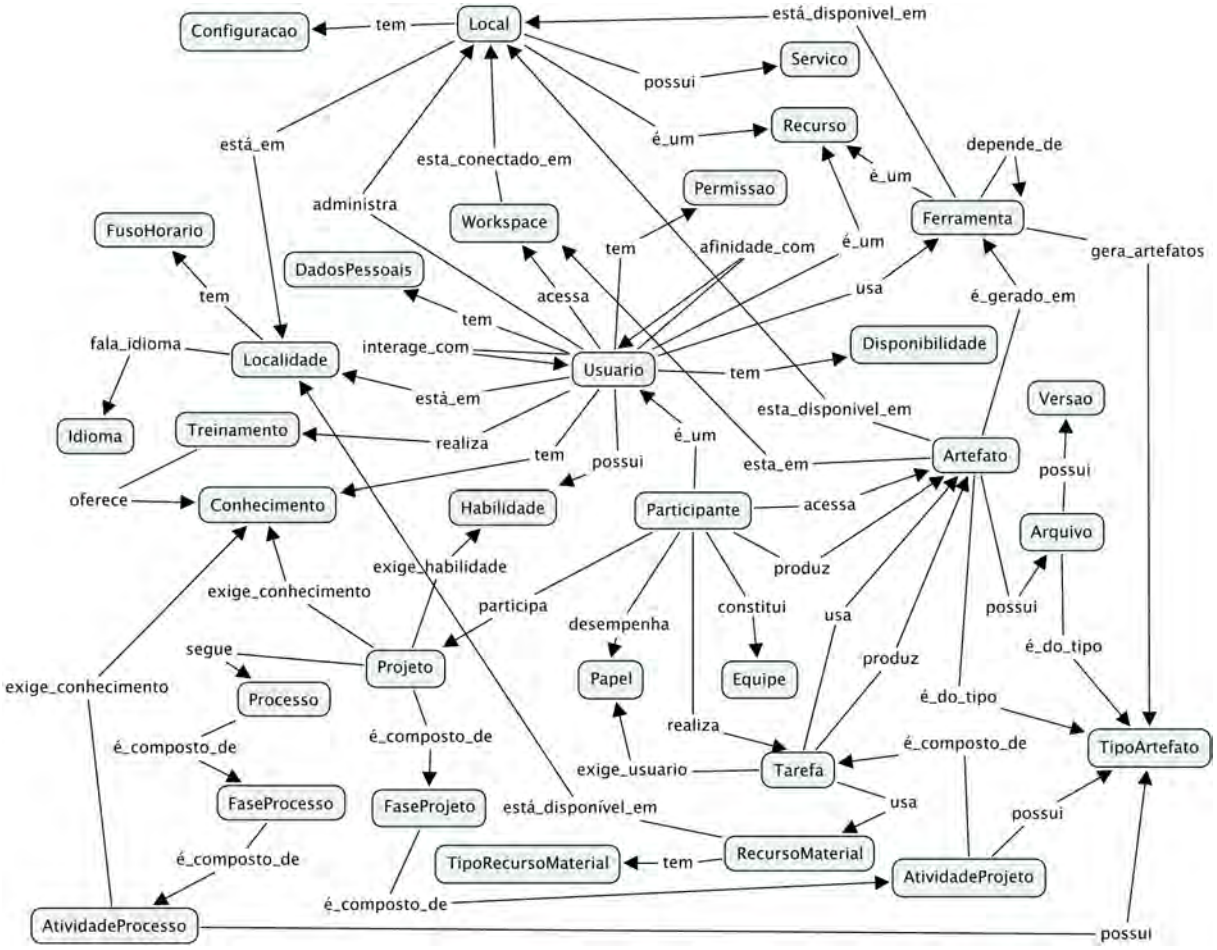

Figura 6.1: Mapa Conceitual – principais conceitos e relacionamentos da OntoDiSEN.

`Arquivo`: arquivos contendo os artefatos gerados pelo sistema. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio das propriedades `pertenceAArtefato`, indicando que cada arquivo corresponde a um determinado artefato, `temEstadoArquivo`, indicando que possui um estado e `temVersao`, indicando que todo arquivo pode possuir n versões. O estado do arquivo é definido pela entidade `EstadoArquivoValuePartition`.

`Artefato`: resultado de uma tarefa, que pode ser usado como matéria-prima para outras tarefas. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio das propriedades `acessadoPorParticipante` para indicar que um artefato pode ser visualizado pelos participantes do projeto, `disponivelEmLocal`, indicando que o artefato precisa estar disponível para acesso em algum repositório, `temArquivo`, indicando quais arquivos formam aquele artefato, `ehDoTipoArtefato`, que representa o grupo de artefatos a que pertence, `estaEmWorkspace` para indi-

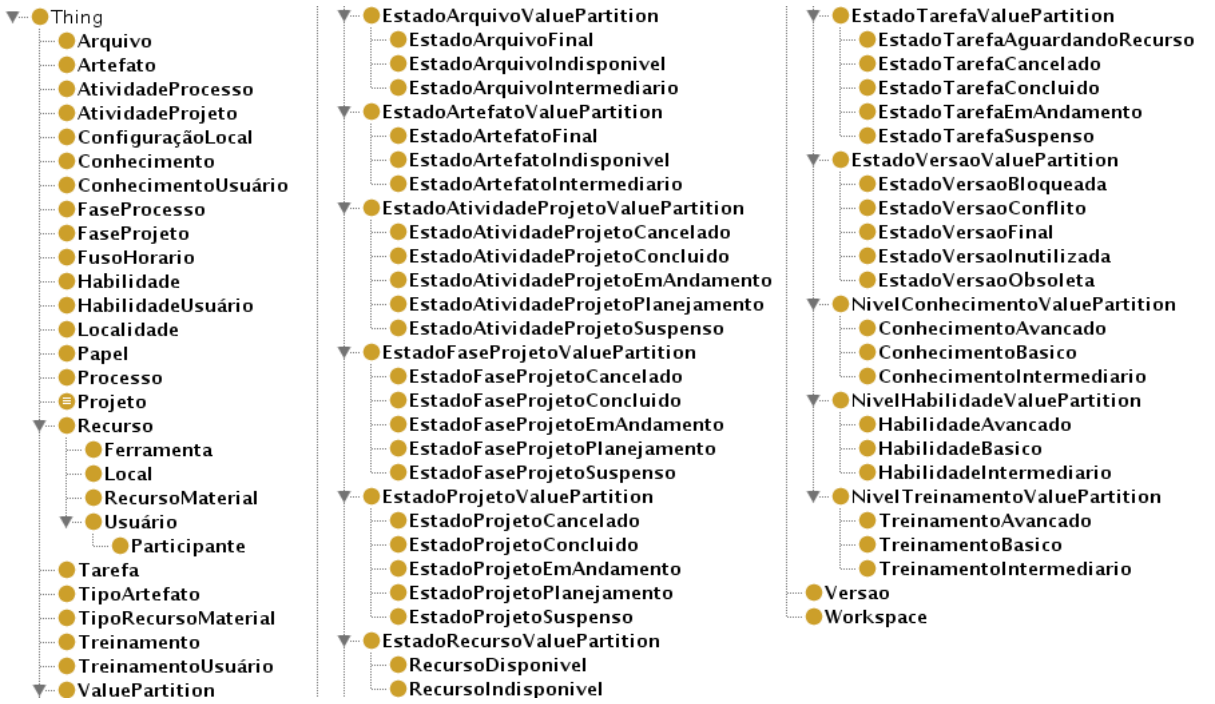

Figura 6.2: Entidades da OntoDiSEN, modeladas na ferramenta Protégé.

car em quais *workspaces* foi baixado, `geradoEmFerramenta`, para saber em qual ferramenta foi produzido, `pertenceATarefa` para indicar a tarefa a que pertence, `produzidoPor`, indicando qual participante é responsável pelo artefato e, por fim, para indicar que possui um estado, `temEstadoArtefato`. O estado de um artefato é definido pela entidade `EstadoArtefatoValuePartition`.

`AtividadeProcesso`: atividade que compõe uma fase de um processo. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio da propriedade `dependeDeAtividadeProcesso`, para indicar que uma atividade pode depender de outra para ser realizada.

`AtividadeProjeto`: atividade que compõe uma fase de um projeto, geralmente baseada em uma atividade definida em um processo. Possui como atributos as propriedades de dados `nome`, `descricao`, `dataInicioPrevisto`, `dataInicio`, `dataFim` e `dataFimPrevisto`. É domínio das propriedades `dependeDeAtividadeProjeto`, indicando que uma atividade pode depender de outra para ser realizada e, para indicar que a atividade possui um estado, `temEstadoAtividadeProjeto`. O estado de uma atividade é definido pela entidade `EstadoAtividadeValuePartition`.

Figura 6.3: Propriedades de objetos da OntoDiSEN, modeladas na ferramenta Protégé.

`ConfiguraçãoLocal`: representa as características que identificam um local. Possui como atributos as propriedades de dados `url`, `senha`, login, `porta`, `fornecedor` e `driver`.

`Conhecimento`: indica os conhecimentos que um usuário do ambiente possui. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio das propriedades `adquiridoPorTreinamento` para indicar que o conhecimento pode ser adquirido quando um usuário realiza um treinamento, `necessarioEmProjeto`, indicando que um conhecimento é útil em determinados projetos específicos.

`ConhecimentoUsuario`: estabelece o conhecimento pertencente a um usuário, especificamente, indicando o nível de conhecimento que aquele usuário possui. É domínio das propriedades `possuiConhecimento`, que representa o conhecimento que o usuário possui, `temNivelConhecimento`, indicando qual o nível de conhecimento o usuário possui e `possuiUsuario`, que representa qual usuário possui o conhecimento. A en-

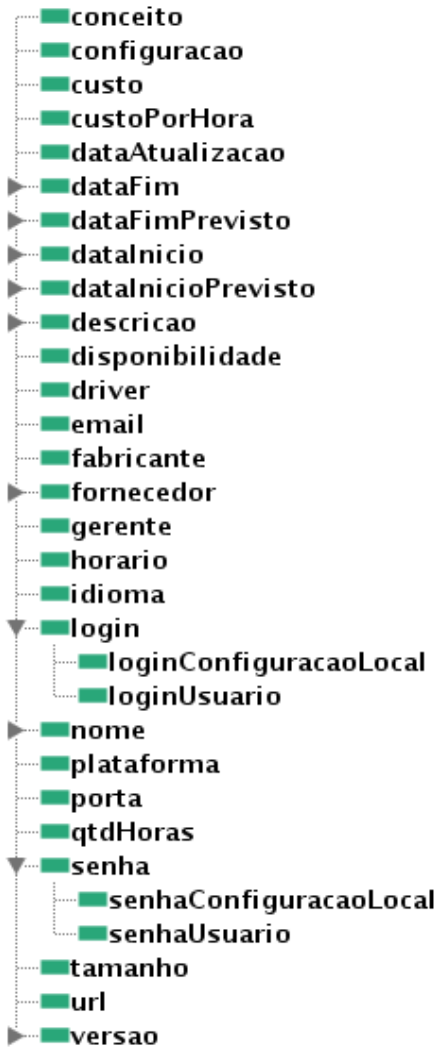

Figura 6.4: Propriedades de dados da OntoDiSEN, modeladas na ferramenta Protégé.

tidade `NivelConhecimentoValuePartition` define os niveis de conhecimento existentes.

`FaseProcesso:` estabelece a divisão do processo em partes gerenciáveis. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio das propriedades `dependeDeFaseProjeto`, indicando que uma fase pode depender de outra para ser realizada e *temAtividadeProcesso* para indicar que uma fase é composta de atividades.

`FaseProjeto:` estabelece a divisão do projeto em partes gerenciáveis, geralmente seguindo a especificação de um processo. Possui como atributos as propriedades de dados `nome` e `descricao`. É domínio das propriedades `dependeDeFaseProjeto`, indicando

que uma fase pode depender de outra para ser realizada, *temAtividadeProjeto*, para indicar que uma fase é composta de atividades e `temEstadoFaseProjeto`, indicando que uma fase possui um estado. O estado da fase é determinado pela entidade `EstadoFaseValuePartition`.

`FusoHorario:` indica o deslocamento de uma determinada localidade com relação ao marco zero.

`Habilidade:` capacidades que os usuários possuem, que favorece o desenvolvimento de alguma tarefa específica. Possui como atributos as propriedades de dados `nome` e `descricao`.

`HabilidadeUsuario:` estabelece a habilidade de um determinado usuário, e o nível de habilidade que aquele usuário possui. É domínio das propriedades `possuiHabilidade`, que representa a habilidade que o usuário possui, `possuiUsuario`, indicando o usuário que possui a habilidade e `temNivelHabilidade`, indicando qual nível de habilidade o usuário possui. Os niveis de habilidade são definidos pela entidade `NivelHabilidadeValuePartition`.

`Localidade:` lugar físico (geográfico) em que algum recurso pode estar. Possui como atributos a propriedade de dados idioma. É domínio das propriedades `temFusoHorario`, indicando o fuso horário que aquela localidade segue e *temUsuario*, indicando que uma localidade pode possuir diversos usuários.

`Papel:` função que um participante desempenha em uma determinada tarefa. Possui como atributos as propriedades de dados `nome` e `descricao`.

`Processo:` conjunto de ações pré-determinadas que devem ser seguidas para o desenvolvimento de um produto de software. Possui como atributos as propriedades de dados `nome`, `descricao` e `versao`. É domínio das propriedades `seguidoPor`, indicando que os processos são seguidos por projetos e `temFaseProcesso`, indicando que um processo é composto de fases.

`Projeto:` instância de um processo, incluindo detalhes específicos da execução do processo, como os objetivos que devem ser atingidos com sua realização, prazos e metas a serem cumpridos. Possui como atributos as propriedades de dados `nome`, `descricao`, `dataInicio`, `dataInicioPrevisto`, `dataFim`, `dataFimPrevisto`. É domínio das propriedades `temParticipante`, indicando que um projeto possui participantes alocados para ele, `temFaseProjeto`, indicando que um projeto é composto de fases, `temEstadoProjeto`, para indicar que um projeto possui um estado,

`segueProcesso`, indicando que o projeto deve ser conduzido por um processo que segue, `segueFusoHorario`, para indicar que um projeto respeita um fuso horario especfico, `exigeHabilidade` e `exigeConhecimento`, indicando que, para a concretização de um projeto, serão necessários participantes com algumas habilidades e conhecimentos específicos. O estado do projeto é definido pela entidade `EstadoProjetoValuePartition`.

`Recurso:` todo material humano, tecnológico ou auxiliar (material de escritório, por exemplo), que pode ser alocado para participação ou utilização em um projeto. É especializado em `Ferramenta`, `RecursoMaterial`, `Local` e `Usuario`. Possui como atributo a propriedade de dados `disponibilidade`. É domínio da propriedade `temEstadoRecurso`, que indica que todos os recursos possuem um estado. O estado do recurso é definido pela entidade `EstadoRecursoValuePartition`.

`Ferramenta:` especialização de `Recurso`, consiste nas ferramentas de software que podem ser instaladas no ambiente e utilizadas para a geração dos artefatos. Possui como atributos as propriedades de dados `conceito`, `dataAtualizacao`, `fornecedor`, `nome`, `plataforma`, `tamanho`, `versao`. É domínio das propriedades `geraArtefato`, para indicar que um determinado artefato é gerado por uma determinada ferramenta, `dependeDeFerramenta`, indicando que uma ferramenta pode depender de outra para seu funcionamento correto, `disponivelEmLocal`, indicando a existência de um repositório de ferramentas para torná-las acessíveis aos indivíduos que desejam utilizá-las, `geraArtefatoDoTipo`, indicando que uma ferramenta pode gerar artefatos de determinados tipos específicos, `instaladaEmWorkspace`, indicando em quais *workspaces* a ferramenta se encontra instalada e `utilizadaPorUsuario`, indicando quais usuários utilizam a ferramenta.

`Local:` especialização de `Recurso`, são máquinas ou equipamentos de *hardware* que oferecem serviços ao ambiente, por exemplo, para funcionar como repositório de dados, artefatos ou ferramentas, para conectar *workspaces*, para interligar servidores. Possui como atributo a propriedade de dados `descricao`. É domínio das propriedades `temConfiguracao`, indicando que cada local possui a sua configuracao, `administradoPorUsuario`, indicando que cada local possui um usuário responsável por garantir a disponibilidade do local e a garantia dos serviços oferecidos por ele, `estaEmLocalidade`, para indicar que um local está disponível em um lugar físico (localidade), `temArtefato`, indicando que um local pode constituir um repositório de artefatos, *temFerramenta*, indicando que um local pode constituir um repositório de

ferramentas, `temRecursoMaterial`, indicando que um local pode possuir recursos materiais utilizados por um projeto (por exemplo, um servidor de impressão) e `temWorkspace`, indicando que usuários acessam o ambiente conectando seu *workspace* em um local.

`RecursoMaterial:` especialização de `Recurso`, são todos os demais recursos utilizados no projeto, que não sejam ferramentas, usuários ou locais, por exemplo, impressoras, computadores, mídias, entre outros. Possui como atributos as propriedades de dados `configuracao`, `custo`, `descricao`, `fabricante`, `nome`, `versao`. É domínio das propriedades `utilizadoEmTarefa`, indicando que um recurso material é consumido por uma tarefa, `ehDoTipoRecursoMaterial`, indicando em que tipo de recurso se enquadra (equipamentos, mídias, materiais de escritório, ou outros que podem haver), `disponivelEmLocal`, para apontar em que local o recurso está disponível e `disponivelEmLocalidade`, para apontar em que localidade o recurso está disponível.

`Usuario:` especialização de `Recurso`, são indivíduos que possuem acesso ao ambiente, podendo ser alocados a projetos, nesse caso, especializados em `Participantes`. Possui como atributos as propriedades de dados `nome`, `senha`, `login`, `custoPorHora`, `email`. É domínio das propriedades `usaFerramenta`, indicando que um usuário pode baixar e instalar ferramentas no seu *workspace*, `estaEmLocalidade`, indicando que um usuário pertence a uma localidade, `administraLocal`, indicando que o usuário é responsável pelo funcionamento de um determinado local e `acessaWorkspace`, indicando que o usuário acessa o ambiente utilizando um *workspace*.

`Participante:` especialização de `Usuario`, participantes são usuários que fazem parte de um projeto. Um usuário pode estar alocado para mais de um projeto ao mesmo tempo. Dessa forma, se o usuário é participante em dois projetos, no contexto desses projetos é considerado participante. Nos demais projetos, é considerado um usuário padrão. Possui como atributo a propriedade de dados `gerente`. É domínio das propriedades `realizaTarefa`, indicando que possui uma tarefa sob sua responsabilidade, `produzArtefato`, para indicar que um participante pode gerar um artefato quando realiza uma tarefa, `possuiAcessoArtefato`, indicando que um participante pode acessar um conjunto de artefatos relacionados ao seu projeto ou à sua tarefa, `participaDeProjeto`, para apontar de quais projetos um usuário é participante, `geraVersao`, para indicar que o participante é responsável por uma

determinada versão de artefato e `desempenhaPapel`, indicando que um participante pode possuir um ou mais papéis específicos dentro de um projeto.

`Tarefa:` subdivisão das atividades do projeto (uma atividade do projeto é composta por um conjunto de tarefas), em que cada tarefa é uma unidade atômica, que gera um artefato específico. Possui como atributos as propriedades de dados `nome`, `descricao`, `dataInicio`, `dataInicioPrevisto`, `dataFim`, `dataFimPrevisto`. É domínio das propriedades `temTarefaDependente`, indicando que uma tarefa pode possuir tarefas dependentes, `dependeDeTarefa`, indicando que uma tarefa pode depender de outra tarefa para a realização dos seus artefatos, `usaArtefato`, para indicar que uma tarefa pode utilizar artefatos de outras tarefas para sua realização, `realizadaPorParticipante`, para indicar que uma tarefa possui participantes responsáveis pela sua execução, `possuiArtefato`, para indicar que uma tarefa deve produzir um artefato, `exigeRecursoMaterial`, indicando que uma tarefa necessita de recursos materiais para a geração dos seus artefatos, `exigePapel`, para indicar que uma tarefa necessita de participantes que desempenhem um papel específicico, `exigeFerramenta`, para indicar que os artefatos de uma tarefa serão gerados utilizando uma determinada ferramenta e `temEstadoTarefa` para indicar que as tarefas possuem estados. A entidade `EstadoTarefaValuePartition` é quem define o estado da tarefa.

`TipoArtefato:` indica as entidades ou categorias de artefatos que podem existir. Possui como atributos as propriedades de dados `nome`, `descricao`. É domínio da propriedade `geradoPorFerramenta`, para indicar quais ferramentas geram artefatos de um determinado tipo.

`TipoRecursoMaterial:` indica as entidades ou categorias de recursos materiais que podem existir. Possui como atributos as propriedades de dados `nome`, `descricao`.

`Treinamento:` cursos preparatórios que os indivíduos realizam com o objetivo de adquirir conhecimentos sobre determinadas àreas. Possui como atributos as propriedades de dados `nome`, `descricao`, `qtdHoras`. Para indicar que os treinamentos oferecem conhecimentos aos usuários que os realizam, a entidade Treinamento é domínio da propriedade `ofereceConhecimento`.

`TreinamentoUsuario:` apresenta os treinamentos que um determinado usuário realizou em um determinado período, indicando o nível do treinamento realizado. Possui como atributo, as propriedades de dados `dataInicio`, `dataFim`. É domínio

das propriedades `possuiTreinamento`, que representa o treinamento realizado pelo usuário, `possuiUsuario`, que representa qual usuário realizou aquele treinamento e `temNivelTreinamento`, indicando qual nível de treinamento foi realizado. Os niveis de treinamento são definidos pela entidade `NivelTreinamentoValuePartition`.

`Versao:` versao dos arquivos que correspondem a um artefato, gerado em uma tarefa. Possui como atributo a propriedade de dados `descricao`. É domínio das propriedades `geradaPorParticipante`, para indicar que a versão foi produzida por um participante responsável e `temEstadoVersao`, indicando que as versões possuem estados. O estado da versão é definido pela entidade `EstadoVersaoValuePartition`.

`Workspace:` espaço de trabalho que contém os artefatos, as ferramentas e as informações que os usuários utilizam para realizar o seu trabalho. É domínio das propriedades `acessadoPorUsuario`, indicando que um *workspace* é acessado por um usuário, `conectadoEmLocal`, indicando que um *workspace* está ativo quando está conectado a um local, `temArtefato`, para indicar que um *workspace* pode possuir cópias dos artefatos, funcionando como um repositório local, para que os usuários possam modificar os artefatos, `temFerramenta`, indicando que um *workspace* pode possuir ferramentas instaladas e `temUsuario`, para indicar que um *workspace* pertence a um usuário, que está logado.

`ValuePartition:` corresponde à implementação de um padrão de projeto para ontologias, usado para representar um conjunto de valores restritos para uma propriedade (Rector, 2005). Por exemplo, para a propriedade `temEstadoTarefa`, cujo domínio é a entidade `Tarefa`, os valores que podem ser assumidos são: aguardando recurso, em andamento, cancelado, suspenso ou concluído, sendo que a propriedade só pode assumir um desses valores por vez. Dessa forma, a entidade `Tarefa` tem a propriedade `temEstadoTarefa`, cuja imagem é `EstadoTarefaValuePartition`, particionada nos valores `EstadoTarefaAguardandoRecurso`, `EstadoTarefaCancelado`, `EstadoTarefaEmAndamento`, `EstadoTarefaSuspenso` e `EstadoTarefaConcluido`. As demais subentidades de `ValuePartition` definidas para a OntoDiSEN são apresentadas na Tabela 6.1.

Depois de criadas as entidades e propriedades de dados e objetos, cada entidade foi formalmente definida, utilizando a lógica descritiva para inserir restrições. Como exemplo,

[7] conflitos em versões de arquivos do ambiente DiSEN são tratados em (da Silva, 2008)

[8] os critérios para medir o conhecimento e as habilidades dos indivíduos não fazem parte do escopo deste trabalho, sendo definidos em (Lima, 2004)

Tabela 6.1: Subentidades de `ValuePartition`.

Entidades	Descrição
`EstadoArquivoValuePartition`	`EstadoArquivoIndisponivel` – o arquivo ainda não foi criado; `EstadoArquivoIntermediario` – o arquivo foi criado, mas não está concluído; `EstadoArquivoFinal` – o arquivo está pronto e nenhuma alteração é necessária.
`EstadoArtefatoValuePartition`	`EstadoArtefatoIndisponivel` – todos os arquivos do artefato estão em estado indisponível; `EstadoArtefatoIntermediario` – existe pelo menos um arquivo correspondente ao artefato em estado intermediário; `EstadoArtefatoFinal` – todos os arquivos do artefato estão em estado final.
`EstadoAtividadeProjetoValuePartition`	`EstadoAtividadeProjetoPlanejamento` – não há tarefas de uma atividade iniciadas; `EstadoAtividadeProjetoEmAndamento` – pelo menos uma tarefa está em andamento; `EstadoAtividadeProjetoConcluido` – todas as tarefas de uma atividade foram concluídas; `EstadoAtividadeProjetoSuspenso` – a atividade foi paralizada, podendo ser retomada, caso o problema que causou a suspensão seja resolvido. Durante o período de suspensão, as tarefas da atividade suspensa também são interrompidas; `EstadoAtividadeProjetoCancelado` – a atividade foi cancelada e não poderá ser retomada. Quando uma atividade é cancelada, as tarefas correspondentes também são interrompidas.
`EstadoFaseProjetoValuePartition`	`EstadoFaseProjetoPlanejamento` – nenhuma das atividades de uma fase foram iniciadas; `EstadoFaseProjetoEmAndamento` – pelo menos uma atividade de uma fase está em andamento; `EstadoFaseProjetoSuspenso` – a fase foi paralizada, podendo ser retomada, caso o problema que causou a suspensão seja resolvido. Durante o período de suspensão, as atividades da fase suspensa também são interrompidas; `EstadoFaseProjetoCancelado` – a fase foi cancelada e não poderá ser retomada. Quando uma fase é cancelada, as atividades correspondentes também são interrompidas; `EstadoFaseProjetoConcluido` – todas as atividades de uma determinada fase foram concluídas.
`EstadoProjetoValuePartition`	`EstadoProjetoPlanejamento` – nenhuma das fases correspondentes ao projeto foram iniciadas; `EstadoProjetoEmAndamento` – pelo menos uma fase do projeto está em andamento; `EstadoProjetoSuspenso` – o projeto foi paralizado, podendo ser retomado, caso o problema que causou a suspensão seja resolvido. Durante o período de suspensão, todas as fases do projeto suspenso são interrompidas; `EstadoProjetoCancelado` – o projeto foi cancelado e não poderá ser retomado. Quando um projeto é cancelado, todas as fases correspondentes são interrompidas; `EstadoProjetoConcluido` – todas as fases de um determinado projeto foram concluídas.
`EstadoRecursoValuePartition`	`EstadoRecursoDisponivel` – o recurso não estão em utilização no momento, podendo ser alocado para ou utilizado por algum projeto ou usuário; `EstadoRecursoIndisponivel` – o recurso está alocado ou em utilização por uma entidade.
`EstadoVersaoValuePartition`	`EstadoVersaoBloqueada` – versão de arquivo em processo de avaliação. As versões bloqueadas não podem ser utilizadas para compor o artefato, entretanto, podem se tornar versões finais ou obsoletas, quando os motivos que causaram o bloqueio forem analisados; `EstadoVersaoConflito` – versão de arquivo que possui alguma inconsistência[7]. Versões em conflito não podem ser utilizadas para compor o artefato, entretanto, podem se tornar versões finais ou obsoletas, quando os motivos que causaram o conflito forem resolvidos; `EstadoVersaoInutilizada` – versão inválida, não podendo ser utilizada para compor um artefato. Uma versão inutilizada não pode ser retomada; `EstadoVersaoObsoleta` – versões de arquivo válidas anteriores à final; `EstadoVersaoFinal` – última versão de arquivo válida.
`NivelConhecimentoValuePartition`[8]	`NivelConhecimentoBasico` – indivíduo com conhecimentos gerais, capacidade de resolver problemas simples; `NivelConhecimentoIntermediario` – indivíduo com conhecimentos mais específicos, capacidade de resolver problemas um pouco mais complexos; `NivelConhecimentoAvancado` – indivíduo com conhecimento detalhado, capacidade de resolver problemas complexos.
`NivelHabilidadeValuePartition`	`NivelHabilidadeBasico` – indivíduo com pouca habilidade, entretanto, com capacidade de desenvolver atividades simples; `NivelHabilidadeIntermediario` – indivíduo com capacidade de desenvolver atividades um pouco mais detalhadas; `NivelHabilidadeAvancado` – indivíduo com capacidade de desenvolver atividades complexas que exijam a habilidade determinada.
`NivelTreinamentoValuePartition`	`NivelTreinamentoBasico` – treinamento realizado pelos usuários que oferece a seus participantes conhecimentos básicos sobre o tema; `NivelTreinamentoIntermediario` – treinamento realizado pelos usuários que oferece a seus participantes conhecimentos intermediários sobre o tema; `NivelTreinamentoAvancado` – treinamento realizado pelos usuários que oferece a seus participantes conhecimentos avançados sobre o tema;

definiu-se que acessar um *workspace* e possuir um login e uma senha são condições necessárias e suficientes para uma entidade ser considerada um usuário. Além disso, definiu-se que um usuário é um recurso que administra um local, está em uma localidade, usa uma ferramenta e que possui apenas essas características. Essa definição é apresentada, em lógica descritiva, na forma

$$Usuario \equiv =1\ acessaWorkspace.Workspace \sqcap =1\ login.String \sqcap =1\ senha.String$$

$$Usuario \sqsubseteq Recurso \sqcap \exists administraLocal.Local \sqcap =1$$
$$estaEmLocalidade.Localidade \sqcap \exists usaFerramenta.Ferramenta \sqcap$$
$$\forall acessaWorkspace.Workspace \sqcap \forall administraLocal.Local \sqcap$$
$$\exists estaEmLocalidade.Localidade \sqcap \forall usaFerramenta.Ferramenta$$

Para essas restrições, utilizando a ferramenta Protegé, gerou-se o OWL, como mostrado na Figura 6.5. Uma especificação OWL, contendo a definição das principais entidades da ontologia, pode ser encontrado no Apêndice C.

Para verificar a consistência das entidades, propriedades e restrições geradas, a OntoDiSEN foi submetida ao raciocinador Fact++, integrado à interface da ferramenta Protégé. Para apontar inconsistências na ontologia, o Fact++ cria, na hierarquia de entidades, uma entidade padrão chamada *Nothing*. Todas as entidades inconsistentes são removidas da hierarquia principal (*Thing*) e inseridas como filhas da entidade *Nothing*. As propriedades são marcadas em vermelho. A Figura 6.6 mostra que o raciocinador Fact++ não encontrou inconsistências na ontologia apresentada neste capítulo, já que a entidade *Nothing* não possui qualquer entidade filha.

6.4 Considerações Finais de Capítulo

Este capítulo apresentou a ontologia OntoDiSEN, definida para representar as informações contextuais existentes em um ambiente de desenvolvimento distribuído de software, mais especificamente, o DiSEN.

Sobre esta etapa do trabalho, é importante salientar que as entidades definidas na ontologia envolvem as informações acerca do domínio como um todo, identificadas na fase de aquisição de conhecimento. Comparando com as entidades contextuais apresentadas no Capítulo 5, nota-se que a ontologia possui algumas entidades além daquelas consideradas na modelagem contextual. Isso ocorre porque a etapa de modelagem concentrou-se na definição dos elementos contextuais em nível de software (ambiente DiSEN), enquanto a representação estendeu seu escopo às informações sobre o domínio como um todo. Dessa

```
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Usuario -->
<owl:Class rdf:about="#Usuario">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#senha"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#acessaWorkspace"/>
            <owl:onClass rdf:resource="#Workspace"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#login"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Recurso"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usuarioEstaEmLocalidade"/>
            <owl:allValuesFrom rdf:resource="#Localidade"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#administraLocal"/>
            <owl:allValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usaFerramenta"/>
            <owl:allValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usaFerramenta"/>
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usuarioEstaEmLocalidade"/>
            <owl:onClass rdf:resource="#Localidade"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:allValuesFrom rdf:resource="#EstadoRecursoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#acessaWorkspace"/>
            <owl:allValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#administraLocal"/>
            <owl:someValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
```

Figura 6.5: Excerto da OntoDiSEN: Restrições da entidade Usuario em OWL.

forma, todas as informações identificadas pela modelagem contextual podem ser mapeadas para a representação, mas nem todas as informações que podem ser representadas estão presentes na modelagem contextual. O objetivo de estender a representação para o domínio como um todo é preparar a estrutura do modelo DiSEN-CSE para que possa incorporar informações contextuais sem provocar grandes impactos na representação.

Ainda assim, como o foco do modelo DiSEN-CSE está nas informações contextuais de usuários, ferramentas e locais, informações específicas sobre os projetos como gerenciamento de qualidade, riscos, manutenção de software, entre outras, não foram consideradas. Estender o modelo DiSEN-CSE para concentrar-se nas informações referentes aos projetos será realizado como trabalhos futuros. Além disso, os componentes de software que

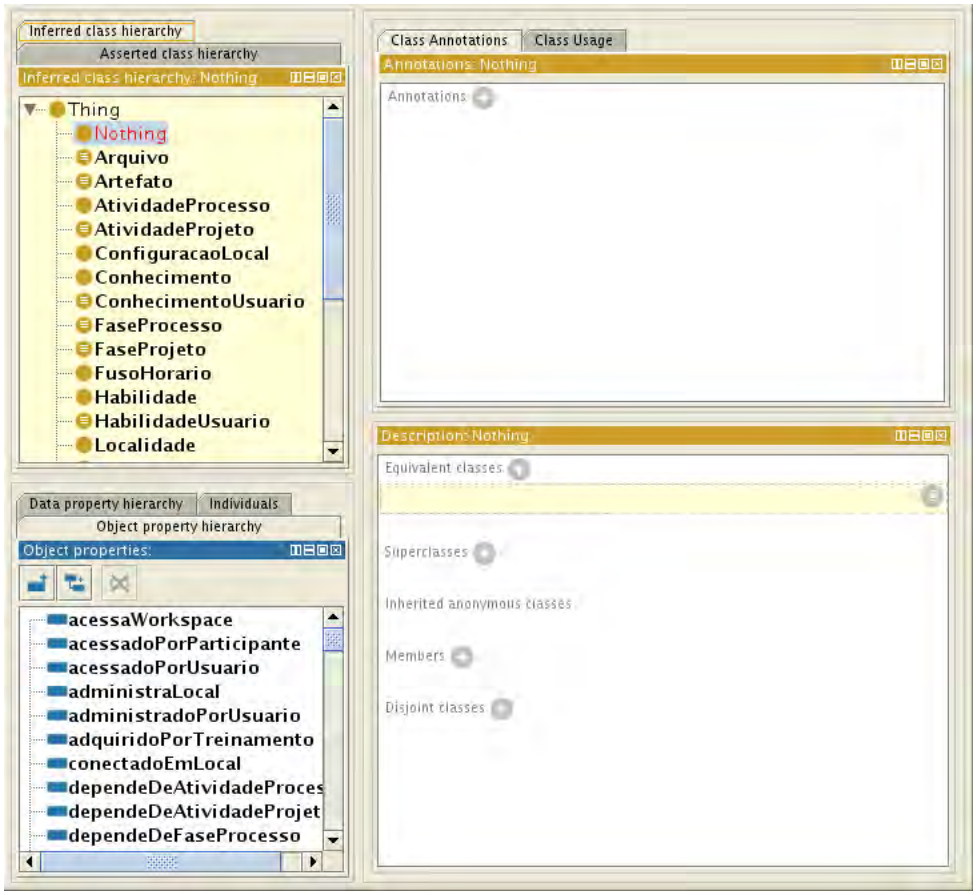

Figura 6.6: OntoDiSEN: Análise de consistência pelo raciocinador Fact++.

mapeiam as informações para o modelo de representação não foram implementados. O objetivo desta parte do trabalho foi definir quais os conceitos fazem parte do domínio e criar o modelo de representação, com a definição das principais entidades do domínio. A implementação completa desses componentes de software será realizada em trabalhos futuros, com a implementação dos elementos `Representação do contexto` e `Suporte ao processamento` do modelo DiSEN-CSE.

Com a modelagem contextual do ambiente DiSEN e a representação ontológica, é possível reconhecer as informações que compõe o contexto para cada foco e facilitar a compreensão semântica dos indivíduos a cerca do domínio tratado, respectivamente. Contudo, o modelo DiSEN-CSE tem como uma das principais características a disseminação dessas informações para os indivíduos interessados. Dessa forma, é importante que exista uma maneira de, automaticamente, enviar essas informações contextuais para os diversos *workspaces* compartilhados. O Capítulo 7 descreve o DiSEN-*Notifier*, o gerenciador de notificações responsável por essa tarefa.

Capítulo

7

Disseminação de informações

O modelo DiSEN-CSE tem como um dos principais objetivos disseminar informações de contexto para os indivíduos geograficamente dispersos. Disseminar consiste em, automaticamente, oferecer informações sobre as ações que ocorrem no ambiente e que influenciam a realização do trabalho cooperativo. Isso significa que é sua responsabilidade possibilitar que os usuários mantenham-se informados sobre as ações que ocorrem no ambiente e influenciam, de alguma maneira, suas próprias ações. Mais que isso, o DiSEN-CSE propõe que esse compartilhamento de informações ocorra de forma automática, sem intervenção dos usuários. Para isso, o modelo precisa de uma estrutura capaz de receber as informações contextuais provenientes das diversas fontes de contexto reconhecidas pelo `Suporte à captura`, saber como essas informações influenciam o trabalho de cada indivíduo e notificá-los, por mensagens, sobre os eventos ocorridos.

Os modelos contextuais apresentados no Capítulo 5 foram desenvolvidos para apoiar o processo de notificação dos indivíduos. Com esses modelos, é possível saber quais são as informações contextuais relevantes para cada foco (pelos modelos estruturais), de que forma essas informações influenciam o contexto (pelos modelos comportamentais) e em que momento os usuários precisam ser notificados sobre mudanças nessas informações (ações de notificação, definidas nos modelos comportamentais).

Para realizar as ações de notificação, é necessário que o modelo DiSEN-CSE ofereça um componente de software que, de posse das informações contextuais, seja capaz de disseminá-las para os usuários interessados. Esse capítulo apresenta o gerenciador de notificações DiSEN-*Notifier*, que compartilha as informações capturadas pelo modelo

DiSEN-CSE. Sua tarefa é receber as informações contextuais e compartilhá-las na forma de notificações de eventos. Além disso, é responsável por decidir quais indivíduos estão interessados nas informações, com base na modelagem comportamental apresentada no Capítulo 5.

A Seção 7.1 apresenta a estrutura do gerenciador de notificações. Em seguida, a Seção 7.1.2 mostra um exemplo de disseminação de informações realizado, com o objetivo de comprovar o funcionamento do modelo.

7.1 Notificações de eventos

Conforme discutido no Capítulo 3, quando se trata de trabalhos cooperativos, a notificação de eventos é a opção mais aplicada para compartilhamento de informações. De acordo com Neto (2006), não há uma maneira padrão de adquirir e manipular informações de contexto e o gerenciamento de eventos é normalmente explorado, via consulta, ou via notificação. A consulta requer esforço dos indivíduos, que precisam solicitar os eventos ocorridos a algum mecanismo de busca/armazenamento. A notificação, por outro lado, permite o compartilhamento automático, sendo responsabilidade do sistema comunicar a ocorrência de um novo evento. Apesar de aumentar a complexidade do sistema, o gerenciamento de eventos por notificação retira do usuário a responsabilidade de buscar pelos eventos ocorridos.

Quando se trata de eventos em ambientes distribuídos, existe uma especificação padrão para serviços de notificação de eventos, chamada *EventService Specification*, proposta pela OMG (*Object Management Group*). Esse serviço é parte integrante do *middleware* CORBA, responsável por permitir interoperabilidade entre aplicações em ambientes heterogêneos e distribuídos.

A idéia do *EventService* é permitir que os componentes distribuídos registrem, ou cancelem o registro, dinamicamente, os seus interesses por eventos específicos. O sistema poderá, então, emitir-lhes uma notificação quando o evento estiver disponível.

Pela especificação do *EventService*, existem dois modelos de notificação (OMG, 2004): *Push Model* e *Pull Model*. No *Push Model*, a entidade que produziu o evento comunica a ocorrência aos consumidores. No *Pull Model*, os consumidores solicitam os dados sobre o evento, realizando uma consulta à entidade que o produz. Essas operações são realizadas por interfaces pré-estabelecidas entre os fornecedores e consumidores do evento.

O DiSEN-*Notifier* baseia-se na idéia do *EventService*. Entretanto, não é responsabilidade das entidades realizar o registro/desregistro de seus interesses. Ao contrário, o gerenciador de notificações decide quem deve receber os dados sobre o evento, com

base no contexto das ações. Os modelos comportamentais, apresentados no Capítulo 5, auxiliam nessa tarefa, à medida que determinam, para cada contexto específico, que comportamento o ambiente precisa realizar e quais indivíduos estão envolvidos.

A Seção 7.1.1 descreve o gerenciador de notificações desenvolvido para o ambiente DiSEN.

7.1.1 DiSEN-Notifier

O DiSEN-*Notifier* foi desenvolvido para compor a infraestrutura do ambiente de desenvolvimento distribuído DiSEN. Essa infraestrutura é especificada pelo framework FRADE (Schiavoni, 2007).

O FRADE é dividido em quatro camadas: aplicação, negócios, infraestrutura e comunicação. A camada de aplicação contém os componentes gráficos para desenvolvimento de aplicativos, que podem ser adicionados dinamicamente no ambiente. A camada de negócios contém os gerenciadores que agrupam as entidades existentes no ambiente. Segundo a definição do FRADE, um gerenciador possui um conjunto de classes que o representa e a definição dos seus atributos e funcionalidades, sendo implementados na forma de componentes de software. A camada de infraestrutura oferece aos gerenciadores da camada de negócios, um conjunto de funcionalidades, divididas em serviços e suportes, sendo que os suportes são utilizados como clientes dos serviços. Os suportes são capazes de ativar os serviços locais e remotos. Por fim, a camada de comunicação é o meio pelo qual os suportes ativam os recursos remotos.

O DiSEN-*Notifier* é um gerenciador da camada de negócios. A Figura 7.1 mostra todos os gerenciadores presentes nessa camada, incluindo o gerenciador de notificações, e a relação com as funcionalidades da camada de infraestrutura. O gerenciador de notificações se comunica com duas funcionalidades: persistência e comunicação.

A estrutura dos objetos de negócio do DiSEN-*Notifier* é apresentada na Figura 7.2. O gerenciador de notificações depende de todos os demais gerenciadores, já que quaisquer entidades de negócio do ambiente DiSEN podem ajudar a compor a informação de contexto que o evento deve disseminar.

No pacote Gerenciador de Notificações, existe a classe `Evento`, que possui os atributos `quem`, indicando quem é a entidade que gerou o evento, `quando`, indicando o momento da ocorrência do evento e `tempoVida`, para indicar o momento em que o evento expira, não devendo mais ser enviado. As classes `EventoLocal`, `EventoUsuario`, `EventoFerramenta`, `EventoProjeto`, `EventoTarefa` e `EventoArtefato` são subclasses de `Evento`. Cada uma dessas classes correspondem às características específicas do evento para uma determinada

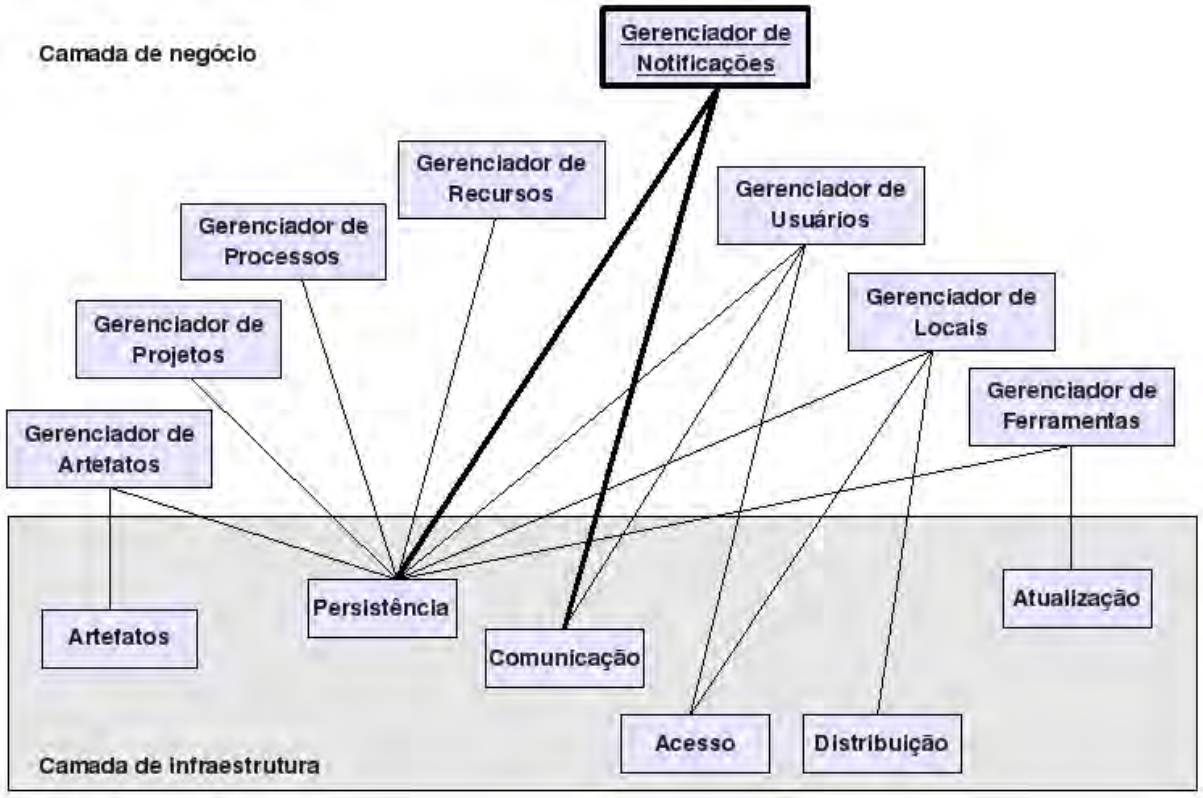

Figura 7.1: Camadas de negócio e infraestrutura do framework FRADE – relação do gerenciador de notificações com os demais componentes do ambiente DiSEN.

entidade. Por exemplo, a classe `EventoUsuario` corresponde às informações de evento específicas da entidade `Usuario`. Possui, além dos atributos herdados da classe evento, os atributos `onde`, indicando em qual *workspace* o usuário que gerou o evento está conectado e *oque*, indicando o tipo de evento ocorrido, representado pela classe `TipoEventoUsuario`. Essa classe corresponde a uma enumeração (classe do tipo `Enumeration`) e define os diversos tipos de eventos (representando mudanças contextuais) que podem ocorrer para a entidade `Usuario`. Como pode ser observado na Figura 7.2, cada subevento possui um conjunto de tipos de evento. Dessa forma, de acordo com o tipo do evento, o gerenciador de notificações interpreta a ocorrência para decidir quem deve receber a notificação sobre eventos daquele tipo, com base nos modelos comportamentais.

Conforme definido no Capítulo 4, o modelo DiSEN-CSE precisa armazenar em um repositório de dados as mudanças contextuais das entidades persistentes. Além disso, todos os eventos gerados serão armazenados para consultas futuras. Por essa razão, o gerenciador de notificações utiliza a infraestrutura de persistência. A persistência é responsável por encapsular as operações sobre bases de dados. Assim, a localização e a forma de acesso a essas bases são transparentes para os gerenciadores que as utilizam.

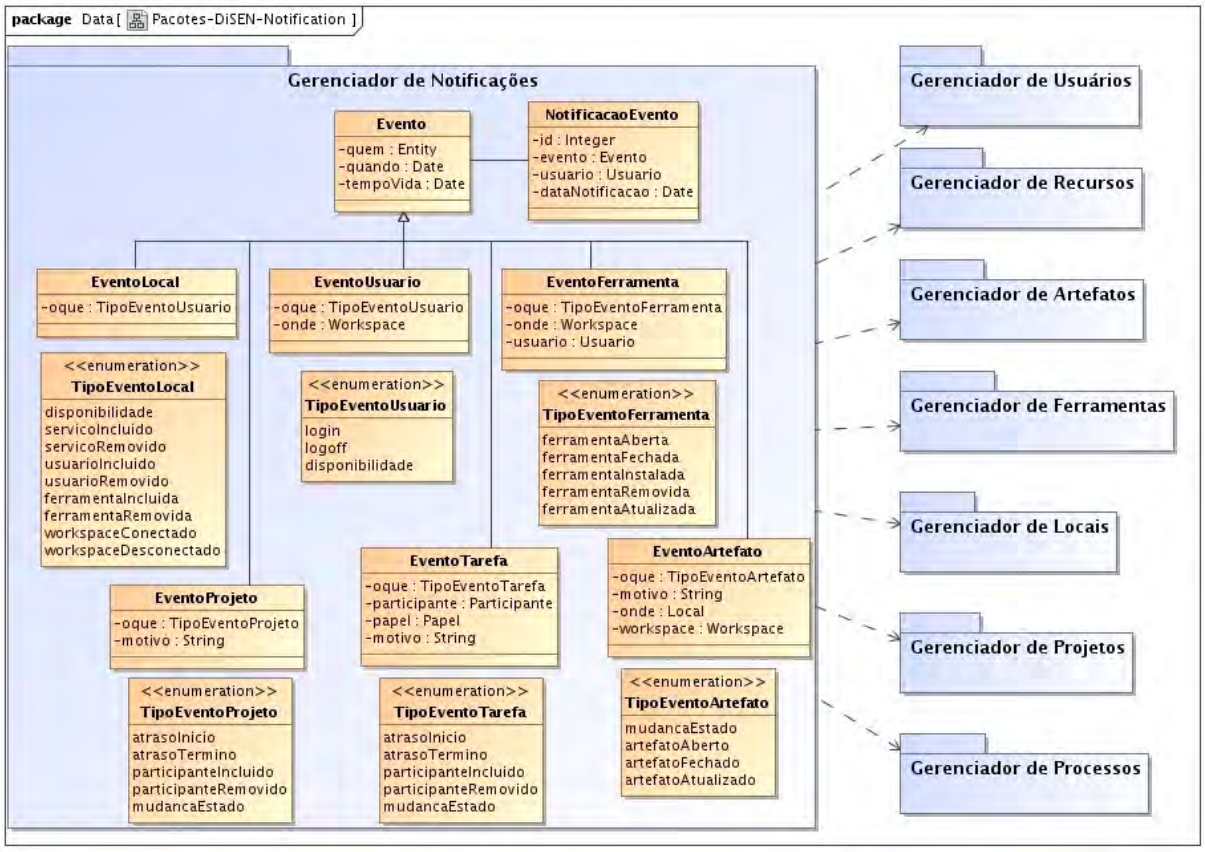

Figura 7.2: Diagrama de Pacotes – DiSEN-*Notifier*.

Quando deseja armazenar/consultar uma informação ou evento, o DiSEN-*Notifier* solicita essa ação ao suporte a persistência.

Além da persistência, o gerenciador de notificações se comunica com a infraestrutura de comunicação. A comunicação é a responsável por encapsular o canal de comunicação, por onde são realizadas as transmissões das informações entre suporte (cliente) e serviços (servidor). Essa funcionalidade possui operações como: localizar usuários conectados, atualizar a lista de usuários conectados, enviar mensagem aos usuários conectados, receber mensagens, entre outros. Quando uma mudança no contexto gera um evento, o gerenciador de notificações verifica quem deve receber a notificação e, então, solicita ao suporte à comunicação que localize os usuários online e envie a notificação na forma de mensagem. Se algum usuário interessado na informação não estiver online, o gerenciador armazena o evento até que o usuário esteja disponível. Quando o usuário se conecta, todos os eventos armazenados para ele são enviados pela comunicação.

A Figura 7.3 mostra a relação do DiSEN-*Notifier* com os suportes da camada de infraestrutura. No gerenciador de notificações, o modelo de negócios de projeto repre-

sentam as classes de negócio descritas na Figura 7.2. As regras de negócio de projeto correspondem às implementações dos modelos comportamentais (descritos no Capítulo 5), que auxiliam a decisão sobre quem deve receber as informações sobre os eventos. Os suportes a persistência e a comunicação representam os clientes da infraestrutura que são acessíveis a partir do gerenciador de notificações.

Figura 7.3: Acesso do DiSEN-*Notifier* aos suportes da camada de infraestrutura.

O acesso à estrutura interna do gerenciador de notificações ocorre a partir da interface apresentada na Figura 7.4.

Figura 7.4: Interface de acesso ao DiSEN-*Notifier*.

7.1.2 Exemplo de uso

O elemento `Mecanismos de Apresentação` do modelo DiSEN-CSE tem como objetivo disponibilizar diversos métodos de apresentação, de modo que cada usuário do ambiente possa receber as notificações de eventos de acordo com suas preferências, perfil e com a relevância da informação para o seu trabalho. Os métodos de apresentação aplicados ao ambiente DiSEN foram definidos em (Pozza, 2005). Entretanto, não basta implementar diversos métodos. É necessário escolher o método mais adequado para cada indivíduo, o que depende de diversos fatores. A implementação do elemento *Mecanismos de Apresentação* será realizada em pesquisas futuras.

Entretanto, para comprovar o funcionamento do DiSEN-*Notifier*, bem como da estrutura básica do modelo DiSEN-CSE, uma interface de notificação foi desenvolvida para receber os elementos capturados e exibi-los aos usuários interessados, de maneira pouco

intrusiva. O objetivo da interface é ilustrar a disseminação de informações contextuais. Como o modelo DiSEN-CSE não foi implementado totalmente neste trabalho, os agentes de software e os sensores capazes de realizar a captura de informações contextuais ainda não estão disponíveis. Dessa forma, os modelos comportamentais de contexto apresentados são, essencialmente, baseados em consultas à informações persistentes (armazenadas em um banco de dados). Além disso, para utilizar o modelo ontológico de representação, seriam necessários componentes de software que acessem e manipulem os dados da ontologia. Por essas razões, embora o exemplo de notificação apresentado seja bastante simples, ajuda a comprovar a existência de uma infraestrutura capaz de enviar notificações com base na implementação de modelos comportamentais de contexto, disponibilizando informações de acordo com o contexto dos indivíduos em cada foco.

Na interface de exemplo, quando um *workspace* do ambiente DiSEN é instanciado, um ícone representando o gerenciador de notificações é exibido na barra de tarefas do sistema operacional. Na Figura 7.5, é possível visualizar um símbolo de alerta, correspondendo ao DiSEN-*Notifier*, no canto direito da barra de tarefas, na interface gráfica Gnome do sistema operacional Linux.

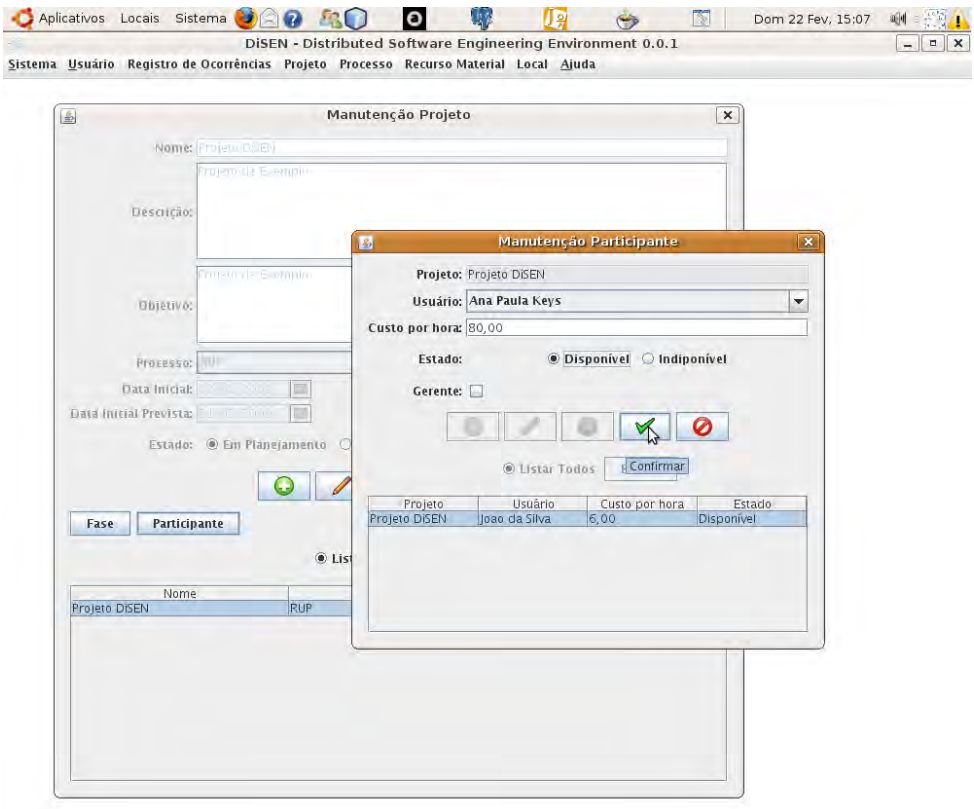

Figura 7.5: Inserindo usuária Ana Paula Keys como participante de um projeto.

Quando ocorre um evento, o gerenciador de notificações decide quem deve receber a informação e envia uma mensagem. As mensagens ficam armazenadas e cada usuário pode consultá-las, clicando no ícone do gerenciador. Ao ser acionado, o DiSEN-*Notifier* exibe uma interface, que manterá a lista contendo os últimos eventos notificados àquele usuário e as suas principais informações.

Suponha um cenário em que o gerente de projetos defina quem são os usuários que devem participar de um determinado projeto (Foco: `Definir Participantes` (Chaves, 2009)). Nesse foco, cujo modelo comportamental é apresentado na Figura 7.6, depois de ter analisado os conhecimentos e habilidades de cada usuário, o gerente de projetos seleciona aqueles que farão parte do projeto em questão, de acordo com a disponibilidade dos usuários. Então, definidos quem serão os participantes, o ambiente notifica-os sobre sua alocação para o projeto.

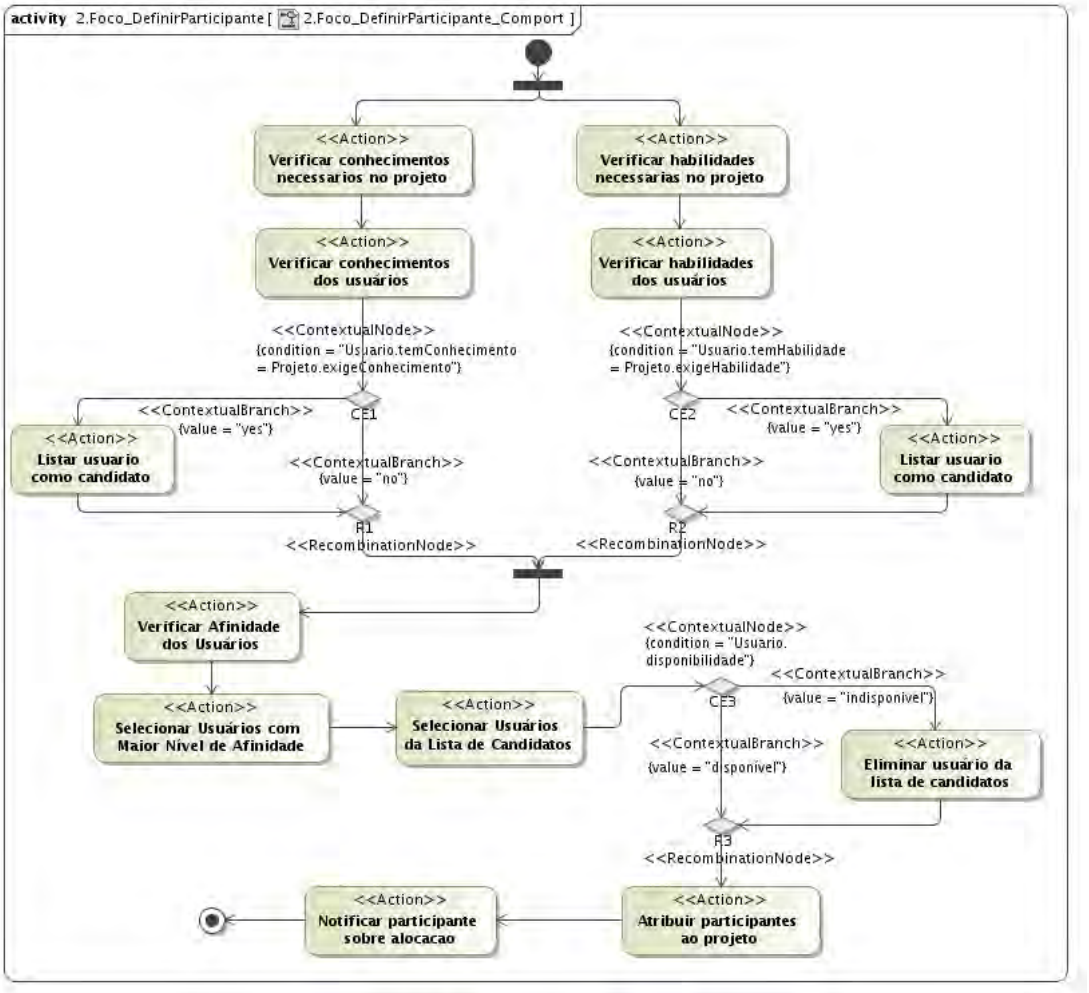

Figura 7.6: Modelo Comportamental – Foco: Definir Participante.

A Figura 7.5 mostra a interface de inclusão de participantes em um projeto. No momento em que o gerente de projetos insere um novo participante e seleciona a opção `Confirmar`, o ambiente reconhece quem é o usuário incluído e envia uma mensagem. Essa notificação corresponde à ação de notificação `Notificar participantes sobre alocação` do modelo comportamental da Figura 7.6. Se o participante estiver online, a mensagem é enviada imediatamente. Caso contrário, a mensagem é enviada no instante em que o usuário se conectar no ambiente. A Figura 7.7 mostra a mensagem recebida por um usuário incluído no projeto. Esse evento é do tipo `participanteIncluido` e corresponde a eventos de Projetos (classe `EventoProjeto`).

Figura 7.7: DiSEN-*Notifier* comunicando ao usuário sobre inclusão no projeto.

A mensagem da Figura 7.7 permanece por alguns segundos e, então, é fechada. Caso deseje ver a notificação, ao clicar no ícone do gerenciador de notificações, é exibida a interface da Figura 7.8, contendo o evento ocorrido, bem como os demais eventos anteriores.

Figura 7.8: Evento comunicando ao participante inserido no projeto.

Com base nesse exemplo, é possível destacar alguns pontos. Em primeiro lugar, a decisão sobre quando enviar uma mensagem e quem deve recebê-la baseia-se nas ações de notificação dos modelos comportamentais de contexto. Isso mostra a importância desses modelos para a disseminação de informações. O DiSEN-*Notifier* torna-se sensível ao contexto à medida que implementa os modelos comportamentais de contexto e consegue,

com base nesses modelos, saber quais indivíduos estão interessados naquele foco e que, portanto, devem ser comunicados sobre as ações ocorridas. Em segundo lugar, como os modelos de contexto definem o contexto sob o qual as notificações devem ser enviadas, quando aquelas circunstâncias pré-definidas ocorrem, o sistema dispara as notificações automaticamente, sem que os usuários realizem qualquer ação para solicitar o envio ou para consultar o recebimento de novas informações. Essa característica agiliza o processo de tomada de decisão e facilita a cooperação, por manter os indivíduos sempre informados, evita que os usuários deixem de comunicar os demais sobre as mudanças ocorridas, seja por esquecimento ou falta de pré-disposição em estabelecer comunicação com os indivíduos.

7.2 Considerações Finais

Devido às dificuldades de comunicação provocadas pela dispersão geográfica, é necessário que os indivíduos possuam uma forma de conhecer as informações referentes ao contexto que envolve a execução das ações, para facilitar a compreensão dos indivíduos sobre o trabalho cooperativo. O modelo DiSEN-CSE define um conjunto de informações contextuais relevantes para o ambiente DiSEN e que ações precisam ser realizadas com base em cada variação de contexto. Além disso, propõe que essas informações sejam automaticamente disseminadas para os indivíduos interessados, aumentando seu conhecimento sobre as ações que acontecem no ambiente e diminuindo a sensação de distância entre os membros das equipes dispersas.

Este capítulo apresentou o gerenciador de notificações DiSEN-*Notifier*, capaz de encontrar os indivíduos interessados em um determinado evento ocorrido e compartilhar com eles informações sobre esses eventos. O DiSEN-*Notifier* implementa os modelos comportamentais apresentados no Capítulo 5 para decidir sobre os usuários interessados em cada evento. Entretanto, para este trabalho, apenas alguns focos foram implementados, com o objetivo de verificar a eficácia do gerenciador para compartilhamento de informações. Os demais modelos comportamentais serão implementados em trabalhos futuros.

Capítulo

8

Conclusões

O objetivo deste trabalho foi apresentar um modelo baseado em *context-awareness* para disseminação de informações no ambiente de desenvolvimento distribuído de software DiSEN. O modelo DiSEN-CSE foi desenvolvido na tentativa de reduzir os impactos causados pela dificuldade de comunicação e pelas incompreensões que ocorrem em projetos com equipes geograficamente distantes.

Com base nas diversas pesquisas encontradas na literatura (destacadas no Capítulo 3), identificou-se que os indivíduos geograficamente dispersos possuem mais dificuldade em trocar informações sobre os trabalhos realizados, mesmo que exista uma infraestrutura de comunicação eficiente. Além disso, quando ocorre a comunicação, informações incompletas e ambiguidades dificultam a realização dos projetos.

Para tentar minimizar esses problemas, esse trabalho propõe a disseminação automática de informações contextuais para os indivíduos participantes do trabalho cooperativo. A disseminação automática é importante à medida que não exige esforço por parte dos indivíduos no compartilhamento das informações sobre as ações que ocorrem no ambiente. O contexto é importante à medida que permite conhecer as condições sob as quais os objetos de cooperação são produzidos, diminuindo a dificuldade cognitiva dos indivíduos.

Assim, o modelo DiSEN-CSE propõe uma divisão em quatro elementos: Suporte à captura, Representação do contexto, Suporte ao processamento e Mecanismos de apresentação. Juntos, esses elementos são capazes de capturar as informações contextuais existentes no ambiente DiSEN, representá-las de maneira única, processá-las, na tentativa

de inferir novas informações a partir destas e disseminá-las para os diversos *workspaces* compartilhados.

Para concretizar esse modelo, este trabalho concentra-se em oferecer a estrutura básica para seu funcionamento, fundamentada em três pilares:

- identificar quais informações são capazes de compor o contexto das ações e como essas informações influenciam o comportamento do ambiente DiSEN;
- desenvolver um modelo de representação que ofereça semântica às informações do ambiente, a fim de diminuir as ambiguidades e inconsistências; e
- desenvolver um gerenciador de notificações, capaz de reconhecer os usuários interessados nas informações, com base no contexto das ações, e comunicá-los, na forma de notificações, sobre os eventos que ocorrem no ambiente.

Para identificar as informações que influenciam o contexto, este trabalho utiliza o metamodelo *Context Metamodel*, proposto por (Vieira, 2008). A modelagem contextual é dividida em modelos estruturais e comportamentais. Os modelos estruturais representam os elementos contextuais, as entidades que os possuem, as relações existentes entre elas, bem como a forma de aquisição de cada elemento contextual. Os modelos comportamentais definem, para cada foco, que ações o ambiente deve realizar a cada variação de comportamento influenciada pelo contexto.

Para o modelo de representação, foi desenvolvida a OntoDiSEN, uma ontologia que representa o conhecimento de um ambiente de desenvolvimento distribuído de software, baseado no ambiente DiSEN. A utilização de um modelo ontológico se justifica pelo formalismo e capacidade de inferência dessa técnica de representação.

Quanto ao gerenciador de notificações, o DiSEN-*Notifier* foi desenvolvido de acordo com os critérios definidos por (Schiavoni, 2007), no *framework* FRADE, para ser integrado à infraestrutura do ambiente DiSEN. O gerenciador de notificações é capaz de receber um evento gerado pelo ambiente e capturado pelo `Suporte à captura`, reconhecer quem são os usuários interessados e disseminar a informação a esses usuários.

Além da construção da estrutura básica do modelo DiSEN-CSE, entre as contribuições desse trabalho, destacam-se:

- as informações contextuais do ambiente DiSEN estão definidas utilizando um metamodelo, facilitando a compreensão e reutilização dos modelos gerados;
- como os elementos contextuais são definidos para cada foco, a modelagem contextual pode ser extendida para englobar outras funcionalidades do ambiente DiSEN, exigindo o esforço mínimo para a realização dessa tarefa;

- os modelos comportamentais facilitam a representação da dinamicidade do contexto e, com base nesses modelos, novas regras de negócio podem ser criadas para a notificação de eventos do DiSEN-CSE;

- a representação das informações utilizando um modelo ontológico diminui as ambiguidades das informações compartilhadas e facilita o raciocínio e inferência sobre as informações existentes;

- o compartilhamento automático das informações contextuais permite que os indivíduos sejam informados de mudanças contextuais ocorridas no ambiente, sem ter que realizar uma consulta ou estabelecer comunicação com outro indivíduo.

Vale salientar que, por se tratar de um modelo bastante amplo, o DiSEN-CSE não foi implementado completamente neste trabalho. Apenas a estrutura básica foi construída, com o objetivo de verificar a viabilidade do modelo. A modelagem contextual foi realizada para os focos apresentados no diagrama de casos de uso da Figura 5.1, porém, este é um conjunto não exaustivo de focos, não considerando, por exemplo, ações gerenciais, como planejamento de projetos e gerenciamento de riscos.

Variações de comportamento que não possuem um instante de tempo determinado para ocorrer (que não podem ser representados em uma linha do tempo) não foram mapeadas, por não ser apoiadas pelo *Context Metamodel*. Os grafos contextuais, utilizados para a modelagem comportamental, possuem uma característica temporal, com ações executando uma após a outra, em uma linha de tempo. O ambiente DiSEN possui comportamentos esporádicos, cujo momento de ocorrência não pode ser determinado no tempo. Um exemplo desse tipo de comportamento são atrasos em projetos. Um projeto está atrasado quando a data prevista é atingida e o projeto não foi iniciado. Não existe uma ação do ambiente que produza essa informação. Ela ocorre esporadicamente no tempo, sem influência de qualquer outro elemento. Por essa razão, não é possível representá-lo em um grafo contextual, sendo necessário encontrar uma outra forma de representação, que permita modelar esse tipo de informação.

No que diz respeito à ontologia OntoDiSEN, foi definido um conjunto fundamental e não exaustivo de entidades, podendo ser incorporadas novos conceitos, a medida que o modelo DiSEN-CSE englobar novas funcionalidades. Entretanto, é necessário que se desenvolva, ainda, um componente de software para mapear as informações do ambiente DiSEN para o modelo ontológico. Quanto ao gerenciador de notificações DiSEN-*Notifier*, as regras de negócio, que correspondem à implementação dos modelos comportamentais gerados na modelagem contextual, não foram completamente implementados. Foram

construídos apenas uma parte dos modelos, como forma de avaliar o funcionamento do gerenciador. Os demais focos modelados, que podem ser encontrados em (Chaves, 2009), serão implementados futuramente. Além disso, todas as mensagens são enviadas aos usuários em uma interface gráfica padrão, sem considerar, até esse momento, a relevância da informação para o foco e o perfil do usuário que receberá a mensagem.

Além das questões discutidas acima, como limitações dessa pesquisa, podemos destacar:

- por ser a primeira iniciativa de utilizar técnicas de contexto para o ambiente DiSEN, as ações cujo contexto foram mapeadas correspondem apenas às ações de software. O contexto pessoal dos indivíduos não foi modelado;
- como os elementos de Suporte à Captura não foram desenvolvidos nesse trabalho, as fontes de contexto limitaram-se à informações armazenadas (*queried*) e fornecidas pelo usuário (*UserDefined*). Portanto, não foram modelados cenários que utilizavam informações obtidas por sensores ou pelo perfil e preferências dos usuários;
- os componentes de software para acessar e utilizar o modelo de representação ontológico não foram desenvolvidos. Dessa forma, não foi possível utilizar o modelo de representação para avaliação do DiSEN-CSE;
- não foram realizados estudos experimentais para verificar se o modelo DiSEN-CSE realmente diminui as ambiguidades e facilita a comunicação, já que os elementos que compõe o modelo não foram completamente implementados;
- o exemplo de utilização do gerenciador de notificações objetiva mostrar apenas que o ambiente é capaz de compartilhar as informações com os usuários, com base nos modelos comportamentais de contexto. Não representa, portanto, um caso de instanciação completa do modelo DiSEN-CSE.

Com base nas limitações destacadas, os próximos passos dessa pesquisa consistem em:

- estender a modelagem contextual para incluir as ações gerenciais e contextos pessoais dos indivíduos;
- definir a relevância dos elementos contextuais, de acordo com o foco, o perfil e as preferências dos usuários;
- definir e implementar agentes de software para realizar a captura das informações contextuais, atuando como fontes de contexto;

- implementar componentes de software para acessar e manipular as informações contextuais mapeadas para o modelo ontológico;
- definir maneiras de processar as informações contextuais existentes a fim de deduzir novas informações e extrair conhecimentos daquelas já existentes;
- definir e implementar diferentes mecanismos de apresentação, para possibilitar o envio de notificações de acordo com a relevância das informações para cada indivíduo, seu perfil e suas preferências;
- implementar os demais modelos comportamentais gerados para o ambiente DiSEN;
- desenvolver uma representação para modelar comportamentos cujo momento de ocorrência não pode ser determinado em uma linha do tempo;
- realizar um estudo experimental para avaliar a capacidade do modelo DiSEN-CSE de minimizar os efeitos da distância geográfica sobre a compreensão dos indivíduos acerca do objeto de cooperação.

Por fim, é importante salientar que, ainda que não hajam experimentos que comprovem a eficácia do modelo DiSEN-CSE para reduzir as ambiguidades e aumentar a compreensão dos indivíduos, o desenvolvimento das partes apresentadas neste trabalho e a avaliação empírica realizada sobre os exemplos testados, indicam que o modelo é capaz de atender ao objetivo proposto, por disseminar, com base no contexto, as informações sobre ações que ocorrem no ambiente DiSEN para indivíduos geograficamente dispersos, sem exigir esforços extras dos usuários para essa comunicação.

Referências Bibliográficas

Araujo, R. M.; Brézillon, P. Reinforcing shared context to improve collaboration. *Revue d'Intelligence Artificielle*, v. 19, n. 3, p. 537–556, 2005.

Baldauf, M.; Dustdar, S.; Rosenberg, F. A survey on context-aware systems. In: *International Journal of Ad Hoc and Ubiquitous Computing*, Inderscience, 2007, p. 263–277.

Bauer, J. *Identification and modeling of contexts for different information scenarios in air traffic*. Dissertação de Mestrado, Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik, Institut für Computergestützte Informationssysteme, Berlin, 2003.

Bazire, M., B. P. Understanding context before using it. In: *5th International and Interdisciplinary Conference, CONTEXT 2005*, Paris, France: Springer Verlag, 2005, p. 29–40.

Belotti, R. *Sophie - context modelling and control*. Relatório Técnico, Swiss Federal Institute of Technology Zurich, 2004.

Biegel, G.; Cahill, V. A framework for developing mobile, context-aware applications. In: *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication*, 2004, p. 361–365.

Bos, N.; Shami, N. S.; Olson, J. S.; Cheshin, A.; Nan, N. In-group/out-group effects in distributed teams: an experimental simulation. In: *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, New York, NY, USA: ACM, 2004, p. 429–436.

Brooks, K. The context quintet: Narrative elements applied to context awareness. In: *Human Computer Interaction*, Grécia, 2003.

Brézillon, P. Representation of procedures and practices in contextual graphs. In: *The Knowledge Engineering Review*, Cambridge University Press, 2003, p. 147–174.

Brézillon, P. Task-realization models in contextual graphs. In: *CONTEXT*, Springer, 2005, p. 55–68 (*Lecture Notes in Computer Science*, v.3554).

Brézillon, P. Cxg community – contextual graph example. [*On-line*], http://www.cxg.fr/.

Brézillon, P.; Pomerol, J. C. Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain*, v. 62, n. 3, p. 223–246, 1999.

Bublitz, F. M. *LOTUS: Um middleware baseado em plug-ins para o desenvolvimento de aplicações em ambientes pervasivos com suporte à ciência de contexto.* Relatório Técnico, Universidade Federal de Campina Grande, 2006.

Chaves, A. P. *Modelagem conceitual das informações de contexto para o ambiente disen.* Relatório Técnico, Universidade Estadual de Maringá, Maringá – Brasil, 2009. Disponível em `http://www.din.uem.br/disen`

Chen, H. L. *An intelligent broker architecture for pervasive context-aware systems.* Tese de Doutoramento, Department of Computer Science and Electrical Engineering – Faculty of the Graduate School of the University of Maryland, USA, 2004.

Dey, A. K. *Providing architectural support for building context-aware applications.* Tese de Doutoramento, Doctor of Philosophy in Computer Science – Georgia Institute of Technology, Georgia, EUA, 2000.

Dey, A. K.; Abowd, G. D. *Towards a better understanding of context and context-awareness.* Relatório Técnico GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, Atlanta GA USA, 1999.

Dey, A. K.; Abowd, G. D.; Salber, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction Journal*, p. 97–166, special Issue on Context-Aware Computing, 2001.

Dourish, P.; Belloti, V. Awareness and coordination in shared workspaces. In: *ACM Conference on Computer-Supported Cooperative Work (CSCW'92)*, ACM Press, 1992, p. 107–114.

ECKHARD, B. *Context-aware notification in global software development.* Dissertação de Mestrado, Institut für Softwaretechnik und interaktive Systeme – Technischen Universität Wien, 2007.

FALBO, R. A. *Integração de conhecimento em um ambiente de desenvolvimento de software.* Tese de Doutoramento, Coppe - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1998.

FERNÁNDEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. Methontology: from ontological art towards ontological engineering. In: *Proceedings of the AAAI97 Spring Symposium*, Stanford, USA, 1997, p. 33–40.

FILHO, J. V.; SACRAMENTO, V.; ROCHA, R. C. A.; ENDLER, M. MoCA: Uma arquitetura para o desenvolvimento de aplicações sensíveis ao contexto para dispositivos móveis. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Curitiba, 2006.

FILIPPO, D.; FUKS, H.; DE LUCENA, C. J. P. Notificação-ação: Informação e acesso ao ambiente de aprendizagem através de notificações para suporte à coordenação de fóruns de cursos a distância. In: *Simpósio Brasileiro de Sistemas Colaborativos*, IEEE Computer Society, 2008, p. 111–121.

FUKS, H.; RAPOSO, A.; GEROSA, M. Do modelo de colaboração 3c à engenharia de groupware. In: *Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia*, 2003.

GRUBER, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, v. 5, p. 199–220, 1993.

GU, T.; PUNG, H. K.; ZHANG, D. Q. A service-oriented middleware for building context-aware services. In: *Journal of Network and Computer Applications*, Elsevier, 2005, p. 1–18.

GU, T.; WANG, X. H.; PUNG, H. K.; ZHANG, D. Q. An ontology-based context model in intelligent environments. In: *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, California, USA, 2004.

GUARINO, N. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In: *SCIE*, 1997, p. 139–170.

GÓMEZ, J. I. V. *A reactive behavioural model for context-aware semantic devices.* Tese de Doutoramento, Programa de Doctorado en Ciencia de la Computación e Inteligencia Artificial – Universidad de Deusto, Bilbao, España, 2007.

HELD, A.; BUCHHOLZ, S.; SCHILL, A. Modeling of context information for pervasive computing applications. In: *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002)*, Orlando, FL, USA, 2002.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Modeling context information in pervasive computing systems. In: *1st International Conference on Pervasive Computing*, Zurich, Switzerland: Springer, 2002, p. 167–180.

HERBSLEB, J.; MOCKUS, A.; FINHOLT, T.; GRINTER, R. An empirical study of global software development: distance and speed. *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, p. 81–90, 2001.

HERBSLEB, J.; MOITRA, D. Guest editor's introduction: Global software development. *IEEE Software*, v. 18, n. 2, p. 16–20, 2001.

HERBSLEB, J. D. Global software engineering: The future of socio-technical coordination. In: *Future of Software Engineering (FOSE'07)*, IEEE Computer Society, 2007.

HOFER, T.; SCHWINGER, W.; PICHLER, M.; LEONHARTSBERGER, G.; ALTMANN, J. Context-awareness on mobile devices – the hydrogen approach. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2002, p. 292–302.

HUZITA, E. H. M.; TAIT, T. F. C.; COLANZI, T. E.; QUINAIA, M. A. Um ambiente de desenvolvimento distribuído de software – disen. In: *I Workshop de Desenvolvimento Distribuído de Software*, João Pessoa - PB, 2007, p. 31–38.

IVCEK, M.; GALINAC, T. Aspects of quality assurance in global software development organization. In: *31st International Convention MIPRO 2008*, Opatija, 2008.

KEVIN C. DESOUZA, YUKIKA AWAZU, P. B. Managing knowledge in global software development efforts: Issues and practices. *IEEE Software*, p. 30–37, 2006.

KOBYLINSKI, R.; CREIGHTON, O.; DUTOIT, A. H.; BRUEGGE, B. Building awareness in global software engineering: using issues as context. In: *Workshop on Global Software Development, part of the International Conference on Software Engineering (ICSE)*, 2002.

KORPIPÄÄ, P.; MÄNTYJÄRVI, J.; KELA, J.; KERÄNEN, H.; MALM, E.-J. Managing context information in mobile devices. In: *IEEE Pervasive Computing*, VTT Technical Research Centre of Finland, IEEE CS and IEEE ComSoc, 2003, p. 42–51.

LIMA, F. *Mecanismo de apoio ao gerenciamento de recursos humanos no contexto de um ambiente distribuído de software.* Dissertação de Mestrado, Departamento de Informática, Universidade Estadual de Maringá, 2004.

MARTIMIANO, L. A. F. *Sobre a estruturação de informação em sistemas de segurança computacional: o uso de ontologia.* Tese de Doutoramento, ICMC – USP, São Paulo, 2006.

MOLLI, P.; SKAF-MOLLI, H.; OSTER, G.; JOURDAIN, S. Sams: Synchronous, asynchronous, multi-synchronous environments. In: *Proceedings of 7th International Conference on CSCW in Design (CSCWD'02)*, Rio de Janeiro, Brasil, 2002, p. 80–84.

NETO, R. F. B. *Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto.* Tese de Doutoramento, ICMC – USP, São Paulo, 2006.

NOY, F. N.; MCGUINESS, D. L. *Ontology development 101: a guide to creating your first ontology.* Relatório Técnico KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

NUNES, V. T.; SANTORO, F. M.; BORGES, M. R. S. Um modelo para gestão de conhecimento baseado em contexto. In: *XXVII Congresso da Sociedade Brasileira de Computação*, NCE–IM Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ: SBSC – Simpósio Brasileiro de Sistemas Colaborativos, 2007, p. 1841–1854.

OMG *Eventservice specification.* Relatório Técnico formal/04-10-02 v.1.2, Object Management Group, Inc., 2004.

PASCUTTI, M. C. D. *Uma proposta de arquitetura de um ambiente de desenvolvimento de software distribuído baseado em agentes.* Dissertação de Mestrado, Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

PINHEIRO, M. K.; DE LIMA, J. V.; BORGES, M. R. S. Awareness em sistemas de groupware. In: *International Database Engineering and Applications Symposium*, San Diego, Costa Rica: Centre de Información Tecnológica (CIT), 2001, p. 323–335.

Pozza, R. S. *Proposta de um modelo para cooperação baseado no gerenciador de workspace no ambiente disen.* Dissertação de Mestrado, Departamento de Informática, Universidade Estadual de Maringá, 2005.

Prikladnicki, R.; Audy, J. Munddos: Um modelo de referência para desenvolvimento distribuído de software. In: *XVIII Simpósio Brasileiro de Engenharia de Software*, 2004.

Ranganathan, A.; Campbell, R. H. A middleware for context-aware agents in ubiquitous computing environments. In: *ACM/IFIP/USENIX International Middleware Conference*, Rio de Janeiro, Brasil, 2003.

Rector, A. Representing specified values in owl: "value partitions" and "value sets". W3C Working Group. Último acesso dia: 31/04/09, 2005.
Disponível em `http://www.w3.org/TR/swbp-specified-values/`

Roberto, R. L. *Uma abordagem para identificação de interesses dos usuários durante a navegação em websites semânticos.* Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação. Universidade Estadual de Maringá, 2006.

Rosa, M. G. P. *Inserindo contexto em groupware.* Dissertação de Mestrado, Instituto de Matemática e Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, 2004.

Sangwan, R.; Bass, M.; Mullick, N.; Paulish, D. J.; Kazmeier, J. *Global software development handbook (auerbach series on applied software engineering series).* Boston, MA, USA: Auerbach Publications, 2006.

Schiavoni, F. L. *FRADE – framework para infra-estrutura de um ambiente de desenvolvimento distribuído de software.* Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação. Universidade Estadual de Maringá, 2007.

da Silva, C. A. *Disenscv: gerenciador de versões de artefatos para um ambiente de desenvolvimento distribuído de software.* Dissertação de Mestrado, Departamento de Informática, Universidade Estadual de Maringá, 2008.

Simperl, E. P. B.; Tempich, C. Ontology engineering: A reality check. In: *OTM Conferences (1)*, Springer, 2006, p. 836–854 (*Lecture Notes in Computer Science*, v.4275).

STRANG, T.; LINNHOFF-POPIEN, C. A context modeling survey. In: *Workshop on Advanced Context Modelling, Reasoning and Management, in 6th International Conference on Ubiquitous Computing*, Inglaterra, 2004.

THERESA EDGINGTON, BEOMJIN CHOI, K. H. T. R.; VINZE, A. Adopting ontology to facilitate knowledge sharing. *Communications of the ACM*, v. 47, n. 2, p. 85–90, 2004.

VIEIRA, V. *Gerenciamento de contexto em sistemas colaborativos.* Tese de Doutoramento, CIn – Universidade Federal de Pernambuco, Recife – Pernambuco, monografia de Qualificação, 2006.

VIEIRA, V. *CEManTIKA: A domain-independent framework for designing context-sensitive systems.* Tese de Doutoramento, CIn – Universidade Federal de Pernambuco, Recife – Pernambuco, 2008.

WANG, W.; CAO, Y.; QIN, Z. A novel context-awareness model. In: *WCICA 2006, 6th World Congress on Intelligent Control and Automation*, Dalian, China: IEEE publisher, 2006, p. 1906–1909.

Apêndice

A

Documento de Visão

A.1 Introdução

Este documento coleta, analisa e define as necessidades e características mais importantes de um modelo baseado em *context-awareness* para auxiliar a disseminação de informações contextuais em um ambiente de desenvolvimento distribuído de software. O objetivo desse modelo é permitir que os membros das equipes geograficamente dispersas estejam cientes do contexto dos outros indivíduos e que essas informações possuam uma representação única, aumentando a clareza e diminuindo a ambiguidade.

As principais características serão representadas nos diagramas de casos de uso e nas especificações suplementares.

A.1.1 Objetivo

O objetivo deste documento é identificar os *stakeholders* do modelo, seus interesses, os usuários designados a cada papel e por que eles são necessários.

A.1.2 Escopo

Este documento está associado à especificação de um modelo para disseminação de informações contextuais para apoiar o desenvolvimento distribuído de software. Apresenta o modelo em uma macro-visão, não definindo seus requisitos específicos.

A.1.3 Organização do documento

Este documento está organizado da seguinte forma: a Seção A.2 traz a definição detalhada dos *stakeholders* e dos usuários do modelo e a Seção A.3 apresenta um resumo de todas capacidades do modelo: define suas características a ser implementadas, as interfaces externas e descreve as prioridades entre as diferentes características.

A.2 Descrição dos Stakeholders e Usuários

Esta seção apresenta o perfil dos *stakeholders* e usuários envolvidos, além dos problemas chave identificados para serem resolvidos pela solução proposta, justificando a necessidade da existência dos mesmos.

A.2.1 Resumo dos stakeholders (não usuários).

A Tabela A.2.1 descreve os indivíduos interessados no projeto, entretanto que não corresponde a usuários do modelo.

Nome	**Descrição**	**Papel**
Orientadora	Professora que participa da especificação do modelo, agindo como cliente.	Responsável por conduzir o trabalho, auxiliar na definição dos elementos, oferecer informações sobre o domínio e participar da avaliação.
Aluna	Desenvolvedora do projeto.	Responsável por gerar a especificação dos elementos do modelo.

A.2.2 Perfil dos stakeholders (não usuários).

As Tabelas A.1 e A.2 descrevem os perfis dos *stakeholders* não usuários Orientadora e Aluna.

Tabela A.1: Perfil do *stakeholder* Orientadora.

Representante	Elisa Hatsue Moriya Huzita
Descrição	Professora que participa do projeto como cliente.
Tipo	Usuário especialista
Responsabilidades	Conduzir o trabalho, auxiliar a definição dos elementos, oferecer informações sobre o domínio e participar da avaliação.

Tabela A.2: Perfil do *stakeholder* Aluna.

Representante	Ana Paula Chaves
Descrição	Desenvolvedora do modelo.
Tipo	Desenvolvedor
Responsabilidades	Gerar todos os artefatos necessários ao detalhamento, especificação e implementação da infraestrutura do modelo.

A.2.3 Necessidades chave do ambiente sobre o modelo (S/U – Stakeholders Usuários)

A Tabela A.2.3 apresenta as necessidades chave que o modelo tenta solucionar.

Necessidade	S/U envolvidos	Solução proposta
Capturar as informações contextuais	Agentes	Especificar maneiras para que agente humanos ou de software consigam identificar quais são as informações de contexto relevantes, perceber a ocorrência de mudanças nesse contexto e capturar essas mudanças por algum sensor físico, lógico ou virtual.
Representação das informações contextuais	Gerenciador de notificações	As informações contextuais são capturadas pelos agentes em forma de eventos. Esses eventos são enviados para o gerenciador de notificações, da camada de negócios do ambiente DiSEN, que deverá representá-las utilizando uma ontologia pré-definida, para que as informações seja compreendidas de maneira única por todos os participantes da interação. Quando necessário, as informações representadas podem ser armazenadas para consultas futuras.
Processamento das informações contextuais	Gerenciador de notificações	As informações contextuais transientes (que não são armazenadas) ou as informações persistentes que foram solicitadas por algum mecanismo de recuperação, são submetidas a um processamento. Esse processamento se torna possível, à medida que as informações estarão representadas em uma ontologia que permitirá inferências sobre as informações, deduzindo informações novas, a partir das já existentes.
Disseminação das informações contextuais	Gerenciador de notificações	Depois de processadas, as informações estão prontas para serem disseminadas pelos mecanismos de percepção. Para isso, o serviço de notificação deve controlar quem está interessado naquela informação e então enviá-la para os mecanismos correspondentes.

A.3 Resumo das capacidades

Esta seção apresenta uma macro visão das funcionalidades do modelo.

A.3.1 Responsabilidades

Agentes de captura: monitoram o contexto das entidades e capturam as informações contextuais, enviando para o serviço de notificação.

Gerenciador de notificações: recebe as informações enviadas pelos agentes de captura, delega a um módulo de representação para a ontologia, armazena quando necessário; quando não necessário, submete a informação a um motor de inferência, verifica quem está interessado na informação e envia notificação.

A.3.2 Interfaces externas

O modelo será integrado ao ambiente DiSEN, de modo que os agentes de captura podem estar em um cliente ou em um servidor, podendo se comunicar com o gerenciador de notificações, que faz parte da camada de negócio do ambiente.

A.3.3 Características do modelo

Todas as características definidas para este modelo poderá ser encontrada nos documentos de especificações gerados a partir dos softwares Netbeans IDE e Magic Draw, versões 15.0 e 16.0.

A.3.4 Precedência e prioridades

O modelo deverá ser especificado respeitando-se a seguinte ordem de prioridade, aqui apresentada em ordem decrescente:

Definir as informações contextuais: especificar quais são as informações que formam o contexto das entidades Locais, Usuários e Ferramentas, para saber que tipos de agentes de captura serão necessários e que tipo de informações precisam capturar.

Definir a ontologia de representação: construir a ontologia com base nas informações de contexto que precisam ser representadas para cada entidade.

Implementar o gerenciador de notificação de eventos: implementar um gerenciador da camada de negócios do DiSEN para receber as informações de contexto, verificar quem está interessado nessas informações e disparar as notificações para os mecanismos de percepção.

A parte do modelo que relaciona os elementos à representação ontológica desenvolvida e os elementos para processamento das informações contextuais e seleção dos métodos de percepção serão desenvolvidos em trabalhos futuros.

Apêndice

B

Especificação dos Casos de Uso

B.1 Introdução

Este documento apresenta as especificações dos casos de uso de um modelo baseado em contexto para disseminação de informações em um ambiente de desenvolvimento distribuído de software, o DiSEN-CSE. Os casos de uso foram utilizados para definir as principais funcionalidades do modelo e os responsáveis por cada uma delas.

B.1.1 Objetivo

O objetivo desse documento é detalhar o fluxo de eventos dos casos de uso do modelo, definindo o objetivo de cada caso de uso, os atores envolvidos e as condições (pré e pós) para a realização dos casos de uso.

B.1.2 Escopo

Este documento está associado à especificação de um modelo para disseminação de informações contextuais para apoiar o desenvolvimento distribuído de software. Apresenta o modelo em uma macro-visão, não definindo seus requisitos específicos.

B.1.3 Organização do documento

Este documento está organizado da seguinte maneira: a Seção B.2 apresenta os atores que interagem com o modelo. A Seção B.3 traz os diagramas de caso de uso e a Seção B.4 contém as tabelas de especificações de cada caso de uso da Seção B.3.

B.2 Especificação dos atores

Figura B.1: Atores do modelo DiSEN-CSE.

B.3 Diagramas de Casos de Uso

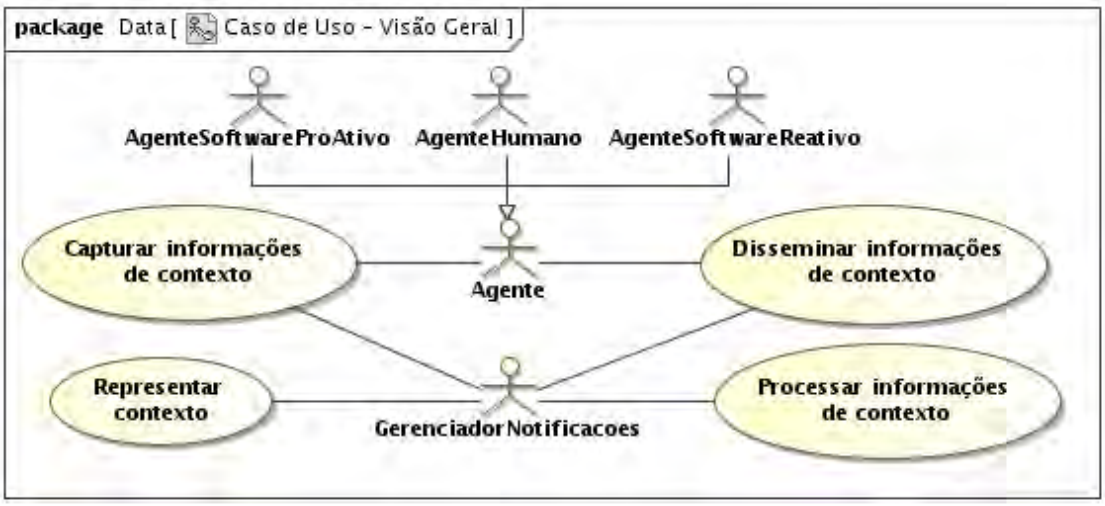

Figura B.2: Diagrama de Casos de Uso – Visão Geral.

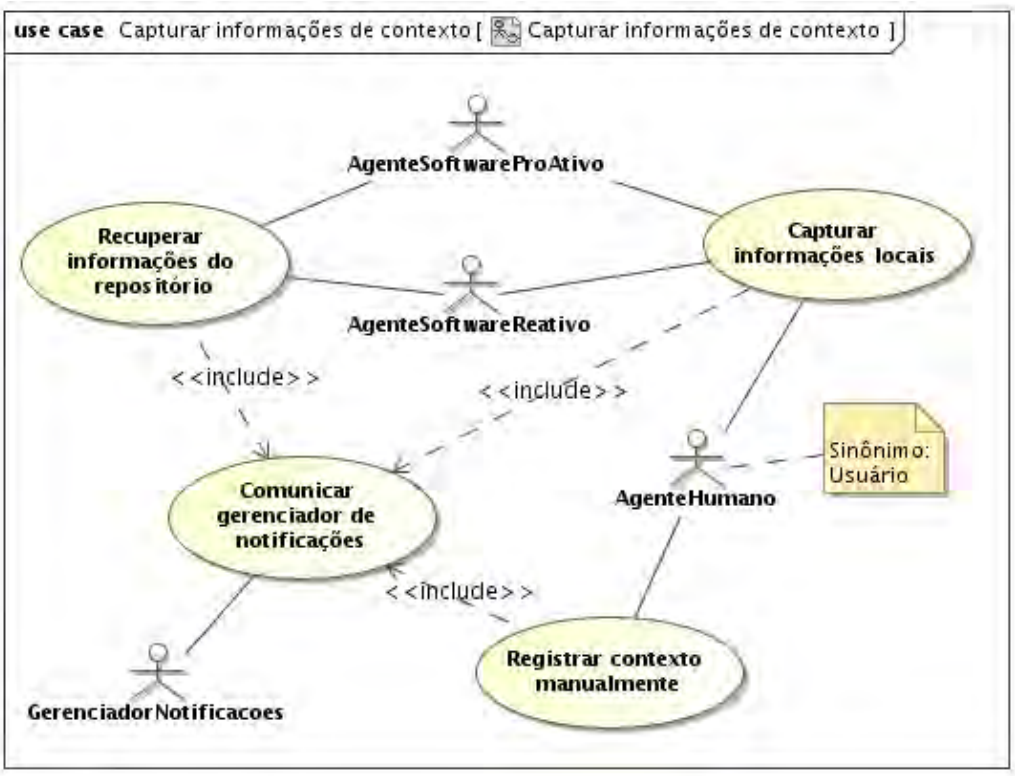

Figura B.3: Diagrama de Casos de Uso – Capturar informações de contexto.

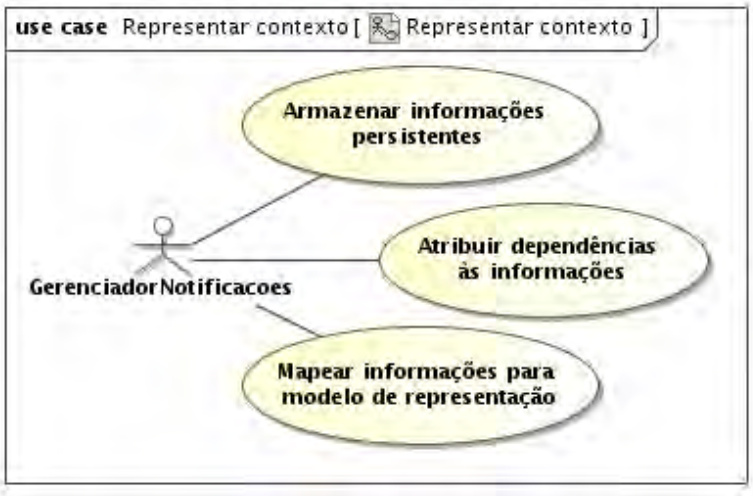

Figura B.4: Diagrama de Casos de Uso – Representar contexto.

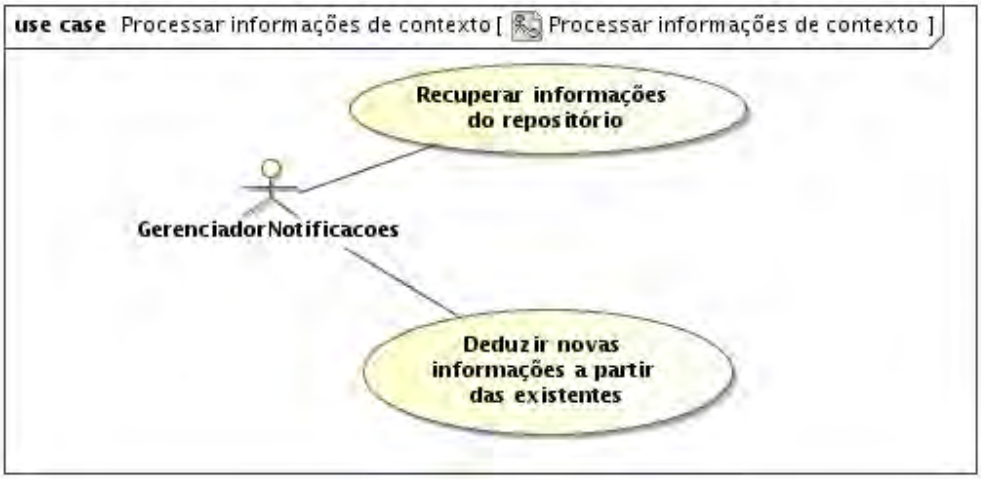

Figura B.5: Diagrama de Casos de Uso – Processar informações de contexto.

Figura B.6: Diagrama de Casos de Uso – Disseminar informações de contexto.

B.4 Especificação dos Casos de Uso

Esta seção apresenta, para cada caso de uso do modelo DiSEN-CSE, sua especificação e a dos cenários correspondentes, contendo descrição, fluxo de eventos, pré e pós-condições, entre outras informações relevantes.

Tabela B.1: Capturar informações de contexto.

Nome do Caso de Uso	Capturar informações de contexto
Autor	Ana Paula Chaves
Data	02/07/2008 11:03
Descrição	Este caso de uso consiste em capturar as informações de contexto, quer sejam informações sentidas por agentes de software, quer sejam informações providas por agentes humanos.

Tabela B.2: Capturar informações locais.

Nome do Caso de Uso	Capturar informações locais	
Caso de Uso Principal	Capturar informações de contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 11:14	
Atores	Agente (todas as especializações de agentes)	
Descrição	As informações contextuais locais podem ser obtidas de arquivos de arquivos de configuração, interfaces gráficas ou por interpretação feita a partir de outros sensores (por exemplo, um sensor físico). Nesse caso, a captura das informações é transparente ao usuário, mesmo que ele esteja fornecendo-as por uma interface gráfica (ação involuntária). Esse caso de uso inclui o caso de uso Comunicar o seviço de notificação.	
Pré-condições	Houveram mudanças no contexto de alguma entidade monitorada.	
Pós-condições	As informações capturadas estão prontas para serem representadas.	
Fluxo de Eventos	**Ações**	
	1	Os agentes monitoram os provedores de informações ou sensores (arquivos, interfaces, sensores físicos).
	2	Os agentes verificam uma alteração no contexto e captura.
	3	Os agentes informam ao seviço de notificação sobre a mudança no contexto.

Tabela B.3: Recuperar informações do repositório.

Nome do Caso de Uso	Recuperar informações do repositório	
Caso de Uso Principal	Capturar informações de contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 13:49	
Atores	Agente de software pró-ativo, Agente de software reativo	
Descrição	As informações de contexto podem ser persistentes ou transientes. Quando são persistentes, são armazenadas em um repositório e podem ser recuperadas quando necessário. Para isso, os agentes de software, seja pró-ativo (quando necessita de informações históricas) ou reativos (quando o usário solicita uma consulta a um determinado mecanismo de recuperação), obtém a solicitação de consulta e busca a informação no repositório de dados. Esse caso de uso inclui o caso de uso Comunicar o seviço de notificação.	
Pré-condições	Exista uma solicitação de consulta a informações de contexto persistentes.	
Pós-condições	As informações capturadas estão prontas para serem processadas.	
Fluxo de Eventos	**Ações**	
	1	O agente de software (pró-ativo/reativo) necessita de/recebe uma solicitação de consulta ao repositório.
	2	O agente de software submete a consulta ao repositório de informações contextuais e recebe a resposta.
	3	Os agentes informam ao seviço de notificação a informação de contexto recuperada.

Tabela B.4: Registrar contexto manualmente.

Nome do Caso de Uso	Registrar contexto manualmente	
Caso de Uso Principal	Capturar informações de contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 15:51	
Atores	Agente humano	
Descrição	Um agente humano informa seu contexto a partir de uma interface gráfica, por exemplo, por um formulário. Diferentemente do caso de uso capturar informações locais, o usuário está ciente de que sua ação fornecerá informações contextuais (ação voluntária). Esse caso de uso inclui o caso de uso Comunicar o seviço de notificação.	
Pré-condições	Houveram mudanças no contexto do usuário.	
Pós-condições	As informações fornecidas pelo usuário estão prontas para serem representadas.	
Fluxo de Eventos	**Ações**	
	1	O usuário registra seu contexto por uma interface gráfica do ambiente.
	2	As informações fornecidas pelo usuário são enviadas para o gerenciador de notificações.

Tabela B.5: Comunicar gerenciador de notificações.

Nome do Caso de Uso	Comunicar gerenciador de notificações	
Caso de Uso Principal	Capturar informações de contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 15:57	
Atores	Agente humano	
Dependências	Inclusão: Capturar informações locais, Recuperar informações no repositório, Registrar contexto manualmente	
Descrição	O gerenciador de notificações recebe as informações de contexto capturadas pelos agentes e decide o tratamento que será dado à informação, de acordo com as características (persistente/transiente/resultado de consulta).	
Pré-condições	Ocorrência de um dos casos de uso dependentes.	
Pós-condições	Informações prontas para a representação.	
Fluxo de Eventos	**Ações**	
	1	O agente envia uma informação de contexto ao gerenciador de notificações.
	2	O gerenciador de notificações verifica se a informação é persistente, transiente, ou se é resultado de uma consulta ao repositório. As informações persistentes ou transientes devem ser representadas. As informações provenientes de consultas já foram formalmente representadas.

Tabela B.6: Representar contexto

Nome do Caso de Uso	Representar contexto
Autor	Ana Paula Chaves
Data	02/07/2008 17:23
Descrição	Este caso de uso consiste em representar formalmente na ontologia as informações contextuais. As informações representadas podem possuir dependências com outras informações já existentes. As informações persistentes precisam ser armazenadas. As informações transientes, depois de representadas, estão prontas para serem processadas.

Tabela B.7: Mapear informações para modelo de representação.

Nome do Caso de Uso	Mapear informações para modelo de representação	
Caso de Uso Principal	Representar contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 17:47	
Atores	Gerenciador de notificações	
Descrição	O gerenciador de notificações, ao receber informações dos agentes de captura, deverá representá-las formalmente, com base na ontologia pré-definida. Este caso de uso é responsável por mapear essas informações para as classes da ontologia, respeitando as relações pré-estabelecidas.	
Pré-condições	Existem informações contextuais capturadas e não representadas.	
Pós-condições	Informações representadas na ontologia.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações determina a relação entre a informação e as classes da ontologia.
	2	O gerenciador de notificações mapeia as informações para a ontologia pré-definida.

Tabela B.8: Atribuir dependências às informações

Nome do Caso de Uso	Atribuir dependências às informações	
Caso de Uso Principal	Representar contexto	
Autor	Ana Paula Chaves	
Data	02/07/2008 18:09	
Atores	Gerenciador de notificações	
Descrição	Algumas classes de informações podem depender de outras informações relacionadas. Este caso de uso é responsável por mapear as dependências da informação atual, para facilitar o processamento, posteriormente.	
Pré-condições	Existem classes na ontologia que dependem da informação obtida.	
Pós-condições	Informação de contexto relacionada às suas dependências.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações verifica a existência de dependências com relação à informação mapeada.
	2	O gerenciador de notificações relaciona a informação às classes dependentes.

Tabela B.9: Armazenar informações persistentes

Nome do Caso de Uso	Armazenar informações persistentes	
Caso de Uso Principal	Representar contexto	
Autor	Ana Paula Chaves	
Data	03/07/2008 09:29	
Atores	Gerenciador de notificações	
Descrição	Quando as informações são persistentes, o gerenciador de notificações deve, após a representação, armazená-la para garantir que esta informação possa ser recuperada posteriormente.	
Pré-condições	Existem informações contextuais persistentes representadas.	
Pós-condições	Informações representadas, armazenadas.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações verifica a necessidade de persistir as informações.
	2	O gerenciador de notificações solicita ao serviço de persistência que armazene as informações em um repositório de informações contextuais.

Tabela B.10: Processar informações de contexto

Nome do Caso de Uso	Processar informações de contexto
Autor	Ana Paula Chaves
Data	03/07/2008 09:59
Descrição	Quando as informações de contexto são representadas, antes de serem disseminadas para os indivíduos interessados, precisam ser submetidas a um processamento, que permite a extração de novas informações a partir daquelas recebidas. Esse raciocínio é possível devido a utilização da ontologia, que possibilita inferência sobre as informações representadas.

Tabela B.11: Deduzir novas informações a partir das existentes

Nome do Caso de Uso	Deduzir novas informações a partir das existentes	
Caso de Uso Principal	Processar informações de contexto	
Autor	Ana Paula Chaves	
Data	03/07/2008 10:07	
Atores	Gerenciador de notificações	
Descrição	O gerenciador de notificações possui um suporte ao processamento que consiste em um motor de inferência capaz de deduzir informações a partir das já existentes. Esse caso de uso consiste em submeter as informações que serão disseminadas a esse motor, obtendo novas informações.	
Pré-condições	Existem informações representadas e não processadas, para ser disseminadas.	
Pós-condições	Novas informações deduzidas.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações submete as informações ao motor de inferência.
	2	O gerenciador de notificações recebe as informações obtidas a partir do raciocínio.

Tabela B.12: Disseminar informações de contexto

Nome do Caso de Uso	Disseminar informações de contexto
Autor	Ana Paula Chaves
Data	03/07/2008 10:25
Descrição	Este caso de uso consiste em disseminar as informações de contexto para todos os indivíduos interessados naquele contexto.

Tabela B.13: Identificar receptores da informação

Nome do Caso de Uso	Identificar receptores da informação	
Caso de Uso Principal	Disseminar informações de contexto	
Autor	Ana Paula Chaves	
Data	03/07/2008 10:26	
Atores	Gerenciador de notificações	
Descrição	O gerenciador de notificações mantém uma lista de receptores da informação contextual. Um receptor pode ser um local (todos os workspaces conectados a um determinado servidor), uma ferramenta (todos os usuários que utilizam determinada ferramenta) ou um usuário (um indivíduo ou um grupo). Quando o gerenciador recebe uma informação representada e processada, ele deve se encarregar de enviar essa informação a todas as entidades que a utilizam.	
Pré-condições	Existem informações de contexto representadas e processadas, prontas para serem disseminadas.	
Pós-condições	Listagem das entidades interessadas na informação contextual.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações verifica em sua lista de receptores, quem está interessado na informação.
	2	O gerenciador de notificações produz uma nova lista contendo apenas os receptores da informação a ser disseminada.

Tabela B.14: Identificar mecanismos de percepção

Nome do Caso de Uso	Identificar mecanismos de percepção	
Caso de Uso Principal	Disseminar informações de contexto	
Autor	Ana Paula Chaves	
Data	03/07/2008 10:33	
Atores	Gerenciador de notificações	
Descrição	As informações precisam ser disseminadas por mecanismos de percepção. Os mecanismos são selecionados de acordo com a relevância da informação para a entidade receptora. Para isso, o gerenciador de notificações mantém um resgistro de como as informações precisam ser exibidas (qual mecanismo utilizar para a disseminação). Este caso de uso seleciona os mecanismos de percepção que serão utilizados para cada entidade da lista de receptores.	
Pré-condições	Existem informações de contexto representadas e processadas, prontas para serem disseminadas.	
Pós-condições	Listagem dos mecanismos de percepção utilizados para cada entidade da lista de receptores.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações verifica quais mecanismos de percepção devem ser utilizados para cada entidade da lista de receptores.
	2	O gerenciador de notificações produz uma lista de mecanismos de percepção para os quais a informação deve ser enviada.

Tabela B.15: Compartilhar informações

Nome do Caso de Uso	Compartilhar informações	
Caso de Uso Principal	Disseminar informações de contexto	
Autor	Ana Paula Chaves	
Data	03/07/2008 11:20	
Atores	Gerenciador de notificações, Agentes	
Descrição	De posse da informação contextual, a lista dos receptores e a lista dos mecanismos de percepção, o gerenciador de notificações se encarrega de enviar as informações para os respectivos mecanismos, para serem exibidas para as entidades receptoras.	
Pré-condições	Informações para disseminação, lista de receptores, lista de mecanismos de percepção.	
Pós-condições	Informações disseminadas.	
Fluxo de Eventos	**Ações**	
	1	O gerenciador de notificações envia as informações contextuais para cada mecanismo de percepção de cada entidade da lista receptora.
	2	Os agente humanos (usuários ou grupos) ou de software (módulos do sistema, ferramentas, locais) recebem as informações pelos mecanismos de percepção e exibem.

Apêndice

C

Documentação da OntoDiSEN

C.1 Introdução

Este documento apresenta os artefatos gerados para a construção da ontologia OntoDiSEN, que corresponde ao modelo de representação de informações utilizados no DiSEN-CSE. O objetivo da ontologia é oferecer semântica aos conceitos do ambiente DiSEN, diminuindo as ambiguidades e facilitando a comunicação entre os indivíduos. Além disso, o processamento sobre as informações representadas pode, futuramente, oferecer conhecimentos sobre o domínio.

C.1.1 Objetivo

O objetivo desse documento é descrever os artefatos gerados no desenvolvimento da ontologia OntoDiSEN.

C.1.2 Escopo

Este documento está associado à especificação de um modelo para disseminação de informações contextuais para apoiar o desenvolvimento distribuído de software. Apresenta o modelo em uma macro-visão, não definindo seus requisitos específicos.

C.1.3 Organização do documento

Este documento está organizado da seguinte maneira: a Seção C.2 apresenta as questões de competência que nortearam o desenvolvimento da ontologia OntoDiSEN. A Seção C.3 apresenta a definição das principais classes do domínio. A Seção C.4 traz um excerto da ontologia, contendo as difinições das principais classes.

C.2 Questões de competência

C.2.1 Questões de competência relacionadas a Usuários

Relacionadas à localidade

- Em que local o usuário está logado?
- Em que local ocorreu o último login?

Relacionadas à identidade

- Quem é o usuário?
- Que papéis o usuário desempenha?

Relacionadas à atividade

- A que grupos o usuário pertence?
- Quais tarefas o usuário realiza?
- De quais projetos o usuário participa?
- Qual o estado do projeto do usuário?
- Quais artefatos o usuário produz?
- Quais artefatos o usuário acessa?
- De quais interações o usuário participa?
- Em qual tarefa está trabalhando no momento?
- Que mensagens o usuário enviou para outros?

- Que mensagens o usuário recebeu de outros?
- Quais ferramentas um usuário utiliza?
- Quais ferramentas um usuário tem permissão para acessar?
- Que ferramenta o usuário está utilizando?
- A tarefa do usuário possui restrições?
- Por que um usuário realizou uma ação?

Relacionadas à tempo e presença

- Em quais horários um usuário permanece logado?
- Quando iniciou/concluiu um artefato?
- Quando iniciou/concluiu uma tarefa?
- O usuário está logado?
- Quais usuários do mesmo projeto estão logados?
- Qual o estado do usuário (ocupado, ausente, presente)?
- Em quais elementos o usuário está interessado?

C.2.2 Questões de competência relacionadas a Ferramentas

Relacionadas à localidade

- Em quais locais está disponível?
- Em quais locais está instalada?

Relacionadas à identidade

- Qual é o tamanho?
- Qual é a plataforma?
- Qual é a versão?
- Quais as dependências com outras ferramentas?

- Quem pode acessar essa ferramenta?
- Como instalar a ferramenta (auto-exec, zip, instalador)?
- Que tipo de artefatos gera?

Relacionadas à atividade

- Quais usuários a utilizam? Qual projeto inclui a utilização dessa ferramenta? Quem está utilizando a ferramenta agora? Quais artefatos foram gerados nesta ferramenta? Por que instalar a ferramenta?

Relacionadas à tempo

- Qual a data da sua última atualização?

C.2.3 Questões de competência relacionadas a Locais

Relacionadas à localidade

- Qual é sua identificação (id/porta)?
- Quem está conectado?
- Quem é o administrador?
- Quais serviços oferece?
- Que ferramentas estão disponíveis?
- Qual a capacidade do hardware (processa/armazenamento)?

Relacionadas à tempo e presença

- Há quanto tempo o local está disponível?
- Qual dos locais permanece mais tempo disponível?
- O local está disponível?

C.3 Glossário de termos

1. Arquivo - um conjunto de dados produzidos em uma ferramenta e armazenados no computador.
2. Artefato – são todos os produtos resultantes da execução de uma atividade. É formado por uma composição de arquivos.
3. Atividade do processo – define, em uma macro visão, a subdivisão das fases do processo, indicando os objetivos específicos para sua realização. Quando instanciadas em um projeto, uma atividade pode ser subdivida em tarefas.
4. Atividade do projeto – agrupamento de tarefas de um projeto.
5. FaseProcesso – define o agrupamento das atividades de um processo.
6. FaseProjeto – agrupamento de atividades de um projeto, geralmente relativa a uma fase de processo específica.
7. Ferramenta – produto que software que possibilita a criação de arquivos.
8. Local – equipamento de hardware em que estão disponíveis os serviços, ferramentas, artefatos e workspaces.
9. Participante – usuário que participa de um projeto.
10. Processo – conjunto de ações pré-determinadas que devem ser seguidos para o desenvolvimento de um produto de software.
11. Projeto – Instância de um processo que determina os objetivos que devem ser atingidos com sua execução, estipulando prazos e metas a serem cumpridos.
12. Recurso – matéria-prima utilizada para a realização de um projeto.
13. Tarefa – unidade atômica de um projeto, corresponde aos objetivos mais específicos que pode ser realizado em um projeto.
14. Usuário – indivíduo que possui um login e uma senha para se conectar em um workspace do ambiente.
15. Versão - cada modificação de um arquivo.
16. Workspace – ambiente virtual. Possui os meios de comunicação para cooperar com os outros indivíduos e é onde as interações ocorrem.

C.4 Código OWL da ontologia OntoDiSEN

Essa seção apresenta a definição das principais classes da ontologia OntoDiSEN.

```
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Arquivo -->
<owl:Class rdf:about="#Arquivo">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoArquivo"/>
            <owl:onClass rdf:resource="#EstadoArquivoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#pertenceAArtefato"/>
            <owl:onClass rdf:resource="#Artefato"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temVersao"/>
            <owl:someValuesFrom rdf:resource="#Versao"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoArquivo"/>
            <owl:allValuesFrom rdf:resource="#EstadoArquivoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#pertenceAArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temVersao"/>
            <owl:allValuesFrom rdf:resource="#Versao"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Artefato -->
<owl:Class rdf:about="#Artefato">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#pertenceATarefa"/>
            <owl:onClass rdf:resource="#Tarefa"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#ehDoTipoArtefato"/>
            <owl:onClass rdf:resource="#TipoArtefato"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoArtefato"/>
            <owl:onClass rdf:resource="#EstadoArtefatoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#produzidoPor"/>
```

```
        <owl:someValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#acessadoPorParticipante"/>
        <owl:someValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#temEstadoArtefato"/>
        <owl:allValuesFrom rdf:resource="#EstadoArtefatoValuePartition"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#ehDoTipoArtefato"/>
        <owl:allValuesFrom rdf:resource="#TipoArtefato"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#produzidoPor"/>
        <owl:allValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#temArquivo"/>
        <owl:allValuesFrom rdf:resource="#Arquivo"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#geradoEmFerramenta"/>
        <owl:someValuesFrom rdf:resource="#Ferramenta"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#temArquivo"/>
        <owl:someValuesFrom rdf:resource="#Arquivo"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#geradoEmFerramenta"/>
        <owl:allValuesFrom rdf:resource="#Ferramenta"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#estaEmWorkspace"/>
        <owl:someValuesFrom rdf:resource="#Workspace"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#acessadoPorParticipante"/>
        <owl:allValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#pertenceATarefa"/>
        <owl:allValuesFrom rdf:resource="#Tarefa"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#artefatoDisponivelEmLocal"/>
        <owl:allValuesFrom rdf:resource="#Local"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#artefatoDisponivelEmLocal"/>
        <owl:someValuesFrom rdf:resource="#Local"/>
    </owl:Restriction></rdfs:subClassOf>
```

```
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#estaEmWorkspace"/>
            <owl:allValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#AtividadeProjeto -->
<owl:Class rdf:about="#AtividadeProjeto">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temTarefa"/>
            <owl:someValuesFrom rdf:resource="#Tarefa"/>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoAtividadeProjeto"/>
            <owl:onClass rdf:resource="#EstadoAtividadeProjetoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeAtividadeProjeto"/>
            <owl:allValuesFrom rdf:resource="#AtividadeProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeAtividadeProjeto"/>
            <owl:someValuesFrom rdf:resource="#AtividadeProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temTarefa"/>
            <owl:allValuesFrom rdf:resource="#Tarefa"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoAtividadeProjeto"/>
            <owl:allValuesFrom rdf:resource="#EstadoAtividadeProjetoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#FaseProjeto -->
<owl:Class rdf:about="#FaseProjeto">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoFaseProjeto"/>
            <owl:onClass rdf:resource="#EstadoFaseProjetoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temAtividadeProjeto"/>
            <owl:someValuesFrom rdf:resource="#AtividadeProjeto"/>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeFaseProjeto"/>
            <owl:someValuesFrom rdf:resource="#FaseProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoFaseProjeto"/>
            <owl:allValuesFrom rdf:resource="#EstadoFaseProjetoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
```

```
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeFaseProjeto"/>
            <owl:allValuesFrom rdf:resource="#FaseProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temAtividadeProjeto"/>
            <owl:allValuesFrom rdf:resource="#AtividadeProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Ferramenta -->
<owl:Class rdf:about="#Ferramenta">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#geraArtefatoDoTipo"/>
            <owl:onClass rdf:resource="#TipoArtefato"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Recurso"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#utilizadaPorUsuario"/>
            <owl:someValuesFrom rdf:resource="#Usuario"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#ferramentaDisponivelEmLocal"/>
            <owl:allValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#geraArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#utilizadaPorUsuario"/>
            <owl:allValuesFrom rdf:resource="#Usuario"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeFerramenta"/>
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#geraArtefatoDoTipo"/>
            <owl:allValuesFrom rdf:resource="#TipoArtefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#instaladaEmWorkspace"/>
            <owl:someValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#instaladaEmWorkspace"/>
            <owl:allValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#ferramentaDisponivelEmLocal"/>
            <owl:someValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
```

```
            <owl:onProperty rdf:resource="#geraArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#dependeDeFerramenta"/>
            <owl:allValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:allValuesFrom rdf:resource="#EstadoRecursoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Local"/>
    <owl:disjointWith rdf:resource="#RecursoMaterial"/>
    <owl:disjointWith rdf:resource="#Usuario"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Local -->
<owl:Class rdf:about="#Local">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temConfiguracao"/>
            <owl:someValuesFrom rdf:resource="#ConfiguracaoLocal"/>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#administradoPorUsuario"/>
            <owl:onClass rdf:resource="#Usuario"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Recurso"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temWorkspace"/>
            <owl:allValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temConfiguracao"/>
            <owl:allValuesFrom rdf:resource="#ConfiguracaoLocal"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localTemArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localEstaEmLocalidade"/>
            <owl:onClass rdf:resource="#Localidade"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temRecursoMaterial"/>
            <owl:someValuesFrom rdf:resource="#RecursoMaterial"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localTemArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#administradoPorUsuario"/>
```

```
            <owl:allValuesFrom rdf:resource="#Usuario"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temWorkspace"/>
            <owl:someValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localTemFerramenta"/>
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localEstaEmLocalidade"/>
            <owl:allValuesFrom rdf:resource="#Localidade"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temRecursoMaterial"/>
            <owl:allValuesFrom rdf:resource="#RecursoMaterial"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#localTemFerramenta"/>
            <owl:allValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:allValuesFrom rdf:resource="#EstadoRecursoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#RecursoMaterial"/>
    <owl:disjointWith rdf:resource="#Usuario"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Participante -->
<owl:Class rdf:about="#Participante">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#participaDeProjeto"/>
            <owl:someValuesFrom rdf:resource="#Projeto"/>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Usuario"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#produzArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#geraVersao"/>
            <owl:allValuesFrom rdf:resource="#Versao"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#realizaTarefa"/>
            <owl:allValuesFrom rdf:resource="#Tarefa"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#geraVersao"/>
            <owl:someValuesFrom rdf:resource="#Versao"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#desempenhaPapel"/>
```

```
            <owl:allValuesFrom rdf:resource="#Papel"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#possuiAcessoArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#desempenhaPapel"/>
            <owl:someValuesFrom rdf:resource="#Papel"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#participaDeProjeto"/>
            <owl:allValuesFrom rdf:resource="#Projeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#realizaTarefa"/>
            <owl:someValuesFrom rdf:resource="#Tarefa"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#possuiAcessoArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#produzArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Projeto -->
<owl:Class rdf:about="#Projeto">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#segueProcesso"/>
            <owl:someValuesFrom rdf:resource="#Processo"/>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temFaseProjeto"/>
            <owl:someValuesFrom rdf:resource="#FaseProjeto"/>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temParticipante"/>
            <owl:someValuesFrom rdf:resource="#Participante"/>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoProjeto"/>
            <owl:onClass rdf:resource="#EstadoProjetoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temFaseProjeto"/>
            <owl:allValuesFrom rdf:resource="#FaseProjeto"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeConhecimento"/>
            <owl:allValuesFrom rdf:resource="#Conhecimento"/>
```

```
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeHabilidade"/>
            <owl:allValuesFrom rdf:resource="#Habilidade"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#segueProcesso"/>
            <owl:allValuesFrom rdf:resource="#Processo"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#segueFusoHorario"/>
            <owl:onClass rdf:resource="#FusoHorario"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoProjeto"/>
            <owl:allValuesFrom rdf:resource="#EstadoProjetoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeHabilidade"/>
            <owl:someValuesFrom rdf:resource="#Habilidade"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#segueFusoHorario"/>
            <owl:allValuesFrom rdf:resource="#FusoHorario"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeConhecimento"/>
            <owl:someValuesFrom rdf:resource="#Conhecimento"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temParticipante"/>
            <owl:allValuesFrom rdf:resource="#Participante"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Recurso -->
<owl:Class rdf:about="#Recurso">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:onClass rdf:resource="#EstadoRecursoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:allValuesFrom rdf:resource="#EstadoRecursoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Tarefa -->
<owl:Class rdf:about="#Tarefa">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#possuiArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></owl:equivalentClass>
```

```
<owl:equivalentClass><owl:Restriction>
        <owl:onProperty rdf:resource="#realizadaPorParticipante"/>
        <owl:someValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></owl:equivalentClass>
<owl:equivalentClass><owl:Restriction>
        <owl:onProperty rdf:resource="#temEstadoTarefa"/>
        <owl:onClass rdf:resource="#EstadoTarefaValuePartition"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction></owl:equivalentClass>
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#dependeDeTarefa"/>
        <owl:allValuesFrom rdf:resource="#Tarefa"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#usaArtefato"/>
        <owl:someValuesFrom rdf:resource="#Artefato"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#exigeRecursoMaterial"/>
        <owl:someValuesFrom rdf:resource="#RecursoMaterial"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#usaArtefato"/>
        <owl:allValuesFrom rdf:resource="#Artefato"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#exigePapel"/>
        <owl:allValuesFrom rdf:resource="#Papel"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#dependeDeTarefa"/>
        <owl:someValuesFrom rdf:resource="#Tarefa"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#realizadaPorParticipante"/>
        <owl:allValuesFrom rdf:resource="#Participante"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#exigeFerramenta"/>
        <owl:allValuesFrom rdf:resource="#Ferramenta"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#exigePapel"/>
        <owl:someValuesFrom rdf:resource="#Papel"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#possuiArtefato"/>
        <owl:allValuesFrom rdf:resource="#Artefato"/>
    </owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
        <owl:onProperty rdf:resource="#temTarefaDependente"/>
        <owl:someValuesFrom rdf:resource="#Tarefa"/>
    </owl:Restriction></rdfs:subClassOf>
```

```
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoTarefa"/>
            <owl:allValuesFrom rdf:resource="#EstadoTarefaValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temTarefaDependente"/>
            <owl:allValuesFrom rdf:resource="#Tarefa"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeFerramenta"/>
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#exigeRecursoMaterial"/>
            <owl:allValuesFrom rdf:resource="#RecursoMaterial"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Usuario -->
<owl:Class rdf:about="#Usuario">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#senha"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#acessaWorkspace"/>
            <owl:onClass rdf:resource="#Workspace"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#login"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Recurso"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usuarioEstaEmLocalidade"/>
            <owl:allValuesFrom rdf:resource="#Localidade"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#administraLocal"/>
            <owl:allValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usaFerramenta"/>
            <owl:allValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usaFerramenta"/>
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#usuarioEstaEmLocalidade"/>
            <owl:onClass rdf:resource="#Localidade"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
```

```
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoRecurso"/>
            <owl:allValuesFrom rdf:resource="#EstadoRecursoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#acessaWorkspace"/>
            <owl:allValuesFrom rdf:resource="#Workspace"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#administraLocal"/>
            <owl:someValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Versao -->
<owl:Class rdf:about="#Versao">
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#geradaPorParticipante"/>
            <owl:onClass rdf:resource="#Participante"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <owl:equivalentClass><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoVersao"/>
            <owl:onClass rdf:resource="#EstadoVersaoValuePartition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#geradaPorParticipante"/>
            <owl:allValuesFrom rdf:resource="#Participante"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#temEstadoVersao"/>
            <owl:allValuesFrom rdf:resource="#EstadoVersaoValuePartition"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/Ontology-DiSEN.owl#Workspace -->
<owl:Class rdf:about="#Workspace">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#conectadoEmLocal"/>
            <owl:allValuesFrom rdf:resource="#Local"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemUsuario"/>
            <owl:onClass rdf:resource="#Usuario"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#conectadoEmLocal"/>
            <owl:onClass rdf:resource="#Local"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemFerramenta"/>
```

```
            <owl:someValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#acessadoPorUsuario"/>
            <owl:onClass rdf:resource="#Usuario"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemArtefato"/>
            <owl:someValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemArtefato"/>
            <owl:allValuesFrom rdf:resource="#Artefato"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#acessadoPorUsuario"/>
            <owl:allValuesFrom rdf:resource="#Usuario"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemFerramenta"/>
            <owl:allValuesFrom rdf:resource="#Ferramenta"/>
        </owl:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
            <owl:onProperty rdf:resource="#workspaceTemUsuario"/>
            <owl:allValuesFrom rdf:resource="#Usuario"/>
        </owl:Restriction></rdfs:subClassOf>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="&owl;Thing"/>
```