CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive Systems

by

Vaninha Vieira dos Santos

D.Sc. Thesis

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, OCTOBER / 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

VANINHA VIEIRA DOS SANTOS

CEManTIKA: A DOMAIN-INDEPENDENT FRAMEWORK FOR DESIGNING CONTEXT-SENSITIVE SYSTEMS

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Pernambuco as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.) in Computer Science.

ADVISOR: ANA CAROLINA SALGADO
CO-ADVISOR: PATRICIA AZEVEDO TEDESCO

RECIFE, OCTOBER / 2008

I dedicate this work to my mother, Florisa, who was always there for me, in good times and in bad times; and to my father, José Milton (in memorian), who taught me that nothing is impossible and that we should never fear change.

“Não me entrego sem lutar
Tenho, ainda, coração
Tudo passa, tudo passará...
E nossa estória não estará pelo avesso
Assim, sem final feliz
Teremos coisas bonitas para contar...
E até lá, vamos viver!
Temos muito ainda por fazer
Não olhe para trás
Apenas começamos!
O mundo começa agora...
Apenas começamos!”

– Legião Urbana

Acknowledgements

I ask you to forgive me, foreign readers, but I do not feel comfortable to write this section in English, so I will thank people in our usual communication language. Since context is a fundamental feature to support communication, I increase thankings with some contextual knowledge about the person participation in the whole doc thing to explain why these amazing people should be thanked for.

Ainda me lembro, como se fosse hoje, como foi tomar a decisão de ingressar no doutorado. O ano era 2003, tinha acabado de terminar o mestrado [Vieira, 2003], e ainda estava sob o efeito da adrelina pós defesa, do êxtase de ter realmente conseguido, e da vaidade de receber elogios e incentivos diversos de fontes heterogêneas dizendo que eu devia simplesmente "continuar". E pareceu realmente natural: porque não "continuar"? Afinal era só isso, não é mesmo? Dar continuidade a algo? Não deveria ser tão difícil assim, vamos! E inebriada com o espírito do "vamos lá" tomei a importante decisão. Já a decisão de trocar o Rio por Recife, foi um pouco mais difícil. Mas também foi tomada com a mesma confiança do "porque não?" Novo programa, novas orientadoras, novos desafios. Ah, vai ser divertido!

Bem, divertido não foi, devo confessar, e as agruras vivivas nesse período renderiam facilmente um livro (hey, porque não?). Mas não posso deixar de reconhecer que muitas pessoas tornaram essa jornada menos difícil, mais prazerosa de seguir, ou pelo menos mais enriquecedora com muita troca de conhecimento. E é a essas pessoas que eu gostaria de agradecer aqui.

A começar, por uma mulher a quem aprendi a amar e a respeitar muito nesse período de convivência: a minha orientadora e amiga ***Ana Carolina Salgado***. Ela foi o meu primeiro contato acadêmico com a UFPE, em 2000 (embora acho que ela mesma nem se lembre), quando fazia entrevista para o mestrado. Quis o destino que voltássemos a nos encontrar para uma jornada mais longa e mais intensa (e coloca intensa nisso, hein Carol? Como esquecer aquela noite na emergência do Hospital Saint-Antoine, que mais nos dava a sensação de estarmos na Restauração?). Carol, gostaria de te agradecer por tudo, tudo mesmo. Você, desde o início, demonstrou confiança em mim e me

deu carta branca para o que quer que eu quisesse fazer. Se não fiz mais, certamente não foi por falta de apoio seu. Obrigada pelas oportunidades, pelo carinho, pelos conselhos, pelos puxões de orelha (sempre bem vindos, acredite!), pelos bons vinhos que serviam de inspiração (☺), e muitíssimo obrigada pelo apoio inestimável dado em um momento realmente difícil da minha vida, e por estar lá por mim. Sem palavras para agradecer, Carol: Muito obrigada mesmo!

Outra pessoa absolutamente fundamental para este trabalho, a quem tenho muito a agradecer, é a minha orientadora e amiga ***Patricia Tedesco***. A primeira lembrança que tenho de Paxi eram os comentários nas revisões de texto ("pelamordedeus, minha filha, o que você quer dizer aqui?"). E que revisões! Críticas e comentários sempre instrutivos, construtivos, visando tornar idéias e texto melhores e mais claros. Além disso, a sua sala sempre estava aberta para sessões de terapia ou para sessões de discussões filosóficas infinitas e recursivas sobre: "o que é contexto?", "o que é foco?", "o que é situação?". Fora as reuniões de orientação inovadoras, usando a mesa do bar como *bureau*, de onde saíram algumas idéias importantes para o trabalho. Paxi, sem palavras para agradecer todo o enorme apoio recebido, os incentivos nos momentos difíceis, as broncas e críticas nos momentos certos, e a confiança transmitida todo o tempo de que tudo vai terminar bem.

Além das minhas orientadoras, muitas outras pessoas me apoiaram na parte acadêmica do trabalho, a quem gostaria também de agradecer:

- A ***Patrick Brézillon***, pour avoir accepté l'orientation de ce travail pendant l'étage de doctorat au LIP6, pour l'échange de connaissances sur le concepte de contexte, pour l'opportunité d'orienter des étudiants du master 1, et pour tout l'aide et support que j'ai reçu pendant mon séjour à Paris. Merci beaucoup !
- Aos professores ***Marcos Borges***, ***Carlos Ferraz*** e ***Nelson Rosa***, pelos valorosos comentários e sugestões oferecidos na defesa do exame de qualificação e proposta de tese, e por terem aceito participar novamente na defesa da tese; e ao professor ***Géber Ramalho***, por ter aceito participar dessa banca;
- À professora ***Cláudia Werner***, presença importante na minha formação acadêmica, com quem aprendi os primeiros passos do que é ser uma pesquisadora. Eterna fonte de inspiração e grande responsável pelo meu ingresso no doutorado, com seus incentivos e carta de recomendação. Obrigada por tudo, Cláudia, e por ter aceito fazer parte dessa banca! Sua participação é realmente especial para mim!!;
- Aos professores do CIn, que ampliaram meu conhecimento com valorosas interações, durante as disciplinas, em sessões informais de "tira-dúvida" e nos vários SAAP. Em especial a: ***Fred, Robson, Anjolina, Fernando, Valéria, Paulo Gonçalves, Augusto*** e ***Flávia;***
- A ***Helô Petry***, pelas interações sobre uso e representação de contexto e pela colaboração do ICARE neste trabalho;

- Aos alunos que tive a oportunidade de co-orientar em trabalhos de conclusão de curso e iniciação científica: ***Diego Zarate, Jorge Ferraz, Daphné Pertsekos, Aymen Lachiheb*** e ***Allan Souza***;
- Aos amigos do doutorado, parceiros no sofrimento do dia a dia, pelas discussões sobre tese, metodologia de pesquisa, e as especificidades de contexto, ontologias e afins, em especial a ***Rosalie***, ***Cadoca, Damires, Joel, Berna, Juliana***, ***Soninha*** e ***Sandra***;
- Aos colegas do CIn, com quem pude travar ótimos debates e realizar trabalhos interessantes: ***Paulo Maciel, Fábio Ávila, Patrícia Muniz, André Felipe, Marcelo, Berthônio, Flávio, Nancy*** e ***Carla Taciana;***
- Aos amigos da COPPE-UFRJ que, mesmo de longe, continuam sempre perto, quando ajuda faz-se necessária, em especial a ***Jonice Oliveira*** (Jow, sem noção todo o apoio que você me deu, sempre de forma totalmente incondicional, muito obrigada!), e a ***Leonardo Murta***, pelas diversas sessões de tira-dúvida por email;
- Ao pessoal de apoio do CIn e da secretaria da pós-graduação, pelo apoio (☺), em especial a: ***Lília***, ***Help***, ***Hilda*** (e seu precioso combustível negro), ***Melo***, e cia ltda... ;
- Às minhas professoras de francês, que tornaram possível a realização do meu grande sonho de viver um tempo nas terras de Napoleão e na cidade luz: ***Carmen Mendonça***, ***Muriel*** e ***Carminha***;
- Aos ***órgãos de fomento*** que me apoiaram em diferentes momentos dessa jornada: CAPES, CNPq e UFBA;
- Ao ***Centro de Informática*** e à ***Universidade Federal de Pernambuco***, por toda a infraestrutura que viabiliza que tudo aconteça.

Obviamente, um doutorado não é realizado apenas nas dependências de um ambiente acadêmico. Sem o apoio familiar e dos amigos, o horizonte fica negro demais e torna-se praticamente impossível prosseguir. Assim, agradeço:

- A ***Kilza***, pelo apoio constante, desde o primeiro instante em que vim para o Recife, sempre presente, dando força, incentivo, motivação, oferecendo alegria para celebrar as conquistas, e um ombro amigo, quando as lágrimas eram inevitáveis;
- A ***minha mãe*** e ao ***meu pai*** (*in memorian*), pelo conjunto da obra, por terem me feito quem eu sou, pelas orientações fundamentais que me ensinaram a valorizar as pessoas acima de tudo, a não desejar o mal, e a sempre lutar pelo que eu quisesse. Se hoje sou alguém de quem vocês se orgulham, sem dúvida a base foram vocês que construíram;
- A minha grande, imensa, ***família***, irmãs, irmãos, cunhadas, cunhados, sobrinhas, sobrinhos, minhas queridas avós (*in memorian*), aos que estiveram mais perto, aos que por causa da vida estão mais longe, mas a vocês que fazem a palavra *família* ter um sentido todo especial para mim; durante esse exílio acadêmico, sem dúvida é do aconchego do dia a dia da minha família que sinto mais falta. Amo muito vocês! Em

especial, gostaria de agradecer ao meu irmão ***Venceslau José***, financiador do meu primeiro pedido de bolsa de estudos, aos 5 anos;

- A minha querida família recifense, que me apoiou com carinho, afeto e aconchego, e que levarei comigo para sempre: ***Tia Lyra***, ***Diana***, ***Terson*** e ***Thaís***;
- Aos amigos de perto e de longe, de quem sempre ouvi palavras de incentivo e conforto durante diversos momentos dessa jornada, e em especial a alguns amigos muito queridos: a ***Marco Sacilotti***, ou Marco Aurélio (pela amizade, carinho, inesquecíveis sessões de degustação dos melhores vinhos e comida que já provei na vida, os muitos ensinamentos, *pour les bequilles*, e por ter me adotado como "filha postiça" durante meu *séjour en France*); a ***Kelli Faria*** (sem dúvida devo muito a você, sua presença foi muito intensa durante o doutorado e sua ajuda sempre muito valiosa, obrigada!); a ***Marine Varret*** (*pour ton support fondamental pendant mon séjour en France et pour ton amitié*); a ***Asif*** (*pour ton support au CFB et les séances de thè au caramel avec des discussions filosophiques sur le monde et les êtres humains*); e a ***Keila*** (minha amiga, irmã, parceira mais certa das horas mais incertas e com quem sei que posso e sempre poderei contar).

Poderia passar dias agradecendo, acho incrível como as pessoas têm importância na nossa vida, mesmo quando sua passagem é rápida. Em situações intensas, como um doutorado, essas presenças ganham um peso gigante. Assim, mesmo àqueles que por problemas de memória eu tenha esquecido de citar, deixo eternizado o meu muito, muitíssimo, obrigado!

Não poderia deixar de agradecer nesse momento, também, às ***forças espirituais superiores*** às quais sempre recorri nos momentos difíceis e, algumas vezes, esqueci nos momentos alegres. Deus, Jeová, Jesus Cristo, Senhor do Bonfim, Oxalá, Xangô, Yansã, Nossa Senhora, Anjo da Guarda, Alá, Buda, não importa o nome clamado, o fato é que sem fé tudo fica mais difícil e obscuro. Sempre que precisei e implorei por ajuda, e acreditei que ela viria, ela realmente veio, de alguma forma. Então: MUITO OBRIGADO!

Resumo

Em uma época em que os usuários precisam processar uma quantidade cada vez maior de informação e executar tarefas cada vez mais complexas em um intervalo menor de tempo, a introdução do conceito de contexto em sistemas computacionais torna-se uma necessidade. Contexto é definido como "as condições interelacionadas em que alguma coisa existe ou ocorre". Contexto é o que viabiliza a identificação do que é ou não relevante em uma dada situação. Sistemas sensíveis ao contexto são aqueles que utilizam contexto para prover informações ou serviços relevantes para a execução de uma tarefa. Projetar um sistema sensível ao contexto não é trivial, uma vez que é necessário lidar com questões relacionadas a que tipo de informação considerar como contexto, como representar essas informações, como podem ser adquiridas e processadas e como projetar o uso do contexto pelo sistema. Embora existam trabalhos que tratem desafios específicos envolvidos no desenvolvimento de sistemas sensíveis ao contexto, a maioria das soluções é proprietária ou restrita a um determinado tipo de aplicação e não são facilmente replicáveis em diferentes domínios de aplicação. Além disso, um outro problema é que projetistas de "software" têm dificuldade em especificar o que exatamente considerar como contexto e como projetar a sua representação, gerenciamento e uso. Esta tese propõe um "framework" de apoio ao projeto de sistemas sensíveis ao contexto em diferentes domínios, o qual é composto por quatro elementos principais: (i) uma *arquitetura genérica para sistemas sensíveis ao contexto*, (ii) um *metamodelo de contexto* independente de domínio, que guia a modelagem de contexto em diferentes aplicações; (iii) um conjunto de *perfis UML* que considera a estrutura do contexto e do comportamento sensível ao contexto; e (iv) um *processo* que direciona a execução de atividades relacionadas à especificação do contexto e ao projeto de sistemas sensíveis ao contexto. Para investigar a viabilidade da proposta, desenvolvemos o projeto de duas aplicações em diferentes domínios. Para uma destas aplicações, foi criado um protótipo funcional, o qual foi avaliado por usuários finais.

Palavras-chave: Sistemas Sensíveis ao Contexto, Modelagem de Contexto, Gerenciamento de Contexto, Metamodelagem, Processos de Software.

Abstract

In times when users need to process an ever increasing amount of information to perform more complex tasks in less time, the introduction of the concept of context in computer systems is becoming a necessity. Context is defined as “the interrelated conditions in which something exists or occurs”. Context is what underlies the ability to identify what is or is not relevant in a given situation. Context-Sensitive Systems (CSS) are those that use context to provide information and/or services relevant to a task execution. Designing a CSS is not a trivial task, since it is necessary to deal with issues associated to: which kind of information should be considered as context, how to represent this information, how it can be acquired and processed and how to project the context usage into the application. Although some works address specific challenges involved in developing CSS, most solutions are proprietary or restricted to specific application domains, and are not easily replicated to different applications. Moreover, another problem is that software designers lack an understanding about what exactly to consider as context, and how to represent it and design their applications to support it. This thesis proposes a framework to support the design of CSS in different domains. It is composed by four main elements: (i) a generic *context management architecture*; (ii) a domain-independent *context metamodel*, which guides context modeling in different applications; (iii) a set of *UML profiles* to account for context structure and context-sensitive behavior; and (iv) a *context process* with guidelines that cover activities related to context specification and CSS design. To investigate the feasibility of the proposal, we developed the design of two applications in different application domains. For one of those applications a functional prototype was implemented and evaluated by final users.

Keywords: Context-Sensitive Systems, Context Modeling, Context Management, Metamodeling, Software Process.

Table of Contents

List of Figures

List of Tables

Abbreviations

Term	*Description*
4WH	Who, Where, When, What,How
AI	Artificial Intelligence
API	Application Programming Interface
CE	Contextual Element
CEKB	Contextual Elements Knowledge Base
CEManTIKA	Contextual Elements Modeling and Management through Incremental Knowledge Acquisition
CK	Contextual Knowledge
CML	Context Modeling Language
CSAPI	Context Source Application Programming Interface
CSCP	Comprehensive Structured Context Profiles
CSCW	Computer Supported Cooperative Work
CSS	Context-Sensitive System
CxG	Contextual Graph
CxM	Context Management
EK	External Knowledge
ERS	Expert Recommender System
GPS	Global Positioning System
HCI	Human Computer Interaction
ICARE	Intelligent Context Awareness for Recommending Experts
IP	Internet Protocol
JEOPS	Java Embedded Object Production System
KB	Knowledge Base
MDD	Model Driven Development

MOF	Meta Object Facility
MSN	Messenger live Windows
OCL	Object Constraint Language
OMG	Object Management Group
ORM	Object-Role Modeling
OWL	Web Ontology Language
PC	Proceduralized Context
PCCB	Proceduralized Context Cases Base
PIM	Platform Independent Models
POCAp	Process for Ontological Context-aware Applications
PSM	Platform Specific Models
RDF	Resources Description Framework
SCK	Semantic Context Kernel
SeCoM	Semantic Context Model
SPEM	Software Process Engineering Metamodel
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

CHAPTER

1

Introduction

With the advance of the internet and the easy access to an increasing amount of information, people are effectively becoming dependent on computing support for making simple personal decisions or performing their daily tasks. For example, people are now relying on Computers to help them choose a movie, buy concert tickets, identify the best path to arrive to an unknown location, plan a trip, make new social contacts or even find a soul mate.

In this *information era*, where people have to process more information to perform tasks that should be executed in less time, a new challenge for computer systems arises: How to lessen the need for users' explicit interactions to obtain what they need? How to provide users with the right information necessary to accomplish their tasks? How to anticipate users' needs by suggesting options they did not even know they wanted until they saw it?

This new market demand and the dynamic and information-laden environment impose that computer system developers look for solutions that make applications more attractive to their users, more adaptable and more proactive. These new requirements can be fulfilled by the provisioning of information and services that could be interesting to users and that could assist them in the task being performed.

Context appears as a fundamental key to enable systems to distil available information into relevant information, to choose relevant actions from a list of possibilities, or to determine the optimal method of information delivery. According to the Merriam-Webster dictionary [Merriam-Webster, 2008], context is defined as “the interrelated conditions in which something exists or occurs”. Context is what underlies the ability to identify what is or is not relevant at a given moment.

Enabling computer systems to change their behavior according to the analysis of contextual information is a challenge that attracts the attention of researchers from several areas of Computer Science more and more. Computer systems that use context to provide more relevant services or information are called context-sensitive systems (CSS).

Differently from human-to-human interactions, where context is used in a natural and easy way, in human-to-computer or computer-to-computer interactions manipulating context is not trivial. Context is dynamic and the information in the context should be interpreted in order to be used. Interpretation always introduces a relevance problem, because different factors should be considered, since what is the considered as relevant to a person for performing a task, may not be considered with the same relevance by another one.

For example, when planning a touristic trip to Europe, different people may have distinct preferences about the places to visit and how long to stay at each one. While a person may privilege visiting historical and cultural places, another one may be interested in knowing the city gastronomy and making night tours. The same person may have different preferences when travelling alone, with a partner or with a group of friends. A CSS that supports users in planning their trips must certainly consider these contextual differences and also calibrate relevance issues accordingly.

Developing a CSS is a complex and expensive task. When designing a CSS one needs to deal with issues associated to: which kind of information consider as context, how to represent this information, how to acquire and process it (considering that it may come from several and heterogenous sources)

and how to integrate the context usage in the system. Context is a concept thas only recently started to be applied to computer systems. It is still immature and thus there is no consensus about definitions, terminologies and related concepts.

In a preliminary experiment (described in Appendix A) we observed that software developers have difficulties on including concept of context into their applications. They lack an understanding about what exactly to consider as context, how to represent this special type of knowledge and how to design their applications to use it. There are difficulties in separating functionalities related to the application business domain and those related to context manipulation. These observations suggest a lack of software engineering support (e.g. processes, methods, models, and architectural reuse) to aid CSS designers when developing their applications.

Most context models are proprietary, and consider only the requirements of the applications they are attached to, defining specific elements to be used in that application (e.g. [Ferrara et al., 2006, Kramer et al., 2005]). There is an open field for research on models that could abstract the specificities of context being reused and instantiated on different applications. Context Metamodels (e.g. [Vieira et al., 2008, Sheng and Benatallah, 2005, Fuchs et al., 2005]) provide a conceptual infrastructure to support the building of specific context models by abstracting and formalizing the concepts related to context. A metamodel can guide a CSS designer on elaborating their own context models.

Despite the many challenges involved in building CSS, context is generally handled in a proprietary way, without taking into account requirements such as modularity, reusability or interoperability [Riva, 2005]. Moreover, only a few approaches found in the literature (e.g. [Henricksen and Indulska, 2006, Bulcão Neto, 2006]) offer an integrated support for CSS designers that combine extensible architecture, context metamodel and software process. Existing approaches do not consider how to integrate the context model with the application conceptual model, allowing designers and developers to distinguish contextual information from existing application models.

This work investigates the specificities related to the concept of context in computer systems from the Conceptual Modeling and Software Engineering perspectives. The research carried out is targeted, especially, to designers of CSS, particularly those responsible for knowledge engineering, requirement analysis and architecture design.

The central questions of this thesis include: How to separate the specification of the functionalities related to context manipulation from the application's business requirements? How to support the context model engineering by reusing knowledge already modeled in the application domain? And still, how to support designers on including the concept of context into their application projects?

We explore the idea that it is possible to modularize the development of CSS by separating the elements related to the application business domain from the specificities associated to context manipulation. The modularization can aid the maintenance and evolution of CSS, diminishing the complexity of building CSS. In this light, we propose the specification of a framework, named CEManTIKA (*Contextual Elements Modeling and Management through Incremental Knowledge Acquisition*), to support context modeling and CSS design, in a generic, domain-independent way. The CEManTIKA approach involves the provisioning of a generic architecture, a metamodel and a software process that defines activities and concepts related to context and context-sensitive systems.

The main contributions of this research include the specification of: a *generic architecture* describing the main *architectural elements* related to a CSS; a domain-independent *context metamodel* that formalizes the concepts related to context manipulation and guides context modeling in different applications; a set of *UML* (Unified Modeling Language) *Profiles* [OMG, 2007a] to account for context structure and context-sensitive behavior; a *software process* with guidelines that cover activities related to context specification and CSS design. To investigate the feasibility of the proposal, we developed the design of two applications in different domains. For one of those applications a functional prototype was implemented and evaluated with final users. This work also represents an advance in the state of the art related to the

understanding of the concept of context (and its associated concepts). The originality of this work stands on the proposed way of thinking about context, on the proposed context metamodel and on the software process for designing CSS.

The organization of the thesis is illustrated in Figure 1-1. The contents of the chapters are detailed in the following.

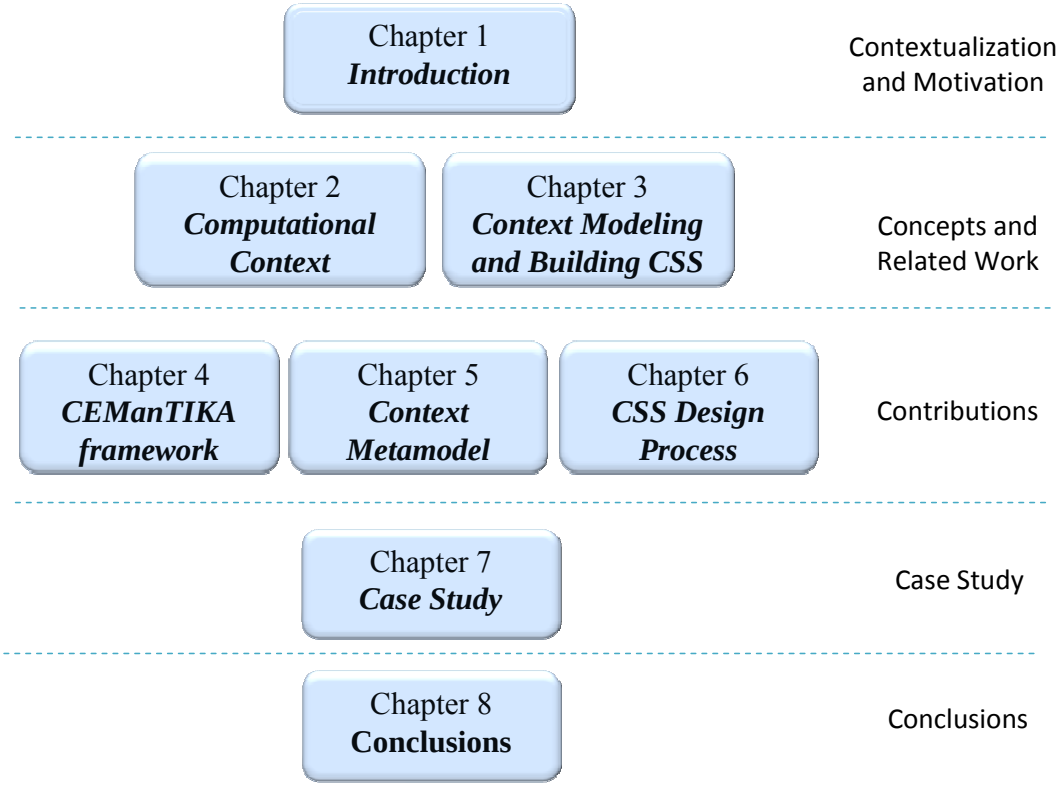

Figure 1-1 Thesis Organization

Chapter 2 presents an overview about context and context-sensitive systems, detailing different views from distinct areas of Computer Science;

Chapter 3 explores the topic of context modeling and reviews the state of the art about proposals to support CSS building, discussing their particularities, strengths and weaknesses and identifying improvements that could be done;

Chapter 4 presents an overview of the CEManTIKA approach. The main ideas behind the framework proposal are discussed and a generic architecture for designing CSS is described;

Chapter 5 presents the proposed *Context Metamodel* and the UML profiles built to support modeling the structure and behavior of a CSS;

Chapter 6 presents our proposal for a *Context Process* that identifies activities and offers guidelines to support context modeling and CSS design;

Chapter 7 discusses, in a case study, the instantiation of the CEManTIKA framework, guided by the *Context Process*, in an Experts Recommender System, named ICARE. The chapter also discusses how a functional prototype of ICARE was implemented and the results of its evaluation by final users;

Finally, **Chapter 8** summarizes the proposed work by discussing the achieved contributions and indicating some directions in which the presented research could be extended.

CHAPTER

2

Computational Context

Context is being object of study of researchers from several areas of Computer Science. Since it lacks a consensus about the concepts related to context, the distinct areas have different views about context, how to define it and how to consider it in computer systems. This chapter presents an overview about the concept of context and how it is being considered in computer systems

This chapter covers the following contents: Section 2.1 reviews definitions and issues related to context and contextual elements; Section 2.2 presents the definition of context-sensitive systems and discusses concerns related to the development of CSS; Section 2.3 presents a review about researches that is being developed on context in distinct areas of Computer Science. Finally, Section 2.4 concludes the chapter with some final considerations.

2.1 Context and Contextual Elements

This section defines the concept of context, discusses some characteristics related to this concept and points out a difference between context and contextual elements.

2.1.1 Definitions

Bazire and Brézillon have catalogued more than 150 definitions about the term *context* [Bazire and Brézillon, 2005], and have thus concluded that the definition about what to consider as context varies strongly according to different domains.

A widely referenced definition states that context is any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application [Dey, 2001]. Zimmermann and colleagues [Zimmermann et al., 2007], in seeking for an operational definition of context, extended Dey's definition by separating the elements that characterize the situation of an entity into 5 categories: *individuality* (properties and attributes defining the entity itself), *activity* (all tasks the entity may be involved in), *location and time* (spatio-temporal coordinates of the entity), and *relations* (information about any possible relation the entity may establish with another entity).

McCarthy's observations [McCarthy, 1993] indicate that: (1) a context is always relative to another context, (2) context has an infinite dimension; (3) context cannot be described completely; and (4) when several contexts occur in a discussion, there is a common context above all of them to which all terms and predicates can be lifted.

Kokinov and colleagues [Kokinov, 1999] developed a dynamic theory of context that defines context as the set of all entities that influence human (or system) behavior on a particular occasion. This theory has four main principles: (1) context is a state of the mind; (2) context has no clear cut boundaries; (3) context consists of all associatively relevant elements; and (4) context is dynamic.

Although there are several definitions of context, researchers agree that: context exists only when related to another entity (e.g. task, agent or interaction); context is a set of items (e.g. concepts, rules and propositions) associated to an entity; and an item is considered as part of a context only if it is useful to support the problem at hand. For example, the proposition "it is

raining" is considered as part of the context in a *traffic jam support system*, since rain has implications in visibility, speed and consequently in the traffic. However, the same proposition is not contextual information in a *museum guide system*.

In this thesis, we make a distinction between the conceps of *context* and *contextual element* and the adopted definitions are described below:

> *A **contextual element** **(CE)** is any piece of data or information that enables to characterize an entity in a domain.*
>
> *The **context** of an interaction between an agent and an application, in order to execute some task, is the set of instantiated contextual elements that are necessary to support the task at hand.*

By this latter definition, we are particularly interested in context applied to the interaction between an agent and an application. An *agent* can be a human agent or a software agent. Moreover, the elements to compose the context have a relevance relationship with the task that the agent is performing. We can observe that a CE is stable and can be defined at design time, while a context is dynamic, and must be constructed at runtime, when an interaction occurs.

2.1.2 Representational versus Interactional Views

Dourish [Dourish, 2004] distinguish the problem of context from two points of view: context as a *representational* problem and context as an *interactional* problem. The former argues that when thinking about context usage in software systems (which are representational, by nature) the central concern is to identify how context can be encoded and represented. The latter, based on Social Science investigations of everyday activity, argues that a central concern with context is the question ''how and why, in the course of their interactions, do people achieve and maintain a mutual understanding of the context for their actions?''.

As discussed in [Chalmers, 2004], definitions of context in the representational view emphasise objective functionalities that can be tracked

and recorded relatively easily, and avoid aspects of the user experience such as subjectively perceived features and the way past experience of similar contexts may influence current activity. The interactional view focuses on intersubjective aspects of context, constructed in and through the dynamic of each individual's social interaction.

According to the representational view, Dey [Dey, 2000] identify some inner characteristics of the contextual information that difficults its usage and manipulatation: it *can be acquired from non-traditional devices*, different from mouse and keyboard (e.g. environment sensors, presence identifiers, or voice recogniser); it *may have low granularity* implying that it may be abstracted to make sense to the application; it *may be acquired from multiple distributed and heterogeneous sources*; it *may change rapidly* implying that these changes must be detected in real time; and *contextual history should also be considered*.

In this work, we assume a hybrid representation, combining aspects of the representational and the interactional views. The representational view is used for the concept of *contextual element*. It means that any CE is a form of information that can be known, encoded and represented, it is possible to define in advance what will be considered as a CE, it is stable and we can indicate relevance associations between a CE and other entities in the application (e.g. agent, task). However, for the concept of *context* we agree with the interactional view which indicates that what will be considered as relevant in the context will be defined dynamically and will depend on a particular interaction or task execution [Dourish, 2004].

2.1.3 Context Classification in Three Types of Knowledge

Brézillon and Pomerol [Brézillon and Pomerol, 1999] propose a model that separates and classifies the context according to a focus of attention. They argue that context cannot be considered in an isolated way, but always related to a focus. That focus can represent a task, a step in a problem solving or in a decision making. The focus determines what should be considered as relevant in the context.

According to the focus, they classify the context into three distinct parts, as shown in Figure 2-1: Contextual Knowledge (CK), External Knowledge (EK)

and Proceduralized Context (PC). In [Pomerol and Brézillon, 2001] they justify the use of the word "knowledge" in the description of parts of the context since these parts are about the general background used by users to carry out their tasks. Context is considered as a shared knowledge space that is explored and exploited by participants in the interaction.

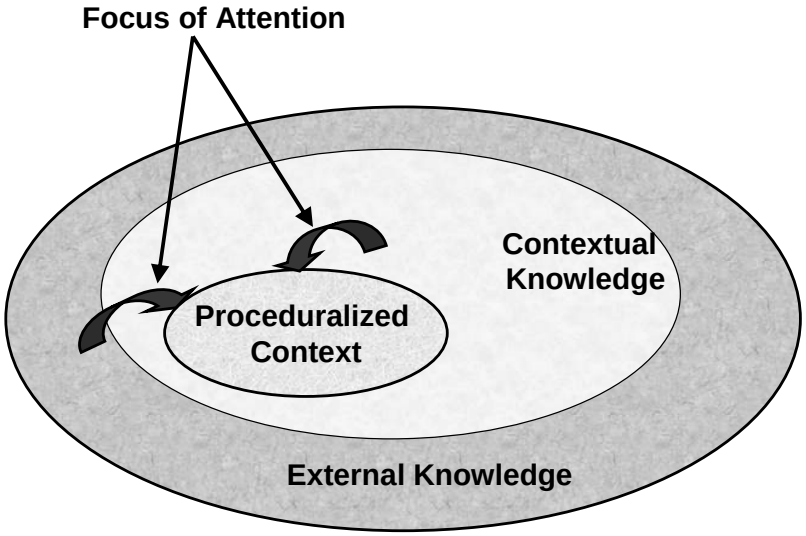

Figure 2-1 Context Classification According to the Focus [Brézillon and Pomerol, 1999]

EK represents the part of the knowledge that has absolutely no relevance to the current focus and that is not necessary to support the task. For example, suppose that a user's focus is to find experts to support her/him in a software development task. So, the EK set may include elements such as the user's height?, his/her marital state, or the printer location, which are existing knowledge related to users, experts and resources, but that are not really relevant to support the task in the focus.

CK represents the knowledge that is relevant and has a strong relation to the focus. In the previous example, the CK set will include information such as the experts' location, presence, availability, abilities, reputation, experience, the devices being used, the software development application and language. CK acts as a filter that defines, at a given time, what knowledge pieces must be taken into account (explicit knowledge) from those that are not necessary (implicit knowledge).

Finally, PC is the subset of the CK that is invoked, organized, structured and situated according to the focus. It is the part of the context that will indeed be used to support the focus. In the example of the software development task, the PC set includes information such as the identification that a user named Charles is present, available and is an expert in Java language, and that another user called John is present, but busy, and has experience in UML. The CK is a backstage knowledge whereas the PC is the knowledge immediately useful for the task at hand [Pomerol and Brézillon, 2001].

2.2 Context-Sensitive Systems

This section presents definitions about context-sensitive systems, a classification for those systems based on our observation about the works in the area, and some challenges involved in the development of these applications.

2.2.1 Definitions

Providing applications with the ability to identify and understand the context of an interaction between the application and its users can greatly improve the communication between users and machines. The ideal application should be able to provide information that is both accurate and relevant without requiring the user to actively seek this information and determine its relevance.

Developers of computer systems are seeking ways to build applications that are more adaptive, flexible and easy to use. The idea is to provide services that transparently ease the interface between human and machines. To this end, Context-Aware Computing [Schilit et al., 1994] studies how knowledge about context may assist applications in adapting their behavior providing information and services closer to users' needs.

The term *context-aware system* is used to refer to systems that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [Dey, 2001]. Other terms are used as synonyms to designate these systems: *context-sensitive system* [Sato, 2004, Cheverst et al., 1999], *context-oriented system* [Desmet et al., 2007] and *context-based system* [Kashyap and Sheth, 1996].

In this work, we adopt the term *context-sensitive system* (CSS), since our goal is to consider context usage by any kind of application. We believe that the most popular term "context-aware system" has an embedded semantic association with Ubiquitous Computing applications. In this work we assume the following definition for CSS:

> ***Context-sensitive systems (CSS)*** *are those that manage contextual elements related to an application domain and that use these elements to support an agent executing some task. This support can be achieved by improving the agent's cognition about the task or by providing adaptations that ease the task performance.*

Similarly to our definition of context, we consider the context usage in a CSS as always associated to support an agent to execute some task. An agent can be a human or a software agent. The basic element manipulated by a CSS are contextual elements. Moreover, the provided support can be associated either to provide adaptations in the system behavior or to improve the agent's cognition about relevant information related to the task.

2.2.2 Different Views on Designing CSS

By analysing different usages of context in computer systems, we consider three main axes for dealing with the relativity of context: *device-centered*, *task-centered* and *human-centered* views.

- *Device-Centered View*

This view considers context from a technology angle. It is motivated by recent advances in devices and sensors technologies along with the requirements of Ubiquitous Computing applications. Main concerns are related to the automatic capture of different types of information mostly related to the physical environment of users and devices (e.g. location, screen size, battery level, or network signal).

Context is associated to indirect and non-traditional acquisition methods (e.g. sensors), to replace conventional information capture methods (e.g. mouse

and keyboard). Most works concentrate on solving issues related to networking, mobility and distributed applications, such as:

1) identification of acquisition modes and interpretation of environmental information (e.g. location, presence or devices' characteristics) (e.g. [Chaari et al., 2007, Yang et al., 2006]);

2) investigation of modeling techniques to support context sharing and interoperability [Gu et al., 2004, Chen and Finin, 2004]; and

3) development of service-based solutions [Raz et al., 2006, Gu et al., 2005, Riva, 2005].

To automatically infer contextual information it may be necessary to combine data from several sources. Therefore, techniques for data processing [Nurmi and Floréen, 2004, Giunchiglia, 1993] and uncertainty treatment [Ranganathan et al., 2004, Korpipää et al., 2003] are also investigated.

- *Task-Centered View*

The *task-centered view* is based on works developed by Cognitive Science (e.g. [Kokinov, 1999]), Artificial Intelligence (AI) (e.g. [Brézillon, 2007b, McCarthy, 1993]) or Computer Supported Cooperative Work (CSCW) researchers (e.g. [Brézillon and Araújo, 2005]).

Context is considered as a way to extract relevant knowledge that could support a task at hand. Main challenges are related to identifying the knowledge related to a task development and how the context influences the steps to be followed to accomplish the task. It can demand a great deal of knowledge about the artifacts involved in the task, the different roles a person can perform while executing it, and the reasoning that constrains each step execution.

Context acquisition is harder to perform, since the granularity of the managed contextual information is higher. The automatic acquisition can demand AI techniques, such as interaction analysis [Siebra, 2007] or data mining [Vajirkar et al., 2003].

- *Human-Centered View*

In this view researchers are interested in investigating how context affects human interactions with an application [Brézillon, 2003a], how tacit knowledge can be modeled [Gonzalez and Brézillon, 2008] or how context adaptation will affect the human actor [Bellotti and Edwards, 2001]. Primarily, the problem is associated to personalization and to providing support for decision making. Researches are mainly related to the Human-Computer Interaction area (e.g. [Kiniry, 2004, Bellotti and Edwards, 2001]), to CSCW applications (e.g. [Nunes et al., 2007, Kirsch-Pinheiro et al., 2005, Ferscha et al., 2004]), and to recommender systems (e.g. [Petry et al., 2008, Ning et al., 2007, van Setten et al., 2004]).

Contextual information is mainly collected from user profiles, existing databases, and interaction history. There are many human aspects that cannot be modelled accurately, sensed or inferred by technological means (e.g. people's intentions, perceptions, interpretations or emotions). The use of AI techniques are necessary, specially to analyze past interactions to identify preferences (e.g. [Pan et al., 2007, Chedrawy and Abidi, 2006]).

2.2.3 Usability Issues in CSS

The development of a context-sensitive system entails more work in comparison to systems that do not take context into account. Designing a system to react according to the context is a challenging task, since misinterpretations may entail undesired behaviors, which will make the system annoying and disturbing instead of useful.

As pointed out by several researchers (e.g. [Dourish, 2004, Greenberg, 2001, Bellotti and Edwards, 2001]) CSS cannot be designed to act on behalf of their users. Users must feel that they have control over the system by being able to grant or to refuse systems' actions in a nonobtrusive fashion. Also, they should have the option to regulate system's adaptations and reasoning rules according to their needs. Users must be able to understand the system actions, especially when they are automatically performed according to the context. In this case, the application should provide explanations about the rationale used

to perform these actions and enable the user to contribute through feedbacks [Bellotti and Edwards, 2001].

The adaptation to the context should occur in a peripheral, non-intrusive fashion, in order to not disturb the users from their current task. In addition, a CSS should be designed with the premise that there is a strong likelihood the system will get things wrong [Greenberg, 2001, Bellotti and Edwards, 2001]. Consequently, they should be conservative in the actions they take, making these actions visible, and leaving "risky" actions to user control. In fact, we see the usage of context as more successful when used in discrete, non-intrusive fashion.

A succesful example of this state of affairs is provided by Google[1]. Despite their excelent approach for page ranking and clean and easy to use interface, two other key factors can be considered for explaining Google success: the first law on Google's philosophy [Google, 2008] is to focus on the user experience first and the revenue will follow; on the business side, Google innovates with its AdSense program that enables targeted advertisements more relevant and useful to users, based on the information displayed on any given page. Its co-founder, Larry Page, said: "The perfect search engine would understand exactly what you mean and give back exactly what you want"[Google, 2008].

A counter-example is the Microsoft Office Clip Assistant, named Clippy. Clippy is a little paperclip who politely offers tips for using Office. In [Swarts, 2003], the author analyzes why so many people hate using Clippy. One identified reason is that people don't like being told what to do (especially when they already know how to do it). Another problem is Clippy's intrusiveness. Users fell like being looked over their shoulders in addition to being disturbed from their current task by annoying and irrelevant questions and suggestions. In Office's more recent versions, Clippy was replaced by another feature called *smart tags*. It is a purple line that underlies a part of a text and has an associated pop-up menu. This menu provides actions related to the underlined text, which the user can activate or not. This feature is less intrusive and enables the user to command the system actions.

[1] http://www.google.com/corporate/

2.3 Researches on Context in Computer Science

The researches involving context in Computer Science may be divided into two main categories. On the one hand there are researchers interested in applying this concept to their applications to improve the services and functionalities offered by their approaches (e.g. [Park et al., 2007, Siebra, 2007]). On the other hand, there are researchers interested in context as a concept, looking for ways to treat it computationally, through formalizations, models, frameworks and methodologies (e.g. [Vieira et al., 2008, Gonzalez and Brézillon, 2008, Hirschfeld et al., 2008]). Results from the latter are applied in researches conducted in the former. Our research is situated in this second group.

In the following sections, we make a review about researches on the concept of context conducted by different areas of Computer Science.

2.3.1 Context in AI

Context started to appear in Computer Science associated to the area of *Artificial Intelligence (AI)*. AI researchers investigate the *formalization* of the notion of context (e.g. [McCarthy, 1993]), the provisioning of *foundational grounds* (e.g. [Akman, 2002]), the usage of appropriate *modeling and reasoning techniques* (e.g. [Franklin, 2003, Bouquet et al., 2003, Giunchiglia, 1993]) and the application of context in the development of *smart applications* (e.g. [Petry et al., 2008, Park et al., 2007]).

2.3.2 Context-Aware Computing

The term Context-Aware Computing was first used in [Schilit et al., 1994] to designate the systems that are capable to examine the computing environment and react to changes in it. This view of context is mostly associated to Ubiquitous Computing, the area envisioned in 1991 by Weiser [Weiser, 1991] of computing “anytime, anywhere, from any device”. Advances in technology, such as the broad usage of small devices, wireless communication, and more sophisticated sensors open possibilities for context-aware systems.

2.3.3 Context Support on Social Interactions

In *Human Computer Interaction* (HCI) context appears strongly related to the problem of *personalization* in topics such as *human-centric computing* [Ranganathan and Lei, 2003, Brézillon, 2003a, Bellotti and Edwards, 2001], *embodied interactions* [Dourish, 2004] and *interface adaptation* [de Bra et al., 2004]. Some authors (e.g. [Zimmermann et al., 2005b]) differentiate *personalization* from *contextualization*, arguing that the former adapts the application according specifically to the user (e.g. needs, goals, knowledge and interests), while contextualization complements personalization so that environmental states and task modeling can also be taken into account.

In the CSCW area, context is strongly related to the concept of *awareness* [Brézillon et al., 2004, Vieira et al., 2004, Gross and Specht, 2001, Rittenbruch, 1999]. It is also used to support the management of *shared workspaces* [Gutwin et al., 2005] and to support the *analysis of interactions* [Siebra, 2007]. An important aspect of context in CSCW is that not only the context of an individual should be considered but also the context of a group, through the idea of *shared context* [Brézillon and Araújo, 2005]. Borges et al. [Borges et al., 2007]discuss the problems that arise from the loss of context when people in the group disagree about the understanding of the shared context, which they call *context mismatch*.

2.3.4 Context Role in Content and Information Manipulation

The *Databases* community investigates how techniques for *information representation* and *conceptual modeling* can support context modeling [Stefanidis et al., 2005, Christopoulou et al., 2004]. Context also appears as an important tool to support *content management* [Belotti et al., 2004], solving semantic conflicts in *data integration* systems [Belian, 2008], improving *query processing* [Bunningen, 2004], reducing the search space and otimizing pattern identification in *data mining* [Vajirkar et al., 2003], and also in issues related to the storage, analysis and manipulation of contextual information [Harvel et al., 2004].

Context also appears as an important tool to improve solutions in *knowledge management* [Zacarias et al., 2005, Degler and Battle, 2000],

decision support systems [Nguyen and Gonzalez, 2006, Bucur et al., 2005], *information filtering* [Kirsch-Pinheiro et al., 2005], and *digital television* [Leite et al., 2007].

2.3.5 Software Engineering for CSS

Since context entails new requirements in the development of computer systems, researchers from *Software Engineering* perceived the need to provide *methodologies* [Ayed et al., 2007, Desmet et al., 2007, Bulcão Neto, 2006] and *architectural support* [Costa, 2007, Bardram, 2005, Constanza and Hirschfeld, 2005, Henricksen, 2003, Dey et al., 2001] to aid the development of CSS. Context is also investigated as a tool to improve software engineering processes [Santoro et al., 2005].

2.4 Concluding Remarks

In this chapter we reviewed the concepts of context, contextual element and context-sensitive systems. We also discussed how context research is being performed by distinct areas of Computer Science. This review supported a better understanding of the concept of context, reinforcing its significance in computer systems.

To illustrate the importance that is being given to context in Computer Science, in 2006 a Brazilian committee composed by researchers from several areas defined the five big challenges in Computer Science for the next 10 years [SBC, 2006]. Context explicitly appears in three of these challenges as a basis to support: information retrieval in large volumes of data by providing information more appropriate to user's preferences and needs; to consider human aspects in the construction of flexible and adaptable interfaces and contents, with the objective to enable the access to knowledge and digital media to all citizens; and in challenges associated to the Ubiquitous Computing area to enable computing any time, anywhere, from any device.

This work investigates the concept of context from the Conceptual Modeling and Software Engineering perspective. Our interest is to investigate techniques for representing contextual information and to provide support for

designing context-sensitive systems. However, the area of Artificial Intelligence offers the motivations for using context (smart and adaptable systems) and offers models and formalisms to support processing and reasoning about context.

Next chapter discusses issues related to context modeling and presents related approaches that aim to provide support on building CSS.

CHAPTER

3

Context Modeling and Support on CSS Design

As previously discussed, developing CSS entails more work in comparison to applications that do not consider the context: in the former, one must care for context-related tasks, such as the acquisition, processing, storage, manipulation and presentation of contextual information. Context modeling is an important topic when developing CSS, since the context model captures the structure and semantics of the contextual information and identifies how this type of information can be manipulated.

Researches related to context modeling focus on: (1) identifying representation techniques that better fit the characteristics of contextual information (e.g. [Strang and Linnhoff-Popien, 2004]); (2) enumerating the elements that should be considered as context in a domain or a set of applications (e.g. [Souza et al., 2008, Cruz et al., 2007, Siebra, 2007, Rosa et al., 2003]); and (3) guiding the context modeling by providing generic context models (e.g. [Bulcão Neto and Pimentel, 2005]) and metamodels (e.g. [Vieira et al., 2008, Sheng and Benatallah, 2005, Fuchs et al., 2005]).

Generic context models aim to describe the information that can be considered as *context* in a generic way. They provide a classification for an initial set of elements that compose the context in a domain. Applications can reuse the modeled information extending it to support their particularities.

There are proposals of generic context models for different areas such as pervasive systems [Chaari et al., 2007], groupware [Vieira et al., 2005c], data integration [Souza et al., 2008], intelligent environments [Gu et al., 2005], software reuse [Cruz et al., 2007].

To support the design and development of CSS, researchers investigate the provisioning of *architectural support*, such as frameworks (e.g. [Bardram, 2005, Henricksen and Indulska, 2004, Klemke, 2000]), middlewares (e.g. [Gehlen et al., 2007, Gu et al., 2005, Sacramento et al., 2004]) and toolkits (e.g. [Zimmermann et al., 2005a, Dey et al., 2001]). However, few solutions on Software Engineering for CSS have been reported (e.g. [Bulcão Neto et al., 2006, Henricksen and Indulska, 2004]), and this is still an open area for research.

This chapter discusses some issues related to modeling context and reviews some approaches that offer support for designing CSS. It begins by reviewing existing techniques used to represent contextual information (Section 3.1). Section 3.2 details one of these techniques, the Contextual Graphs, used in this work to model the context dynamics. Section 3.3 discusses some approaches related to this work, with emphasis on proposals that integrate generic solutions on context modeling and on software engineering support. Section 3.4 presents a discussion about the reviewed approaches and concludes the chapter with some final considerations.

3.1 Techniques for Representing Contextual Information

An issue that was discussed by several authors (e.g. [Roque, 2005, Henricksen et al., 2004, Strang and Linnhoff-Popien, 2004]) is what existing techniques for knowledge and information representation best suits context requirements. Some of these techniques are briefly described in the following.

3.1.1 Key-value pairs

The *key-value pairs* (e.g. [Zimmermann et al., 2005a, Dey et al., 2001]) is the technique that uses the simplest data structure for representing context. Contextual information is described through pairs composed by a key that

identifies the information, and a value associated to this key. For example, the pairs (date = *after* April 16; time = *between* 8 and 12 a.m.; location = room 35; presence = person arrival) indicate a configuration of a context model where the contextual information are date, time, location and presence.

Actions can be associated to these instances indicating what a system should do when contextual information fits the parameters. For example, when a person arrives in the room 35, between 8 and 12 a.m. after the day April 16, the system should play a good morning message. Due to its simplicity, it is easy and fast to build a context model using this technique. However, it does not provide more sophisticated structures to organize the information. The consequence is that models developed using this approach are hard to maintain and the identification of contextual information should be done using exact string matching algorithms. That makes this technique inefficient for more complex problems.

3.1.2 Markup schemas

Markup schemas use the standard XML (*eXtensible Markup Language*) to model contextual information (e.g. [Ryan, 1999]). XML has as main characteristics the presence of hierarchies that contain markup tags with attributes and contents. Since it is a standard, XML eases the information sharing between heterogeneous systems. An example of markup schemas model is the CSCP (*Comprehensive Structured Context Profiles*) [Held et al., 2002] that represents context as session profiles. String exact matching is used to retrieve an element, and the attribute names are interpreted according to their position in the XML schema.

3.1.3 Topic maps

A *Topic Map* is a framework for information retrieval [Garshol, 2004]. It is a subject-based classification technique that associates individuals, easing navigation and visualization of contents. While the other techniques concentrate in defining and formalizing the concepts in a domain, topic maps focuses on the instances, how they are connected to each other and how to reach individuals by semantic relationships between them.

Some works (e.g. [Goslar and Schill, 2004, Power, 2003, Degler and Battle, 2003]) propose using *topic maps* for implementing context models. They argue that topic maps apply well when one needs to integrate contextual information that come from several heterogenous context sources. The disadvantages on using topic maps also include immaturity on tools and standards. In addition, topic maps lack formalisms for the information representation, which difficults reasoning. This approach to focus on individuals instead of concepts difficults information maintenance and may impact on the application performance.

3.1.4 Ontologies

An ontology is a specification of a conceptualization [Gruber, 1993]. A domain ontology establishes a common vocabulary for information sharing in a knowledge domain [Guarino, 1998]. Ontologies combine expressivity (to ease human reading) and formality (to ease machine processing) that enable knowledge sharing between human and software agents. People should commit to use a specific ontology for a domain of interest. The advent of standard languages for representing ontologies, such as OWL (*Web Ontology Language*) [Bechhofer et al., 2004], eases knowledge reuse between systems, and enables the usage of existing inference engines.

In the last five years, a huge number of ontology-based context models have being proposed for different areas. Examples include ontologies for Ubiquitous and Pervasive Computing [Chaari et al., 2007, Yau et al., 2006, Chen, 2004], Intelligent Systems [Gu et al., 2004, Preuveneers et al., 2004], Collaborative Systems [Vieira et al., 2005c], Geographical Information Systems [Souza et al., 2008], Knowledge and Content Management [Ferrara et al., 2006, Zacarias et al., 2005, Gauvin et al., 2004] and Service-Oriented Systems [Costa, 2007, Bulcão Neto and Pimentel, 2005].

A drawback related to ontologies, reported in works, such as [Bulcão Neto, 2006, Henricksen et al., 2004] and empirically verified in studies developed in our group [Zarate, 2006, Vieira et al., 2006b], is that the tools and standards for manipulating ontologies are still immature and hard to use. Moreover, reasoning over ontologies impacts in the application performance.

3.1.5 Graphical models

Graphical models are particularly useful for structuring the contextual information, using graphical elements, such as diagrams. Examples include UML (*Unified Modeling Language*), ORM (*Object Role Modeling*) and contextual graphs. *UML* provides graphical elements to support designing the structure of the context model (following an object-oriented approach) and also the system behavior (e.g. [Simons and Wirtz, 2007, Sheng and Benatallah, 2005]). Henricksen extended *ORM* diagrams with elements to ease the specification of context characteristics [Henricksen and Indulska, 2006], creating the graphical notation that composes the CML (Context Modeling Language). *Contextual graphs* are based on semantic networks [Brézillon, 2005] and support the representation of task models. They consider the relation between procedures stablished by an organization for executing a given task, and how contextual information influences the real execution of these tasks (practices). Other methods for conceptual modeling, such as the Entity-Relationship Models [Chen, 1976] can also be used for modeling contextual information.

3.1.6 Discussion

During the development of this work we have analyzed and experimented some of these techniques, with the motivation to identify advantages and disadvantages. XML was used in a work that aimed to investigate issues associated to context acquisition [Ferraz, 2006]. In special, we worked with ontologies, using OWL [Bechhofer et al., 2004], to represent context in the domain of groupware systems [Vieira et al., 2005b, Vieira et al., 2005c] and investigated implementation and reasoning issues related to ontologies [Zarate, 2006, Vieira et al., 2006b].

Table 3-1 presents a summary of the advantagens and disadvantages of each investigated technique, indicating its appropriateness. Key-value pairs and markup-based languages are the easiest techniques, but they do not offer support to visualize the modeled information and are hard to maintain and evolve, especially in complex applications. Although topic maps provide the flexibility to integrate and navigate through information coming from different

sources, it still lacks tools to support their creation and manipulation. Ontologies are relevant to formalize the concepts to be considered in the context, supporting interpretation and information sharing. This is by far the most used approach to model contextual information, due to its support on formalization and reasoning. However, ontologies only model the structure of a context model and does not offer support on modeling CSS behavior.

Table 3-1 Summary of Context Representation Techniques

Technique	Strong Points	Weak Points	Processing/Retrieval
Key-value pair	Simple structure, easy to implement and use	Do not consider hierarchies. Not suitable for complex systems.	Linear search using exact string matching
Markup Language	Consider hierarchies. Standard – eases context sharing.	Data incompleteness or ambiguities are not considered. Not suitable for complex systems.	Query language, based on XML
Topic Maps	Easy navigating between concepts and model readability by humans.	Immature tools; lack formalisms	Navigation through semantic networks
Ontology	Improve context semantics definition. Standards enable the use of existing tools for inferencing. Improve model readability by humans and machines.	Immature tools, overhead in system performance.	Inference engine, query language based on frames or OWL
Graphical Models	Ease concepts design and specification, and defining context requirements	Do not allow concepts processing. Needs additional data structure	Can be translated to XML format and use the XML processing tools

Conceptual modeling, as defined in [Mylopoulos, 1992], is the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication. Conceptual modeling is relevant in a system's design process since conceptual specifications are used to support understanding, problem-solving, and communication among stakeholders about a given subject domain.

Graphical models are useful to support context conceptual modeling since it supports to visualize how pieces of contextual information relate to each other and how context affects the CSS behavior. UML extensions are particularly interesting to enable the reuse of existing application models and

modeling tools. Contextual graphs are useful to describe the context dynamics identifying how context affects the behavior of a CSS. The problem with these approaches is that they provide only visual representations of the models, and they should be translated into a more formal and expressive structure (e.g. Ontologies or OO-models) to be instantiated and manipulated. Since each approach has strong and weak points, a hybrid approach seems to be appropriate.

Due to its relevance for this work, next section details the Contextual Graphs technique. UML extensions are further explained in the Appendix B.

3.2 Modeling Context Dynamics with Contextual Graphs

A Contextual Graph (CxG)[2] is a representation formalism proposed by Brézillon and colleagues [Brézillon, 2007b, Brézillon, 2003b, Brézillon, 2003c, Brézillon et al., 2002] to model contextual reasoning. It models the reasoning involved in solving a problem, executing a task, or making a decision, considering specific conditions in which these activities are accomplished. Their motivation for this formalism is based on the observation that there is a discrepancy between the way companies establish work procedures and the way these procedures are effectively executed by operators in their daily routine (the practices). When solving a problem, different operators develop their own practice, tailoring an existing procedure in order to take into account their current context, which is particular and specific [Brézillon, 2003c].

A CxG is an acyclic directed graph that represents the actions to undertake according to the context, indicating a path from a problem to a solution. Each contextual graph (and any sub-graphs in it) has exactly one root (representing the initial state), one end node (representing the final state) and a serial-parallel organization of nodes connected by oriented arcs. The branches show different paths expressing different contextually-dependent ways to achieve the final state. The path presents a time-directed representation, indicating the sequence of actions to be performed in the problem solution.

[2] http://www.cxg.fr

The main concepts of a contextual graph (illustrated in Figure 3-1), are [Brézillon, 2003c]:

1) *Contextual node*: corresponds to a contextually-dependent decision. It has one input and *n* outputs (branches). The node contains the identification of the element to be analysed. Each output branch corresponds to a possible instantiation value for that element. In Figure 3-1the contextual nodes correspond to the numbered circles;

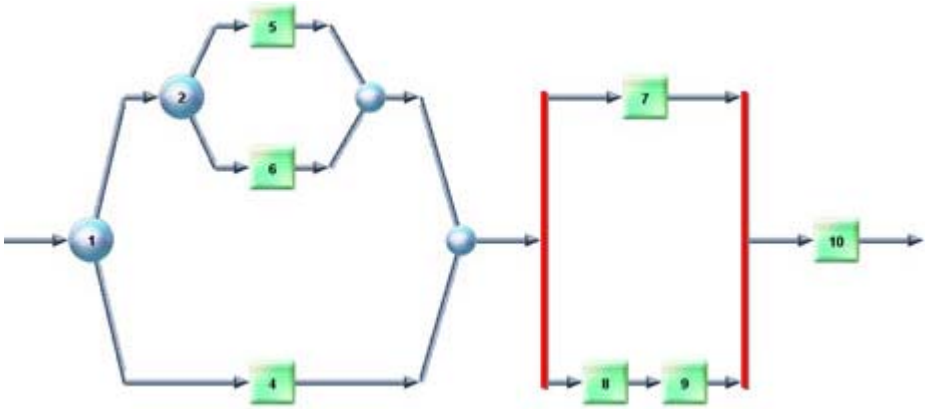

Figure 3-1 Concepts in a Contextual Graph [Brézillon, 2007a]

2) *Action*: is the basic node in a contextual graph and represents an executable method. Actions correspond to the squares in Figure 3-1;

3) *Recombination node*: is always associated to a contextual node, having *n* inputs and one output. The input branches represent the convergence of the different alternatives, identified in the contextual node, towards a same action sequence to execute after the condition is verified. In Figure 3-1 the recombination nodes correspond to the small circles;

4) *Parallel action grouping*: represents a set of *m* actions that can be performed in parallel, or in any order, but all must be accomplished before to continue to the next step. The ordering of the actions to execute in a parallel action grouping depends on contextual elements that do not appear in the contextual graph. This is a way to deal with the incompleteness of local information;

5) *Activity*: is a complex action assembling different elements, such as another contextual graph. A change in an activity appears in all contextual graphs where the activity has been identified.

The initial structure of a CxG (its skeleton) is defined by an established procedure. An example is illustrated in Figure 3-2, describing the reasoning involved in solving a video problem in a DVD player.

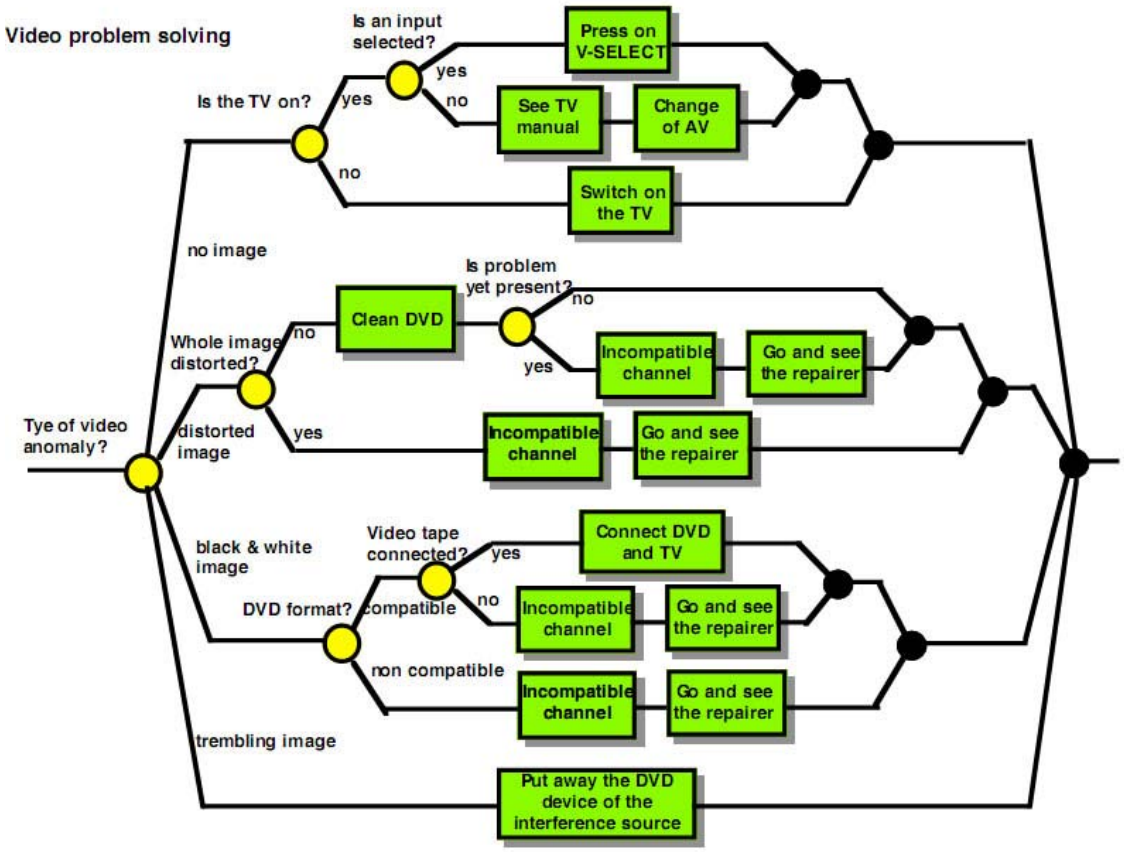

Figure 3-2 Contextual Graph for a Video Problem Solving Process for a DVD player [Brézillon, 2007b]

The graph starts by the contextual node that verifies the element "Type of video anomaly". Four instantiations are defined for this element: "no image", "distorted image", "black & white image", and "trembling image". Assuming the first instantiaton (i.e. [Type of video anomaly = no image]), next step is another contextual node, which verifies the element "Is the TV on". The possible instantiations for this element are: "yes" and "no". Assuming the first instance [Is the TV on = yes], the next step is another contextual node with the element "Is an input selected", with possible values: "yes" and "no". Assuming the second instance [Is an input selected = no], the next step is to perform in sequence the actions "See TV manual" and "Change of AV". The next step indicates a sequence of three recombination nodes which, respectivelly,

finalizes the contextual nodes “Is an input selected”, “Is the TV on”, and “Type of video anomaly”.

Contextual graphs appear as an interesting approach to model a CSS behavior, since it describes the actions that should be performed by the application and explicitly indicates how context affects these actions.

3.3 Approaches for Supporting Context Modeling and CSS Design

This section reviews some approaches that aim to support the design of CSS. Three aspects were investigated: the provided architectural elements, their support on context modeling and their support on processes for CSS design. We detail the approaches that offer integrated support for these three issues. Other related approaches are discussed in Section 3.3.4.

3.3.1 Software Engineering Framework for CSS

A software engineering framework to support CSS building is presented in [Henricksen and Indulska, 2006, Henricksen, 2003]. To support modeling context, they propose a graphical modeling notation called Context Modeling Language (CML), conceived as an extension to the Object-Role Modeling (ORM) [Halpin, 2006]. Figure 3-3 illustrates the CML notation (box *Key*) and its usage in an example. The elements inside the circles represent objects from the modeled domain (e.g. *Person*, *Activity*, *Device*). Fact types enable to identify associations between two objects (e.g. *engaged in*, *located near*, *has channel*). The contextual information is represented in terms of fact types. Additional notation was introduced to classify the fact types (*sensed*, *static*, *profiled* or *derived*), to associate quality metadata (e.g. *Certainty*) and to indicate inference rules for derived fact types (e.g. *located near*, *engaged in*).

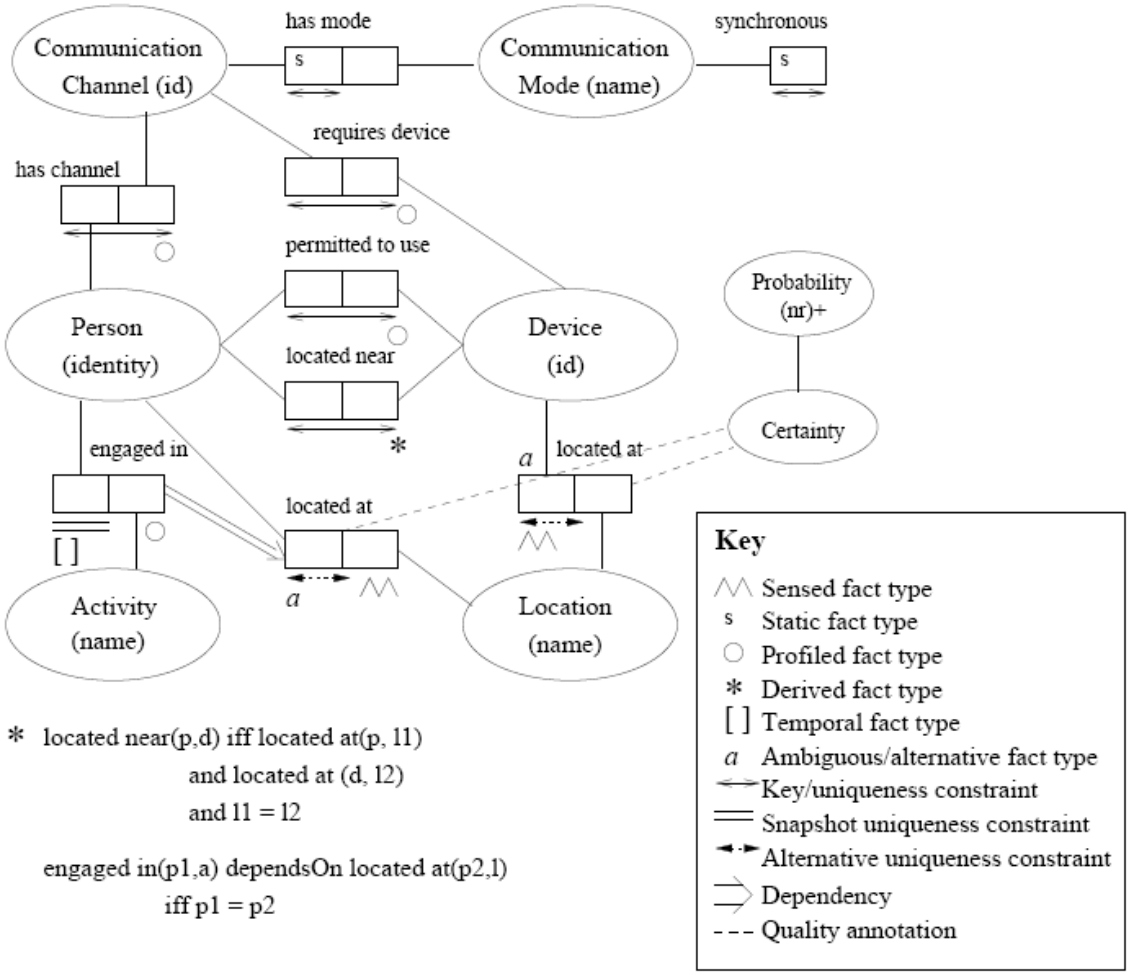

Figure 3-3 Example of a CML Context Model [Henricksen and Indulska, 2006]

They propose a programming toolkit based on the management of situations (logical expressions associated to the context fact types) and preferences (scoring expressions to classify the relevance of context fact types). To support the usage of this programming toolkit and the management of CML models they developed a softwate architecture organised into six loosely coupled layers (Figure 3-4): *context gathering layer*, acquires context information from sensors and processes it through interpretation and aggregation; *context reception layer*, translates inputs from the context gathering into the CML representation; *context management layer,* maintains a set of CML context models and their instantiations; *query layer* provides an interface to query the context management layer; *adaptation layer* manages repositories of situation, preference and trigger definitions, and evaluates these

on behalf of applications' needs; and *application layer* supports applications through their programming toolkit model.

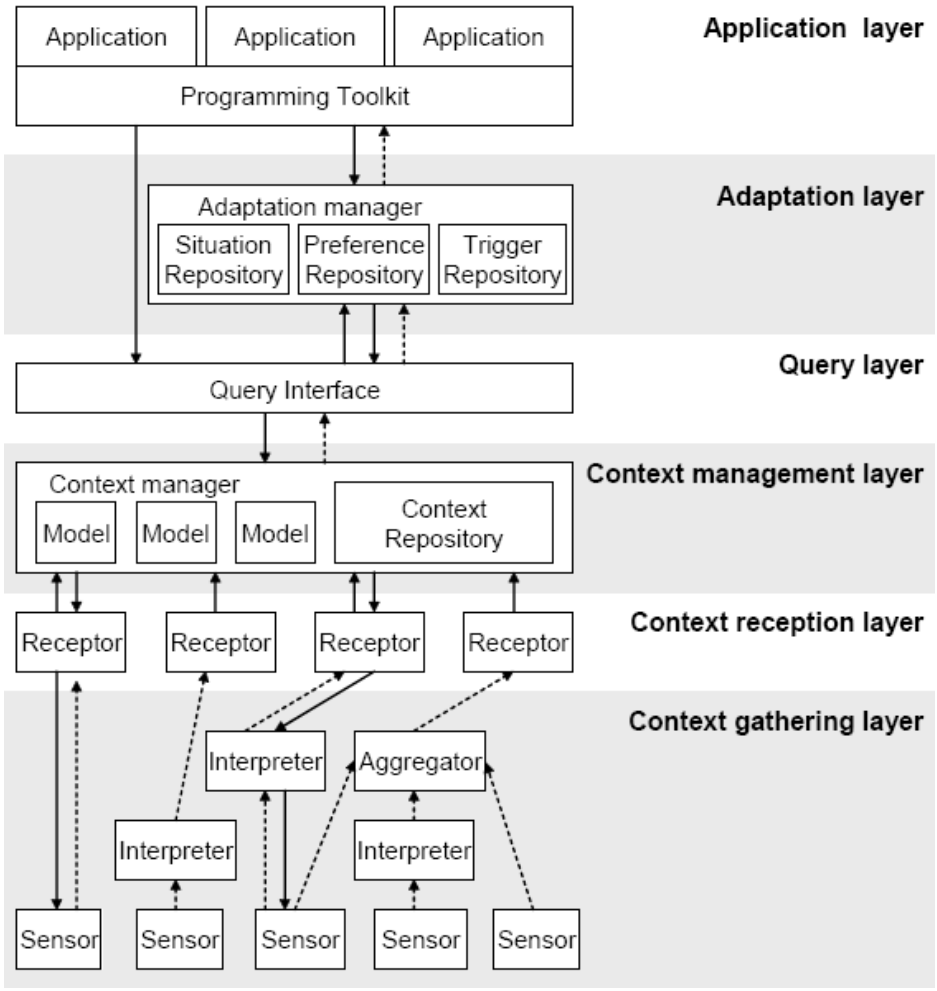

Figure 3-4 Layered Architecture of the Software Infrastructure [Henricksen, 2003]

To guide the development of applications using their proposed tools (CML model and architecture) they propose a software engineering methodology [Henricksen and Indulska, 2006], which specifies five main activities in the CSS development: *Analysis* and specification of the context fact types; *Design* of the triggering mechanisms for the application; *Implementation* of the application according to the programming toolkit; *Customisation* of the abstract models (mapping of the CML model into relational models and identification of samples for testing); and *Testing* (modules, overall system and application acceptance with final users).

This approach is very interesting since it combines the assistance of an expressive context modeling notation with a modularized architecture and provides a set of guidelines to support the developer to use these artifacts. However, although authors argue in pro of using ORM as base notation, this is a controversial advantage since this notation is not largely used by systems developers. Besides, the proposed graphical notation does not have any tool to support context modeling, which difficults its usage in practice. The adoption of more widely used notations (e.g. UML) could improve the model sharing and reuse.

The proposed methodology only indicates the high level activities to be performed and provides a flow to indicate the sequence of execution. It does not specify the details related to each activity, such as: which artifacts can be used as input or guidance to support the activity execution, which work products are produced at each activity; and which role, in the software development team, should be assigned to perform each activity. Moreover, they do not indicate how existing application models could be reused to construct the context models.

3.3.2 The SeCoM-SCK-POCAp Approach

Bulcão Neto [Bulcão Neto, 2006] proposes a software engineering approach to support the development of ontology-based context-sensitive systems. The approach is composed by three elements: the *Semantic Context Model* (SeCoM), a set of ontologies related to different dimensions of a contextual information [Bulcão Neto and Pimentel, 2005]; the *Semantic Context Kernel* (SCK), an infrastructure to manipulate ontologies; and the *Process for Ontological Context-aware Applications* (POCAp), a structured set of activities for developing ontology-based CSS.

SeCoM (Figure 3-5) is a generic context model composed by several ontologies, each one responsible for providing semantic descriptions about a dimension of the contextual information. These dimensions, related to an interaction, are divided according to the 4WH questions: *Who* are the interaction's participants? (Actor ontology), *Where* does the interaction take place? (Spatial ontology), *When* does the interaction take place? (Time

ontology), *What* does the interaction describe? (Activity ontology), and *How* is context captured and accessed in the interaction? (Devices ontology). Some support ontologies (Knowledge, Relationship, Role, Contact, Document and Project) model aspects related to the actors. The SCK is an infrastructure developed to process the SeCoM model. It extends the services of the Jena framework [Jena, 2006] to support ontology management, querying, persistence and inferencing.

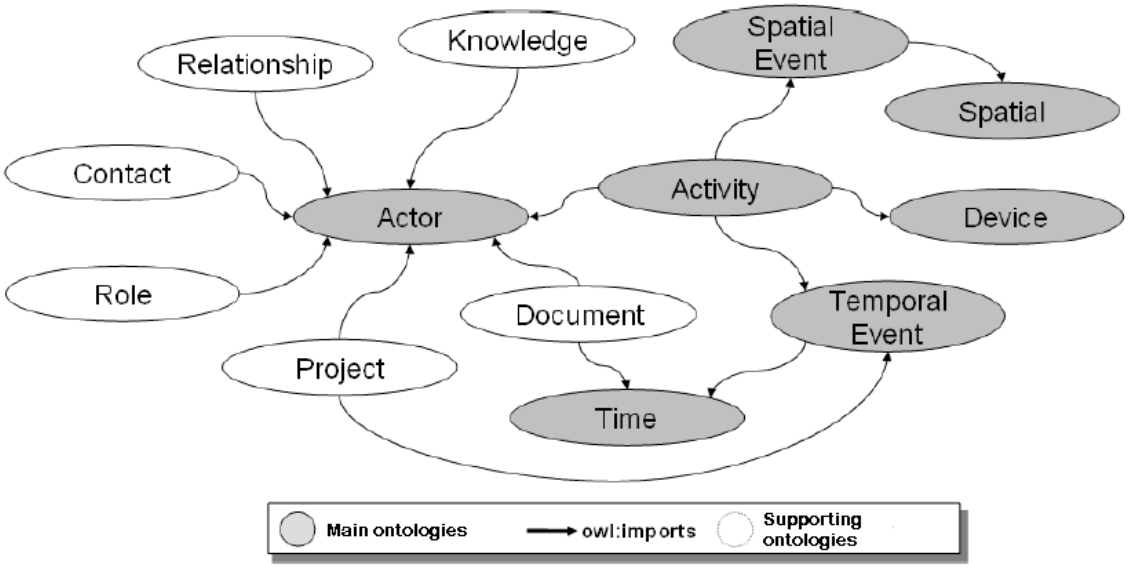

Figure 3-5 SeCOM: Overview of the Association between the Defined Ontologies [Bulcão Neto et al., 2006]

POCAp is a software process proposed as a structured set of activities for developing ontology-based CSS [Bulcão Neto et al., 2006]. The process is modeled using the Software Process Engineering Metamodel (SPEM) [OMG, 2008a]. It considers the CSS development according to the four main activities: *analysis and specification*, *design*, *development*, and *verification and validation*. For instance, Figure 3-6 illustrated the *analysis and specification* activity.

The *analysis and specification* activity should be performed by an *Analyst* who must execute four activities in the following sequence: (a1.1) *requirements analysis and specification*; (a1.2) *analysis and specification of context information*; (a1.3) *analysis and specification of model reuse*; and (a1.4) *analysis and specification of model extension*. The SeCoM model is used as input to support activities a1.3 and a1.4. The SCK is used to support the *design*

activity, indicating the ontology manipulation services to compose the CSS being developed.

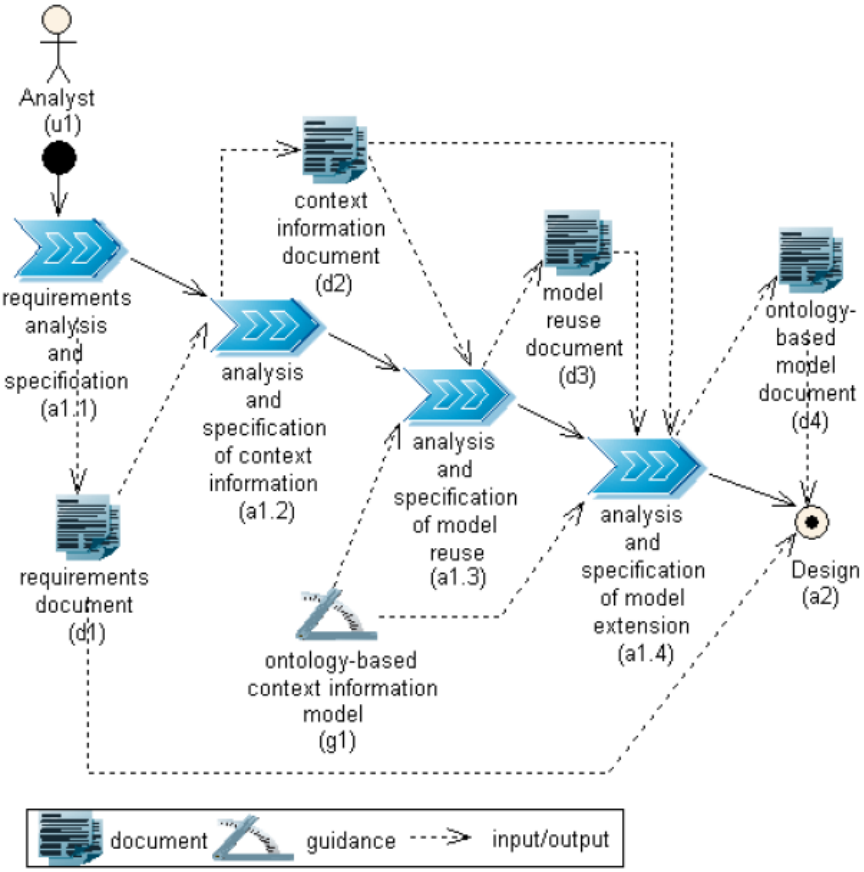

Figure 3-6 POCAp: Analysis and Specification Activity [Bulcão Neto et al., 2006]

This approach is centered on the modeling and manipulation of the ontologies associated to a CSS, assuming that the CSS is ontology-based. The POCAp process is an interesting proposal since it frames and organizes different activities and issues related to the development of context-aware applications. The SPEM notation provides elements to detail the activities, specifying input and output work products as well as guidelines that can assist each activity execution. Main drawbacks of the overall proposal includes: it is limited to ontology-based CSS; it does not address the issues related to context manipulation in a broad sense; the offered support is related to challenges associated to ontologies creation and manipulation; and the context model do not integrate information about the context dynamics and the CSS behavior.

3.3.3 MDD-Based Approaches for CSS

Model Driven Development (MDD) [OMG, 2003] is an approach that uses machine-readable models at various levels of abstraction to build software. The key idea is to automatically transform highly abstract models into more concrete models from which an implementation can be generated in a straightforward way.

A MDD-based approach for CSS was proposed in [Ayed et al., 2007]. Contextual information is modeled according to the UML Profile presented in Figure 3-7. It describes the following stereotypes: (i) *Context*, indicates the context type; (ii) *CollectionProcess*, represents the elements necessary to acquire the context; (iii) *ContextQuality* indicates quality attributes to be satisfied by the context; (iv) *ContextState* specifies conditions associated to the context type, in the form <contextType; operator; contextValue>. To identify composition of context states, they define two types of associations, the stereotypes *and* (for *binary conjunction*) and *or* (for *binary disjunction*).

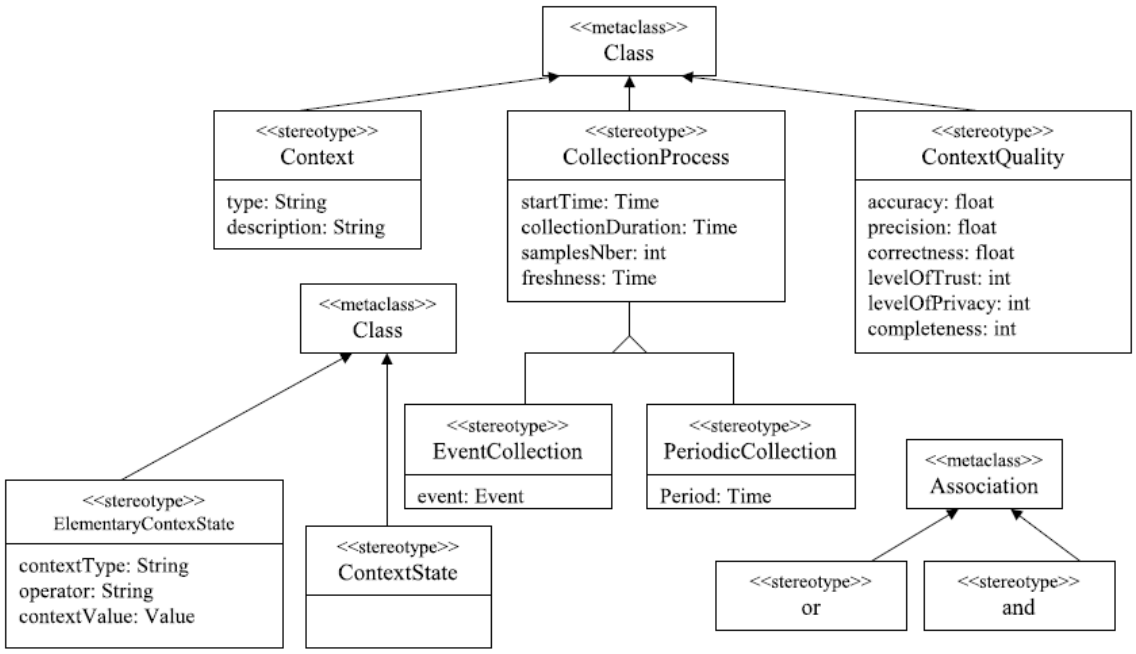

Figure 3-7 UML Profile for Context Structure Modeling [Ayed et al., 2007]

Based on the MDD specification, Ayed et al. [Ayed et al., 2007] propose a set of steps for developing CSS, composed by six phases classified into four layers (Figure 3-8). The first and second layers aim to describe the Platform Independent Models (PIM) for the CSS. A PIM is independent of specific

implementation platforms and contains a set of high level abstract models which defines the CSS elements. These layers comprise the following phases: (1) *identification of the required context information*; (2) *definition of the application variability;* (3) *identification of the context collection mechanisms*, and (4) *identification of the adaptation mechanisms*.

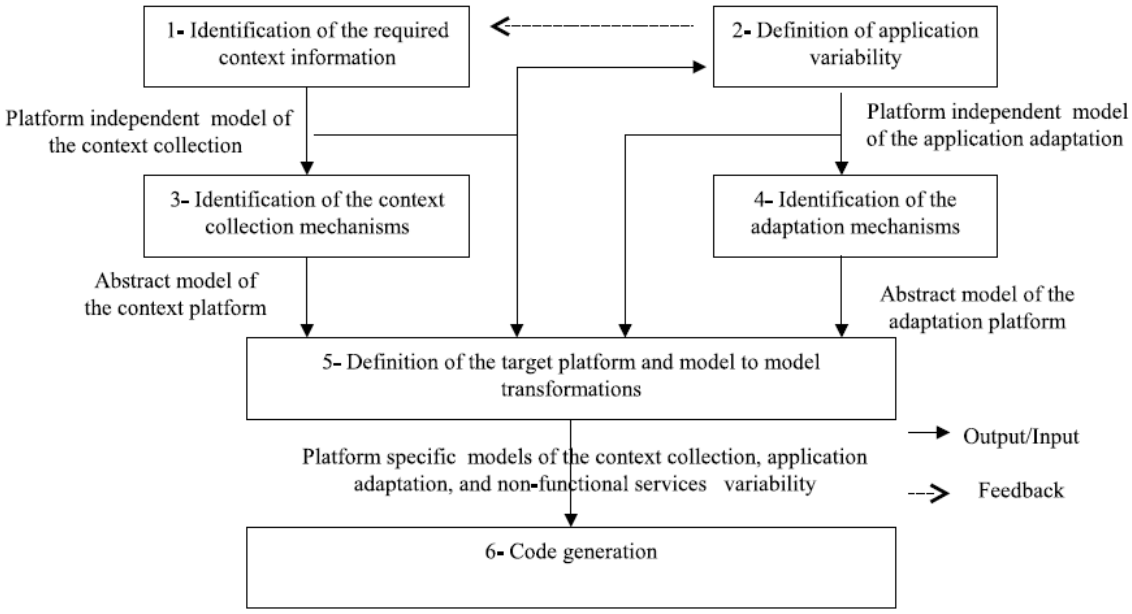

Figure 3-8 MDD Phases for the Development of CSS [Ayed et al., 2007]

The third layer intends to identify the CSS implementation platform and to generate the Platform Specific Models (PSM). A PSM is a representation of the same system defined in the PIMs containing all technical details that are needed to realize the system on a concrete technology platform. The mapping between PIM and PSM is realized using an automatic transformation. In this sense, the PIMs defined in phases (1) to (4) are automatically transformed into PSMs in the phase (5) *definition of the target platform and model to model transformations*. Finally, the fourth layer defines the phase (6) *code generation* which specifies model to code transformations in order to generate the CSS code.

It is possible to observe, in the recent context community literature, an increasingly number of proposals that use MDD and UML Profiles to assist the development of CSS (e.g. [Simons and Wirtz, 2007, Farias et al., 2007, Seyler et al., 2007, Sheng and Benatallah, 2005, Van den Bergh and Coninx, 2005]). Advantages of the MDD approach is that code generations follow specifications

defined in conceptual models, which improves maintenance and quality of the produced software. However, this is still an immature technology not easy to implement in real and complex projects.

3.3.4 Other Approaches

Many approaches for context modeling have been proposed. In this section, we discuss some other works that are related to the context modeling approach proposed in this thesis.

Bucur et al. [Bucur et al., 2005] propose to use two different and integrated ontologies: one that defines a domain ontology (similar to other context ontologies) and another ontology that describes the context attributes managed by the CSS. A *context attribute* designates the information defining one element of context. They propose a generic representation for a context attribute modeled as a class (Figure 3-9). Each context attribute has a name, an indication of the number of entities it is related, the list of entities, at least one value, where the value depends on the entities to which the attribute relates. This approach is interesting in the sense that they separate the context definition from the concepts defined in the application domain. However, they describe only the structure of the context attribute and do not consider other elements related to context, such as acquisition parameters.

Property Name	Property Type	Domain	Range	Multiple values
name	Datatype	#ContextAttribute	String	No
noEntities	Datatype	#ContextAttribute	Integer	No
entitiesList	Object	#ContextAttribute	#Entity	Yes
valueType	Object	#ContextAttribute	#Entity	No
multipleValue	DataType	#ContextAttribute	Boolean	No

Person – related	Time-related
IsSupervisorOf :(Person, Person)-> Boolean StatusPerson :(Person) -> String Supervises : (Person) -> Person* RoleOfPersonInGroup :(Person,Group)-> Role	TimeZone : (Time) -> Integer DayOfWeek : (Date) -> String TimeOfDay : (Time) -> String

Figure 3-9 Description of the Context Attribute Class and Examples of instances of related to Person and Time [Bucur et al., 2005]

The ContextUML metamodel [Sheng and Benatallah, 2005] is a UML profile developed to support the modeling of context-aware Web Services. As illustrated in Figure 3-10 it separates the modeling activity into two categories:

context modeling (definition of context types and context sources) and context-awareness modeling (definition of context binding to objects and triggering of actions). The disadvantage of this approach is that it is strongly related to the Web Services category of CSS. The proposed UML extension is heavyweight (i.e. it modifies the semantics of the UML), meaning that it cannot be used by existing UML modeling tools.

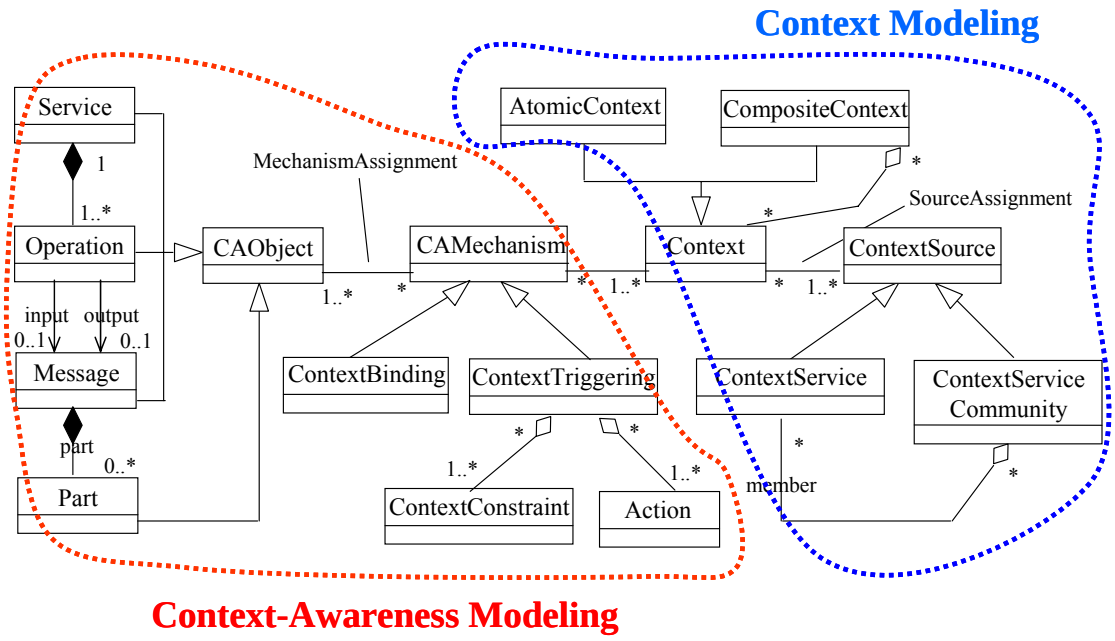

Figure 3-10 ContextUML Metamodel [Sheng and Benatallah, 2005]

Costa et al. [Costa, 2007] propose a context model based on foundational ontologies for conceptual modeling [Guizzardi, 2005]. Their context model is based on three foundational concepts: *Entity*, *Context*, and *Context Situation* (Figure 3-11). In addition, context is characterised as either *intrinsic context* (it belongs to the essential nature of an entity) or *relational context* (it depends on the relation between distinct entities). A *formal relation* (e.g. *greater than*, *subset of* and *nearness*) can be defined between two pieces of context. A *Context Situation* is composed of entities, contexts or other situations and exhibits the time interval during which the situation holds.

Drawbacks of this approach is that the way context and entities are modeled may lead to ambiguities and conflicts, since it is difficult to understand the differences between the two concepts. For example, to indicate that a class Container is a context and not an entity they modeled it twice as

ContainerEntity and ContainerContext [Costa, 2007]. It does not consider reusing existing application models, compelling context models to be developed from scratch as a new separate model. Also, they do not consider relevance association between context and systems' users or the task the user is involved. As a consequence, all contexts have the same weight and relevance no matter who the current user is or if a user is performing different roles.

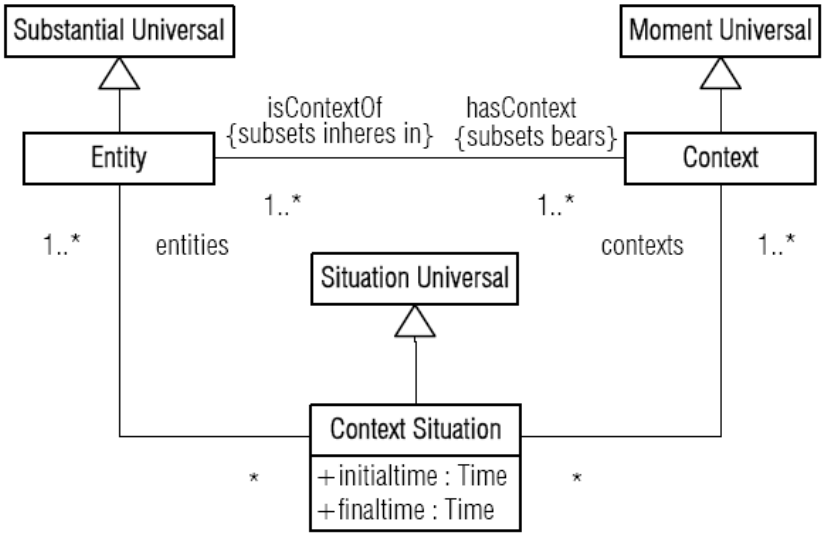

Figure 3-11 Fragment of the Foundational Context Concepts [Costa, 2007]

3.4 Concluding Remarks

This chapter presented an overview about context modeling and approaches to support the design of CSS. Some techniques for representing contextual were analyzed and the contextual graphs approach for context dynamics modeling was detailed.

Most architectural support for building CSS focus on providing mechanisms for integrating contextual information provided by multiple services and sensors to ease context sharing. They offer services for resource and service discovery and intermediate the communication between context providers and applications. The information they manage is generally provided by sensors, such as location and presence of users and devices. They are strongly influenced by the requirements imposed by the areas of Ubiquitous Computing and Smart Spaces (smart homes, mostly). Context processing and

reasoning activities are mainly related to ontology consistency and classification and the inference of high level contexts from low level contexts.

Current context-sensitive systems base their context model on existing languages, such as OWL [Ferrara et al., 2006, Wang et al., 2004], without considering the reuse of a context metamodel. We believe that, since these languages are conceived for general purposes (not specifically for treating context particularities), they offer little support and abstractions for building context models. There is a lack of research and consensus in specification of generic context models and context metamodels. In metamodels a challenge is to identify what concepts should be considered, what they mean, how they are related to each other, and how to formalize them.

The analysed software processes for designing CSS proposals *describe* the specification phase as a fundamental step. However, they do not detail how that specification should be done. Specially, they do not support the association of the contextual information according to the adaptation it is intended to support. They consider the adaptation specification isolated from the context identification. They do not consider reusing existing arteficts already produced by the application developers or domain specialists for identifying the application unaware behavior.

Although much work has been proposed to address part of the discussed challenges (software infrastructure, context modeling and software processes), there is still a lack of integrated, domain and technology independent, generic solutions, to support designing CSS.

Next chapter describes our proposal for a domain-independent framework, which is centered on a generic and extensible architecture for CSS, a context metamodel and a software process with guidelines to support the activities related to context specification and CSS design.

CHAPTER

4

A Framework for Designing CSS

In the previous chapters we presented the motivation and the core theoretical issues of this thesis. This chapter presents an overview of our proposal for a framework to support the design of CSS, named CEManTIKA (*Contextual Elements Modeling and Management through Incremental Knowledge Acquisition*). A framework is a generic term for an object-oriented reuse technique that typically emphasizes the reuse of design patterns and architectures [Kobryn, 2000]. It includes models, design projects and abstract classes that are specifically designed to be refined and adapted for specific applications.

The CEManTIKA framework is centered on three main objectives: (1) to support the design of architectural elements related to context manipulation; (2) to support context specification and representation in a generic domain-independent manner; and (3) to aid developers on modeling context and designing CSS. In this sense, the artifacts offered by CEManTIKA comprise: a *Context Manipulation Architecture*; a *Context Metamodel* (presented in Chapter 5), and a *CSS Design Process* (described in Chapter 6). This chapter emphasizes the architecture description.

The development of CEManTIKA follows two axes: (i) *theoretical*, which entails understanding the concept of context and the dynamics associated to its usage; and (ii) *practical,* concerning the framework instantiation in a CSS by implementing the metamodel concepts and architectural elements according to the specified process.

Before describing the architecture and its elements (Section 4.4), Section 4.1 presents our working definition of context, Section 4.2 presents a conceptual definition of the activities involved in a CSS, and Section 4.3 discusses the aspects related to dealing with context dynamics.

4.1 Our Working Definition of Context

Our working definition of context is based on two definitions, presented in Chapter 2. The first states that context is any information that can be used to characterize the situation of an entity (e.g. person, place, object, application) [Dey, 2000]. The second indicates that context is always related to a focus and that, at a given focus, the context is the aggregation of three types of knowledge: Contextual Knowledge (CK), External Knowledge (EK) and Proceduralized Context (PC) [Brézillon and Pomerol, 1999]. An illustration of our working definition of context is shown in Figure 4-1, which is divided into two parts: conceptual view and implementational view.

The conceptual view presents Brézillon and Pomerol's view about context [Brézillon and Pomerol, 1999]. The EK, CK and PC are part of a Context. The transformations from EK to CK and from CK to PC are guided by the focus. The implementational view is an extension of the conceptual view designed to turn their definition into a computable one. The central element of this view is the *Contextual Element* (CE). For us, a CE is any piece of data or information that is associated to an *Entity* in an application domain, and that can be used to characterize that entity. The concepts EK, CK and PC are constructed as a composition of CEs.

When dealing with knowledge-based systems it is necessary to separate the part that represents statements about the world (i.e. CEs) from the part that indicates learned associations between those statements (i.e. rules). When

activated, the rules trigger pre-defined behaviors. Rules affect the transformation from CK (the set of all manageable CEs) to PC (the subset of instantiated CEs relevant to the Focus). The generated PC affects the triggered behavior.

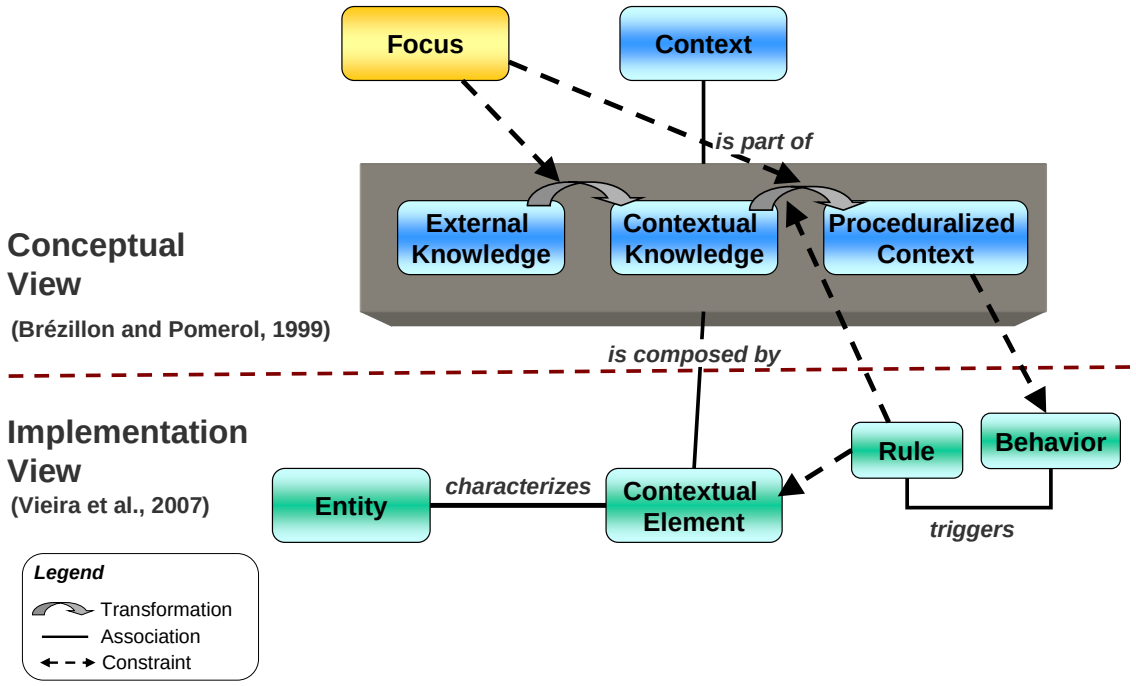

Figure 4-1 Illustration of our Working Definition of Context

To treat context computationally it is important to make this distinction between context and contextual element. C*ontext* is a dynamic concept that is constructed at run time when a focus is identified. *Contextual Element* is a static concept, defined at design time, and used to compose the context.

4.2 Classification of the Tasks Involved in CSS Development

As defined in Section 2.2, a CSS is an application that uses the knowledge about context to provide relevant information and services to its users. In this sense, a CSS development project must consider a set of tasks specifically related to the context manipulation. We classify these tasks into three main categories [Vieira et al., 2008]: *context specification***,** *context management* and *context usage*.

1) *Context specification* refers to the identification of the possible variations in a CSS behavior that can be affected by the context, and the definition of *what* should be considered as context to support the decision about a variation triggering;

2) *Context management* (CxM) is related to *how* context will be implemented and used in the system and is defined in terms of the main tasks it comprises, as follows:

   ```
   CxM = acquisition + storage + processing + dissemination
   (of CEs);
   ```

3) *Context usage* refers to the employment of the specified and managed CEs to guide the variations in the CSS behavior, either by enhancing users' awareness, by influencing recommendations, or by enabling adaptations of any kind.

We consider that a CSS may be seen from two different perspectives: a part that is *domain-dependent* (context specification and context usage) and another *domain-independent* part (context management). *Context specification* and *context usage* are strongly dependent on the CSS being developed. Different domains or different applications will demand different sets of CEs and will imply in different considerations for their usage. On the other hand, *context management* can be modularized and treated in a domain-independent fashion, since it encompasses the mechanics of dealing with context according to defined CEs and rules. A *context management system*, or *context manager* for short, involves the definition of solutions to enable the separation of context-related tasks from the applications' business features [Vieira et al., 2007b].

As illustrated in Figure 4-2, a context manager is an intermediate layer between context sources and context consumers, and it aims at providing CEs acquired from these sources to interested consumers. Context sources are software elements (e.g. external bases, physical or logical sensors, profiles, or user dialog interfaces) that can provide up-to-date information about the entities considered in the CSS domain. Context consumers are software elements that identify relevant CEs to support the triggering of a context-sensitive behavior. Both, sources and consumers, are linked to the context manager through interfaces.

The context manager circumscribes the needed mechanisms: to *acquire* CEs from multiple context sources; to *process* and semantically interpret the acquired CEs, according to defined and learned rules; to *store* the sensed and inferred CEs in a shared knowledge base; and to *disseminate* the managed CEs to interested context consumers. The context consumer will *use* the CEs for different purposes according to their needs.

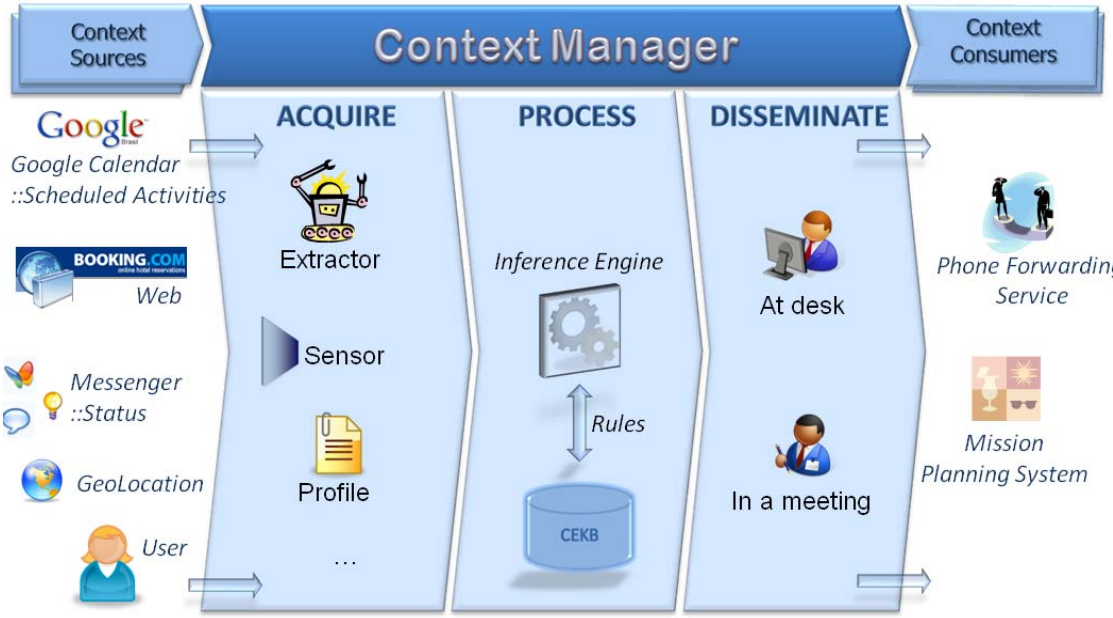

Figure 4-2 Conceptual Elements in a CSS Architecture and an Interaction Example

For example, as illustrated in Figure 4-2, a context consumer *Phone Forwarding Service* expects to receive information about a user's *current location*, *availability* and *scheduled activities* in order to *redirect phone calls* more appropriately. The *current location* is acquired using the context source *GeoLocation*, the user's availability is inferred from information acquired from the context source *Messenger*[3], and the user's scheduled activities is obtained by querying a third context source *Google Calendar*[4].

The context manager acquires the information from these heterogeneous sources, processes them and disseminates them to the interested context consumer (the *Phone Forwarding Service*). The consumer will use the information to guide its behavior. For example, by analyzing the obtained

[3] http://www.windowslive.com/messenger/overview.html
[4] http://www.google.com/calendar/

information, the service may observe that the user is identified as "absent" and that s/he has a meeting scheduled at the current time. The consumer may, then, infer that the user "is busy in a meeting" and thus the phone calls should be redirected to the user's mobile phone (if it is an urgent call) or to her/his answering machine (otherwise).

4.3 Dealing with Context Dynamics

Since context is a subtle and complex concept, it is necessary to reduce the scope of what will be considered by the context manager, delimiting a knowledge domain. As illustrated in Figure 4-3, the dynamics associated to manipulating context comprises four requirements [Vieira et al., 2007b]:

1) *CK Construction*: refers to the representation of the CEs in a domain and the acquisition of CEs from different context sources;
2) *PC Building*: indicates the identification, given a focus, of the set of CEs relevant to that focus;
3) *Behavior Triggering*: entails the binding of the identified CEs to trigger appropriate behaviors; and
4) *Incremental Knowledge Acquisition*: is related to enhancing the existing knowledge by learning new rules and patterns.

The information flow (from EK to CK, from CK to PC) follows the process of context evolution according to [Brézillon and Pomerol, 1999], as described in Section 2.1.3. These four requirements guide the principles of CEManTIKA and are embedded in the proposed architecture, metamodel and process. Next sections discuss them in more details.

4.3.1 CK Construction

Since context is what enables the characterization of entities and entities exist within a knowledge domain, context is also strongly influenced by the domain it is applied in. It means that when including and managing CEs for an application domain one must first identify the set of CEs that characterize the relevant entities in that domain.

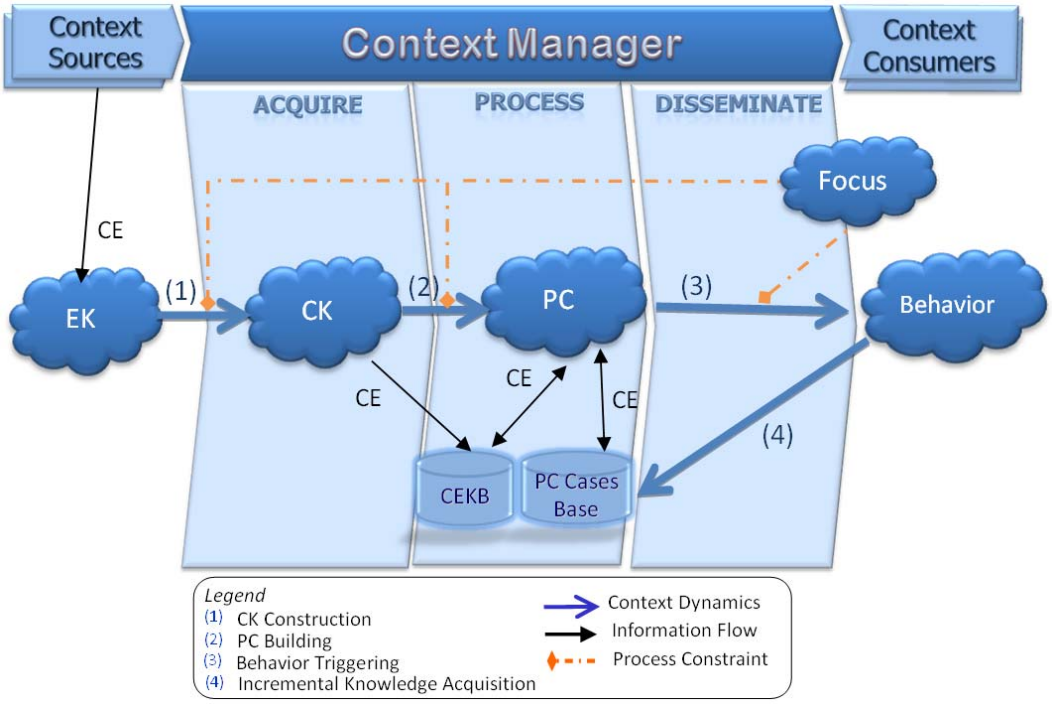

Figure 4-3 Illustration of Issues Associated to Context Dynamics

The first concern of a context manager is to identify, define and store as much CEs as possible related to the application domain. The elements may come from several and heterogeneous sources. The specified and acquired CEs are used to construct a Contextual Elements Knowledge Base (CEKB) for that domain. Different domains necessarily entail building different CEKB, since a CE can have different meanings, when analyzed in different domains.

A well designed and filled CEKB is a key factor in a CSS. It is necessary to look increasingly deeper in the domain, so that the manager can identify how a change in the context affects the state of the entities and consequently the system's actions and events. For example, considering a Global Positioning System (GPS) application that aids drivers to identify the best itinerary from one place to another. The CEs considered in this application includes the device's current location and the location of different entities. According to the current location the system is capable to show, for example, the itinerary the user should follow to arrive at a given destination. To provide the appropriate paths the system must have as much information as possible about the region where it is being used. For instance, the GPS application containing itineraries information related to the France will be useless if the user is in Brazil.

4.3.2 PC Building

To execute the task properly, the person must have previous knowledge about the task and its knowledge domain. When a person focuses in a task, only a portion of his/her body of knowledge is activated: the portion that is relevant to the task at hand. This portion must be activated and instantiated to support the person on making decisions or triggering actions. Similarly, the CSS should identify what part of the knowledge is relevant to support the task being developed.

The PC building process is related to distinguishing, given a focus, the CEs that should be considered (i.e. relevant) from those that should be ignored (i.e. not relevant). To build a PC in a focus, the manager receives the indication about the focus, identifies and extracts from the CEKB a subset of CE that should be considered as relevant to that focus. This relevance relation can be defined following stated criteria and using heuristics to process these criteria.

To support relevance identification, the context manager keeps a case repository relating historical episodes of built PC. We call this repository Proceduralized Context Cases Base (PCCB). Past occurrences of PC built for a focus, extracted from the PCCB, can support identifying a new set of relevant CEs in new occurrences of the focus. A *case* contains information such as a timestamp indicating its creation time, a reference to the focus, the CEs identified for the focus, the triggered behavior and the user's feedback (if any).

4.3.3 Behavior Triggering

The Merriam-Webster dictionary [Merriam-Webster, 2008] translates *behavior* as "the way in which something functions or operates". Each application has a set of predefined behaviors, which are triggered according to what was specified by the designers. Context is something that can affect the predicted behavior of an application. A behavior variation indicates the execution of a different set of actions according to the occurrence of a set of related conditions.

The *Behavior Triggering* process is the core of any CSS and is related to the identification of the appropriate behavior to be executed according to conditions associated to the identified CEs. This is not an easy task, since

several criteria must be taken into account. Usability issues must be considered to avoid intrusiveness, information overload, and to enable the users to control how the adaptations occur (as discussed in Section 2.2.3).

4.3.4 Incremental Knowledge Acquisition

The difficulty of managing CEs in a domain is that the interpretation of their meaning and relevance may widely change, according to different users. Hence, it is difficult and not reliable for a system designer to describe *a priori* all CEs and processing rules related to the knowledge domain based exclusively on her/his own experience. To be useful and to really attend users' expectations, a CSS must consider ways of incrementally acquire knowledge about the context process (i.e. CEs and their processing rules) during a system's usage.

The CSS should provide users with ways to indicate if the provided information was useful and if the triggered behavior was appropriate. This feedback can be used to support learning from the users' opinion and experience. Learning may occur by considering users feedback, by allowing the user to define new rules or by the analysis of historical interactions and usage (e.g. machine learning). In this sense, the incremental knowledge acquisition occurs through learning of new CEs and rules.

4.4 Context Architecture

This section presents an extensible architecture for CSS, named Context Architecture (Figure 4-4). The architectural elements follow the classification presented in Section 4.2 (*Context Sources*, *Context Manager* and *Context Consumers*). The designed functionality is inspired by the tasks classification presented in Section 4.2 and the principles for dealing with context dynamics described in Section 4.3.

The Context Manager is organized into four main modules: *Controller*, *CEAcquisition*, *CEProcessing* and *CEDissemination*. The Context Consumer includes the modules: *BehaviorTrigger* and *FeedbackHandler*. Adapters (*CSAdapter* and *CCAdapter*) enable the Context Manager to communicate with

the Context Sources and the Context Consumers. Next sections detail each architectural element, describing its internal modules.

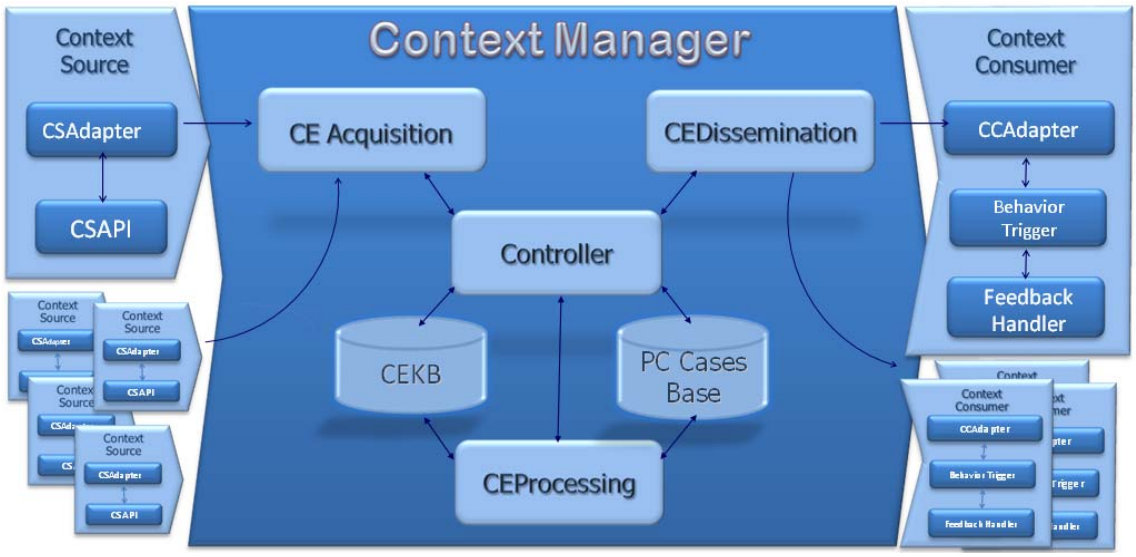

Figure 4-4 Context Architecture Overview

4.4.1 Context Source

Context sources are, by nature, heterogeneous, autonomous and dynamic. This is due to the fact that they exist independently from the context manager or the CSS and they can provide the same set of CEs to different CSS. Moreover, they may serve to different purposes other than providing CEs. These sources may be added, made unavailable or removed at any time, according either to the CSS requirements or to issues related to the context source (e.g. the CSS final user may explicitly deactivate a context source).

The CSS project may demand the creation of new context sources or it may be used existing context sources (created in other CSS projects). To allow compatibility, each context source should implement two modules: *CSAPI* (Context Source Application Programming Interface) and *CSAdapter* (Context Source Adapter). The former isolates the internal functionalities of the context source software, enabling different applications to access its information. The latter specifies the communication protocols between the context source and the context manager.

Each context source can provide specific CEs. The *CSAdapter* must implement the appropriate translation for each CE between the context source format and the foreseen format on the CSS context model. For example,

considering the CE person's location, a context source may provide this information in the format of geographical coordinates and a CSS may expect this information in the format of a tuple <country, city>. The CSAdapter for that context source must translate the location by receiving the geographical coordinate from the context source and sending the corresponding tuple <country, city> to the CSS context manager. Different CSS will, normally, demand different CSAdapters. However, different CSS may reuse the CSAPI.

4.4.2 Context Manager

This section describes the modules for the Context Manager, as explained in the following.

- *Controller Module*

The *Controller Module* is responsible to orchestrate the communication between the other Context Manager modules, by controling the activities related to context acquisition, processing, dissemination and storage in the managed repositories. Two repositories were defined for the Context Manager (according to the discussion presented in Section 4.3): Contextual Elements Knowledge Base (*CEKB*) and Proceduralized Context Cases Base (*PCCB*). The former stores the CEs managed by the system, while the latter keeps historical cases. The *CEStorage module* is responsible to handle persistence issues related to the shared repositories, by encapsulating the methods for accessing the shared repositories.

- *CEAcquisition Module*

The *CEAcquisition Module* is responsible to manage the context sources used by the CSS and to query/receive CEs to/from each context source. One of the purposes of separating the functionalities related to context management is to make the access to different and heterogeneous context sources transparent to different context consumers, allowing the reuse of context acquisition solutions. This is valuable, since the information acquired from a source may be used by different processes in a CSS.

The functionalities associated to the CEAcquisition module includes:

- To register a new context source;
- To activate or to deactivate a context source;
- To manage the communication with the context source according to the update frequency defined for each CE acquired from that source.

Context acquisition may occur in two directions: from the sources to the manager and from the manager to the sources. The first happens when a context source intentionally communicates a CE value to the manager, which occurs, for instance, when a new source enters the system or when the CE value is updated in the source. The second direction takes place when the manager needs an updated value of a required CE. In this case, the CE Acquisition module should query the needed value in the corresponding context source.

- *CEProcessing Module*

The principle of knowledge-based systems is to combine a knowledge base (KB) with an inference mechanism [Russell and Norvig, 2003]. The KB stores sentences about the world, while the inference mechanism enables the system to infer new sentences, which will be used to decide what action to take. The *CEProcessing Module* uses the CEKB to assist the processing of known CEs and the identification of new CEs. Since different engines can be used to process the CEKB, this module abstracts the interaction between the context manager and external inference engines. It is also responsible for the tasks related to context dynamics, i.e., to identify, given a focus, what CEs stored in the CEKB are relevant to that focus.

Relevance heuristics and algorithms should be considered to identify the relevant CEs according to a focus. Distinct factors can influence the relevance of a CE, as for example:

- *Relevance weight*, attributed to the relation between the CE and the focus, indicating how relevant the CE is to that focus;
- Context source *reliability*, which implies that the more reliable the source, more relevant is the CE;
- CE *acquisition mode*, which indicates, for instance, that a CE directly informed by a user is more relevant than another captured by a sensor;

- CE *age*, suggesting that the older a CE value the less relevant it is.

It is especially important to consider relevance in association to a fundamental element: the CSS final user. Different users may have different conceptions about the relevance of a CE related to the task s/he is executing.

- *CEDissemination Module*

This module entails the tasks related to the communication between the context manager and context consumers in order to deliver required CEs. The context consumer can ask the context manager for: current value of specific CEs, current value of all CEs related to a given entity, or the CEs relevant to an informed focus. The CEDissemination module is responsible to register interested consumers, to enable the identification of CEs those consumers have interest in, and to notify those consumers when a change occurs in the value of the required CE.

4.4.3 Context Consumer

Context consumers are software elements that change their behavior according to conditions related to the context. A CSS may include different context consumers, each one responsible to manage a specific focus defined in the system. The *Context Consumer* element in the Context Architecture is composed by three modules: CCAdapter, BehaviorTrigger and FeedbackHandler.

The *CCAdapter* specifies the communication protocols between the context consumer and the context manager, isolating the internal functionalities of the context consumer. It may effectuate, when necessary, appropriate translation for a CE value between the context manager format and the format used in the context consumer.

The identified CEs will be used by the *BehaviorTrigger* module. This module contains references to all possible behaviors defined in the focus, and the conditions on which each behavior variation should be executed. It is responsible to identify the appropriate behavior to execute, according to the values identified for the CEs.

The *FeedbackHandler* module has two main concerns: (i) to exhibit an explanation indicating why a given behavior was triggered; and (ii) to ask the user to inform how useful and near to their needs was the actions taken by the CSS. This feedback could allow incremental knowledge acquisition (as described in Section 4.3). It will support identifying a case episode related to the context usage to support further interactions between that user and the CSS.

4.5 Concluding Remarks

This chapter presented an overview of the general ideas and concepts underlying CEManTIKA, a framework for supporting the design of CSS. It also described the CEManTIKA proposal for a generic and extensible architecture for CSS. This architecture separates the tasks related to context management and usage into independent and integrated modules. This division in modules intends to improve reusability and extensibility, by loosely coupling the elements and clearly separating concerns. This approach brings four additional advantages:

- *Reusability*: the solution for each task can be developed in a generic way, separating the context manipulation functionalities from the CSS's business features, and the modules can be reused by different CSS;
- *Context source independence*: the CSS can be developed independently from the necessary context sources;
- *Sharing*: different CSS can share CEs acquired from distinct and heterogeneous context sources.

To support the ideas discussed in this chapter, it is necessary to represent the information managed by the CSS in a structured format. The concepts related to context management should be formalized in order to be manipulated by the manager. To this end we propose a *Context Metamodel*, described in the next chapter.

CHAPTER

5

A Domain-Independent Context Metamodel

As discussed in Chapter 3, context models, in general, aim to specify concepts that could be used to identify and describe situations in a domain. They enumerate all domain concepts that can be eventually considered as context in that domain (e.g. user's context, device's context, or location's context). No specific formalism neither their relationship to the context dynamics is considered when structuring these elements. Moreover, much of the information modeled as "context" in these models is actually a redesign of information specified by domain specialists or system's analysts. Additionally, these models do not associate the contextual information to its usage. As part of our framework to support CSS design (presented in Chapter 4), we propose a context metamodel that abstracts the concepts related to the context manipulation. The metamodel is independent of specific application domains and intends to support the creation of context models.

This chapter is organized as follows: Section 5.1 presents a scenario of example that will be used to illustrate the metamodel description; Section 5.2 gives an overview of the metamodel discussing some design decisions; Section 5.3 presents the concepts related to the structural part of a CSS; Section 5.4 discusses the concepts related to a CSS behavioral part; Section 5.5 presents the

proposed UML Profiles related to the structure and behavior concepts; and Section 5.6 presents the concluding remarks for this chapter.

5.1 Example Scenario

In order to illustrate the context metamodel presentation, this section introduces a context-sensitive system as a scenario of use.

Consider a system that supports researchers on planning their academic missions. An academic mission is any scientific or academic event attended by researchers, professors or students (e.g. conference, stage, lecture or meeting). The person who is attending to a mission is called a 'missionary'. Missions may have particular characteristics (e.g. duration, location, tasks to be accomplished). Distinct missionaries may execute different steps and fulfil specific requirements when planning a mission. For example, in a university, the steps to be executed and the available resources (e.g. finantial support) to a professor are substantially different from those available to a student.

To accomplish a mission there are common tasks to be performed, such as: to register a mission participation demand (start a new mission), to request financial aid, to book a hotel for stay during the mission (in cases where the mission occurs in a city different from the missionary's residence), to book and buy transportation tickets to arrive at the mission location (when the city is not the missionary's residence), and to accomplish the steps to finalize the mission. These main requirements are summarized in the Use Cases Diagram for the Academic Mission Support System (Figure 5-1). These features can be performed by a professor or a student. The *Book Hotel* feature can be provided by an external system, called *HotelBookingService*.

The features that can be influenced by the context include: to identify the steps to be accomplished and requirements to be filled according to the mission type and the missionary profile; to support the missionary on booking transport and accommodation; and to identify other concerns related to the mission that could be useful to the missionary (e.g. academic activities, money exchange or tips for touristic plans). Personal history of the missionary in similar missions could also be useful (to identify preferences or previous decisions).

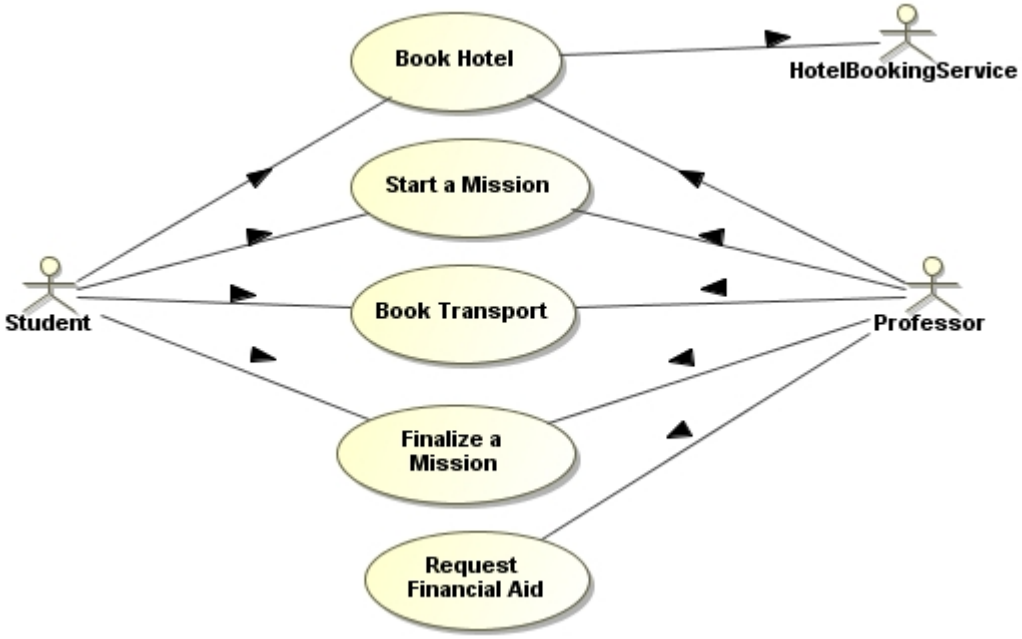

Figure 5-1 UML Use Cases Diagram for the Academic Mission Support System

Figure 5-2 presents the Conceptual Class Diagram for the Academic Mission Support System. *Person* is a generalization for *Professor* and *Student*. A *Person* has the attributes *age*, *name*, *eyeColor*, *availability* and *sex*. A *Student* also has the attribute *academicDegree*. *Student* has an *advisor* relationship with *Professor* indicating that a student can be advised by one or two professors, and that a professor can advise multiple students. *Person* has the following relationships: *participates* in zero or more *Missions*; *isClient* of zero or more *Hotels*, have the identification of one *living Location* and one *current Location*. A *Mission* has the attributes *whoPays*, *duration, startDate, endDate* and *type*, and the relationship *location* indicating in what location the mission takes place. A *Hotel* has the attributes: *category*, *isCheap* and *distanceToTown*, the relationship with Person indicating the clients it has (*hasClient*), and a relationship with *Location* indicating where the *Hotel* is located. A *Location* has the attributes *countryName* and *cityName*.

In this scenario we introduce three people, named Mary, Lucy and Tom. Mary is a professor at a Computer Science department of a University located in Recife, Brazil. Tom is a professor at a Computer Science laboratory of a University located in Paris, France. Lucy is a PhD student, supervised by Mary,

who is currently in an academic mission performing a stage at Tom's laboratory. Their interactions with the system will be explained when needed.

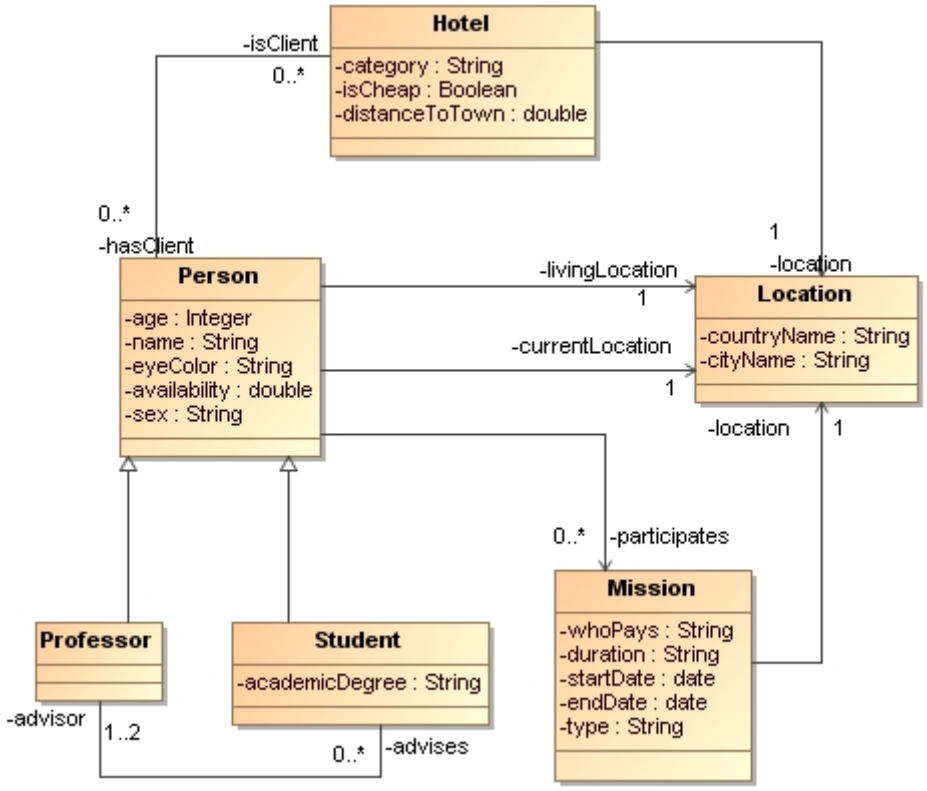

Figure 5-2 Conceptual Class Model for the Academic Mission Scenario

5.2 Context Metamodel Overview

The Context metamodel defines the semantics for the main concepts that should be used to build context models. It provides new modeling elements related to context. Such a metamodel should abstract and specify the concepts related to context and its manipulation, providing a conceptual infrastructure to support building context models. By specifying a metamodel we can support system developers in the context specification phase, since they will have a basis on which to structure their model.

Since context is a novel and not well understood area, a challenge in context metamodeling is to identify and specify what concepts are related to the context manipulation, how they relate to each other, what their semantics are and how to formalize them.

5.2.1 Objectives and Design Principles

The following goals were defined for the Context Metamodel:

- To provide a conceptual framework that identifes the main concepts related to context manipulation and usage, in a domain independent way;
- To support the reuse and extension of existing context models or application models.

In order to fulfil these objectives we chose to build the metamodel as an extension of the elements provided by the UML 2.0 Metamodel [OMG, 2007a], following the UML standard semantics and notation. The UML specification is also used to support the graphical representation of the metamodel concepts, through its Class Diagram notation.

5.2.2 Context Metamodel in the Four-Layer Architecture

Modeling and metamodeling are similar activities – the difference being one of interpretation. A model is an abstract representation of a real-world system or process. A four-layer metamodeling architecture was established in [OMG, 1997] and is commonly used by the metamodeling community. Table 5-1 situates the Context Metamodel in that architecture (layer M2).

Table 5-1 Context Metamodel in the Four-Layer Metamodeling Architecture

Layer	Example
(M3) Meta-metamodel	MOF model (Metaclass)
(M2) Metamodel	UML Metamodel (Class, Attribute) **Context Metamodel**
(M1) Model	Hotel Model (Hotel, name, Hotel has name)
(M0) Data objects	Hotel Database (<h1>,<“Ritz”>, <h1, “Ritz”>)

The *meta-metamodel* layer (M3) defines a language for describing metamodels. MOF (Meta Object Facility) [OMG, 2006a] is a standard language that specifies constructs for creating metamodels (e.g. *Metaclass*). A *metamodel* (layer M2) is an instance of a meta-metamodel and it defines a language for specifying models. For example, the UML Metamodel [OMG, 2007a] defines constructs that enable to describe models in that language (e.g. *Class* and

Attribute). A *model* (layer M1) is an instance of a metamodel that specifies a language for describing an information domain. For example, the conceptual model related to a *hotel booking* scenario may define a *Class* called "Hotel", an *Attribute* called "name" and an association indicating that a *Hotel* "has" a *name*. *Data objects* (layer M0) are instances that conform to the model and define a specific information domain. In the *hotel booking* example, an instance of the *Class Hotel* may be the *object h1*, an instance of the Attribute name is the *string "Ritz"*, and the *tuple <h1, "Ritz"*> indicates that h1 has the name "Ritz".

Existing context modeling approaches are generally related to the model layer (M1) and the context models are constructed as extensions of existing languages (e.g. OWL, UML). Since these languages are conceived for general purposes, they do not offer the appropriate support and abstraction for building context models. The Context Metamodel defines the context-related concepts in a high level domain-independent layer. It aims to guide developers to create their context models.

5.2.3 Metamodel Organization

The Context Metamodel is divided into two main packages that organize the concepts in two categories (Figure 5-3):

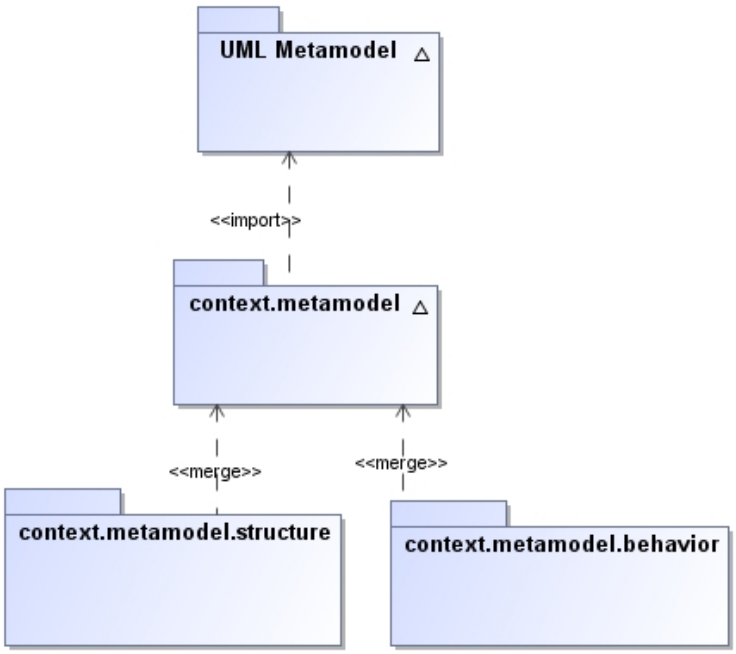

Figure 5-3 Context Metamodel Packages Organization

- `context.metamodel.structure`, which describes the concepts related to the conceptual and structural elements of a CSS (context conceptual model);
- `context.metamodel.behavior`, which contains the concepts related to the behavioral aspects of a CSS (context behavior model).

The `UML Metamodel` package offers the constructs available in the UML 2.0 specification [OMG, 2007a] used to compose the Context Metamodel. Next sections describe the metamodel concepts according to this package separation: structure concepts (Section 5.3) and behavior concepts (Section 5.4).

5.3 Context Metamodel Structure Concepts

The main concepts in a context model related to the CSS structure (Figure 5-4) are: `ContextualEntity`, `ContextualElement`, `ContextSource`, `Focus` and `Rule`. Other supporting concepts are: `Task`, `Agent`, `role`, `relevance` and `acquisition`. Some datatypes were specified to support the model description: `AcquisitionType`, `UpdateType`, `RelevanceType` and `ContextType`. Figure 5-4 also presents the relation between the main concepts and the metaclasses they extend from the UML Metamodel. Next sections describe these concepts.

5.3.1 ContextualEntity

Conceptual modeling, as defined in [Mylopoulos, 1992], is the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication. A conceptual model, generally, identifies the entities that are relevant to define the world being modeled, the relationship among those entities and abstract mechanisms to classify and organize those entities. An entity represents a concrete representation of a real world object that can be distinctly identified and that is relevant to describe a domain. An entity is used to classify sets of individuals with similar

characteristics and it contains descriptions of these individuals through encapsulated attributes. The entities in the conceptual model of the Academic Mission scenario (Figure 5-2), are: *Person*, *Professor*, *Student*, *Mission*, *Hotel* and *Location*.

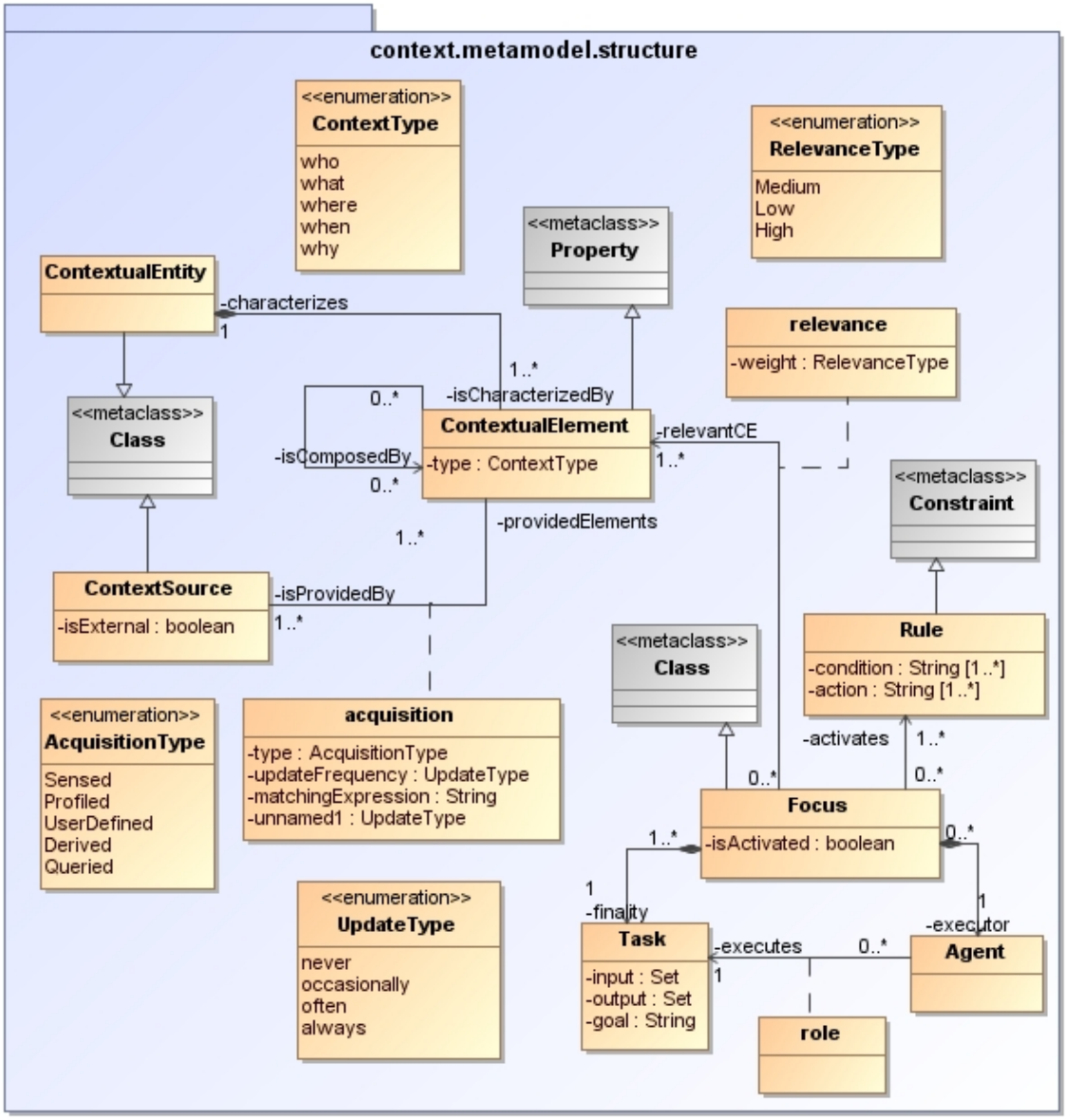

Figure 5-4 Context Metamodel Structure Concepts

In a context model, a `ContextualEntity` represents the entities that should be considered for context manipulation purposes. These entities can be identified from the application conceptual model. A `ContextualEntity` is characterized by at least one `ContextualElement`, which is explained in the following.

5.3.2 ContextualElement

Properties are binary relations that link two individuals (or one individual with itself) or an individual to a data value. The former type of property is commonly called relationship and the latter is known as attribute. A `ContextualElement` (`CE`) represents a property used to characterize a `ContextualEntity`. A `CE` can be identified from the set of attributes and relationships associated to an entity. Examples of `CEs` in the Academic Mission scenario (illustrated in Figure 5-2) are:

- *age*, *sex*, *academicDegree*, *advisor*, *livingLocation* (characterizes the `ContextualEntity` *Student*);
- *location*, *duration*, *type* (characterizes the `ContextualEntity` *Mission*);
- *location*, *category*, *isCheap*, *distanceToTown* (characterizes the `ContextualEntity` *Hotel*).

`CE` is the basic unit of information in the Context Metamodel. A context will be composed by an aggregation of `CEs`. A `CE` should always be associated to a `ContextualEntity`. However, not necessarily all properties of a `ContextualEntity` is classified as `CE`. The criterion to identify if a property is a `CE` is subjective and strongly dependent on the context requirements defined for the CSS.

A `CE` may be derived from one or more `CEs`. For example, the `CE` *Mission.duration* may be defined from two other `CEs`: *Mission.startDate* and *Mission.endDate* using a transformation function. This relation between `CEs` is indicated by the association `composes`.

Some authors argue the need to categorize the CEs according to the type of information it provides in order to ease its identification and usage (e.g. [Bulcão Neto and Pimentel, 2005, Jang et al., 2005, Truong et al., 2001]). The 5W classification indicates whether a CE is related to one of the following questions: `who` (identity), `what` (activity), `when` (time), `where` (location) and `why` (motivation). In the Context Metamodel (Figure 5-4), this classification composes the concept `ContextType`. The attribute `type` enables to inform

this classification for a CE. For example, the CE *Person.livingLocation* is of type `where` and the CE *Mission.duration* is of type `when`.

5.3.3 Focus

Focus is what enables to determine what CEs should be instantiated and used to compose the context. We adopt in this work the definition of focus proposed by Brézillon and Pomerol [Brézillon and Pomerol, 1999] that considers it to be "a step in a task execution, in a problem solving, or in a decision making". As stated by this definition, a focus has a strong relation with the *task* being executed. We extend this interpretation by emphasizing that a focus is determined by the task together with who is executing it. The task executor is an *agent*, which can be a person, a group of people, a process or a software agent. An agent may perform different *roles* when executing the task.

Therefore, in this work a `Focus` (as represented in Figure 5-4) is defined as a composition of a `Task` and an `Agent`, where the Agent executes a Task performing a role. For example, from the requirements defined in the Academic Mission Support System (Figure 5-1) we can identify that an agent *Professor* may use the CSS to execute the task *Request Financial Aid*, or an agent *Student* may use the CSS to execute the task *Book a Hotel*. Each tuple `<agent; task>` constitutes a distinct Focus in the CSS context model.

5.3.4 CE Relevance to a Focus

An important issue in a context model is to identify the association between a focus and the CEs that are relevant to support it. By our definition, context is a dynamic concept that should be rebuilt at each new focus, being composed by all CEs that are relevant to support the task defined in the focus. The relevance level of this association is affected by the agent who is performing the task. For example, when executing the task *book a hotel*, an agent *Student* can indicate that the hotel's price must be considered with a higher relevance than the hotel's comfort. We may consider, on the other hand, that an agent *Professor* could prefer a more comfortable hotel even if it is not the cheapest one. In this sense, the CEs *Hotel.isCheap* and *Hotel.category* must be associated as relevant

to the foci <*Student*; *book a hotel*> and <*Professor*; *book a hotel*>. However, these associations should have different relevance weights.

In the Context Metamodel (Figure 5-4) the CE relevance to a Focus is indicated through the concept `relevance`, an association between a `Focus` and a `ContextualElement`. A relevance `weight` indicates how relevant the CE is to the focus and can assume one of the following values: `High`, `Medium` or `Low` (defined in the `RelevanceType` datatype). These relevance weight values can be used, for instance, to compose heuristic functions where different CEs can be combined. In the *book a hotel* task example, an agent *Student* may indicate that s/he prefers hotels that combine an intermediate relation between price and comfort (i.e. relevance `weight` for the CE *Hotel.price* = *Medium* and for the CE *Hotel.category* = *Medium*).

5.3.5 ContextSource and Acquisition association

One characteristic of CSS is that the values of a CE may originate from heterogenous and external context sources (e.g. user dialog interfaces, profiles, physical sensors, desktop sensors and external databases). For example, a person's location may be provided by a GPS device (for outdoor places), a badge identification service (for indoor places) or still an IP (Internet Protocol) locator service (for network connections). A context model should provide ways to inform how the CE acquisition occurs.

In the Context Metamodel this can be done through the concept `ContextSource` and the association `acquisition` between a `ContextSource` and a `ContextualElement`. The attribute `isExternal` in the `ContextSource` indicates whether the context source is external to the CSS (i.e. implemented by other applications) or an internal element of the CSS. For example, in the Academic Mission Support System, we consider three context sources: an internal *User Profile*, an internal *Mission Form* and an external *IP Location Service*.

The `acquisition` association concept indicates and parameterizes the relationship between a `ContextualElement` and a `ContextSource`. For example, in the Academic Mission Support System (Figure 5-2): the CE

Person.currentLocation is provided by the context source *IP Location Service;* the CEs *Person.age* and *Person.sex* are provided by the context source *User Profile*; and the CEs *Mission.startDate*, *Mission.endDate* and *Mission.type* are provided by the context source *Mission Form*. The way each CE is acquired may vary, according to the CE and context source characteristics. The `acquisition` association uses three attributes to configure how the acquisition occurs: `type`, `updateFrequency` and `matchingExpression`:

- `type`: classifies the CE according to the manner it is acquired. It expects a value of type `AcquisitionType`. Based on classifications found in previous works ([Simons and Wirtz, 2007, Henricksen, 2003]), the datatype `AcquisitionType` accepts the values: `Sensed`, `Profiled`, `UserDefined`, `Queried` and `Derived` (explained in Table 5-2).

Table 5-2 Values for the `AcquisitionType`

Type	Description
`Sensed`	Provided by a physical or virtual sensor. A Sensed CE, in general, changes frequently and is prone to be incorrect, unkown or aged, due to sensors' failures or network disconnections.
`Profiled`	Extracted from an existing profile (e.g. person profile or device profile). A Profiled CE may change but not so frequently as a Sensed CE, and depends on the user to keep the value up to date.
`UserDefined`	Directly informed by the agent, at run-time, on demand (e.g. through a dialog interface). A UserDefined CE is often related to CEs that cannot be inferred automatically (e.g. a person's mood, a person's motivation for making a decision) and it should be asked every time it is needed.
`Queried`	Extracted from repositories external to the CSS. A Queried CE, in general, changes but not as frequently as a Sensed CE, and it tends to be reliable, since it is normally related to transactional data used by an organization.
`Derived`	Inferred from other CEs through transformation functions or inference rules. A Derived CE must be very reliable (e.g. a person's age, derived from her birth date) or prone to imperfections (e.g. a person's availability, derived from her status informed in a messenger service).

- `updateFrequency:` indicates the periodicity for the validity of a CE value. This attribute supports the CSS to decide if it can rely on the last value assigned for the CE or if it should ask the context source for an updated value. It accepts a value of type `UpdateType` (described

in Table 5-3) that accepts the following values: `never`, `occasionally`, `often` and `always`;

- `matchingExpression:` indicates a transformation function or derivation rule used to convert a value received from a context source into a value compatible with the CE described in the context model. For example, in the Mission scenario (Figure 5-2), a CE *Person.availability* expects a value of type `double` (higher the value more available the person is), and a context source provides a `string` with possible values "Busy", "Away", and "Available". A transformation function should be created to convert the `string` value into a corresponding `double` value;

Table 5-3 Values for the `UpdateType`

Type	Description
`Never`	The CE value is stable and never changes (e.g. a person's birth date).
`Occasionally`	The CE value may change, but not regularly (e.g. a person's age).
`Often`	The CE value is dynamic and changes regularly, demanding frequent communications between the context source and the CSS to keep it up to date (e.g. a person's current location).
`Always`	The CE value is volatile, changes constantly and is out of date right after its acquisition (e.g. the current time).

5.3.6 Rule

When processing a CE or identifying the behavior of a CSS, it may be necessary to consider associated rules. A well known and widely used type of rule is the so called production rules. A production rule is a statement of programming logic that specifies the execution of one or more actions in the case that its defined conditions are satisfied [Russell and Norvig, 2003].

In the Context Metamodel (Figure 5-4) a `Rule` is represented as a set of one or more `conditions` and a set of one or more `actions`. Each condition represents an expression, which results in a value *true*, *false*, or *null* (*unknown*) when matched to available data. An action indicates a procedure that must be executed when the rule's conditions are satisfied. In a CSS, the type of the actions may include: to *trigger* a system's behavior; to *assign* a CE value as a

result of a composition from other CEs; or to *assign* a new relevance weight for an association between a CE and a Focus.

For example, in the Academic Mission scenario a rule related to the Focus <*Professor*; *Book a Hotel*> is described as:

```
Rule1:
   Conditions
      Professor.age >= 30;
      Professor.age < 60;
      Mission.duration = “long”;
   Actions
      setPriceWeight(0.5);
      setCategoryWeight(0.8);
```

This rule indicates that mature professors involved in long duration missions consider with a high relevance weight the hotel’s comfort, and with a medium relevance weight the hotel’s price, when performing a task *book a hotel*.

5.4 Context Metamodel Behavior Concepts

The package `context.metamodel.behavior` defines the concepts related to the CSS context behavioral model, which indicates how the CSS behavior is affected by the context. To support the design of the CSS behavioral part, we use the concepts defined in the formalism of Contextual Graphs [Brézillon et al., 2002]: *contextual node*, *action*, *recombination node*, *activity* and *parallel action grouping*. The semantics of each concept is the same stablished in the formalism, as explained in Section 3.2. The restrictions stablished for contextual graphs are also considered (e.g. each contextual graph must have one initial node and one final node; each contextual node must have a corresponding recombination node, and so on).

Figure 5-5 shows a graphical representation of the contextual graphs concepts and restrictions, using the UML Class Diagram, as part of the package `context.metamodel.behavior`. In this representation we made some changes in the name of the concepts (e.g. ActionNode instead of Action, and ActivityNode instead of Activity), in order to avoid conflicts with the concepts defined in the UML Metamodel. We also included some concepts that appear

implicitly in the contextual graphs definition (e.g. InputBranch and OutputBranch)

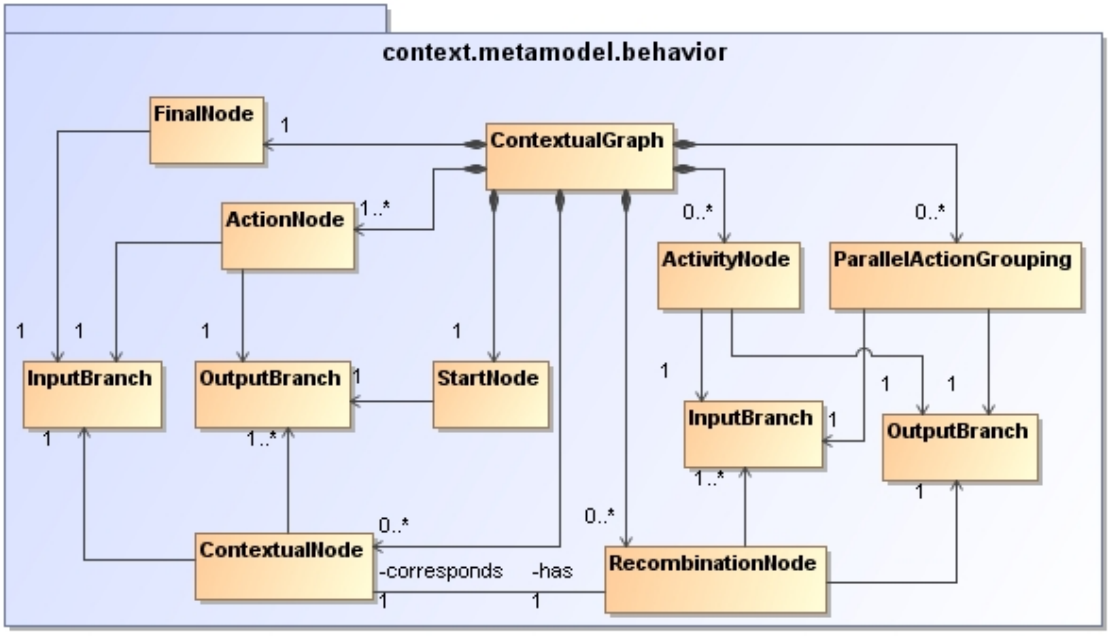

Figure 5-5 Context Metamodel Behavior Concepts

5.5 UML Profiles for Context Modeling

As explained in the Appendix B, the *UML profile mechanism* enables the customization of the UML Metamodel for a specific problem domain. This is achieved by extending existing metaclasses in the UML Metamodel using three extension constructs: *stereotypes*, *tag definitions* and *constraints*. We defined two profiles for the Context Metamodel according to the concepts defined in the packages `structure` (*Context Profile*) and `behavior` (*CxG Profile*). These profiles are explained in the following sections.

5.5.1 Context Profile

Figure 5-6 illustrates the stereotypes and tag definitions defined for the Context Profile. To exemplify the stereotypes descriptions, we extended the previously defined diagrams for the Academic Mission Support System (shown in Figure 5-1 and Figure 5-2). The semantically enriched models are illustrated in Figure 5-7 (Use Cases Diagram) and Figure 5-8 (Conceptual Class Diagram). The Conceptual Class Diagram presented in Figure 5-8 is an excerpt of the diagram

shown in Figure 5-2. For the sake of clearness we brought only the elements considered relevant to explain the Context Profile stereotypes. New elements were included in the diagram of Figure 5-8 to explain the usage of the stereotypes related to the concepts `Focus` and `ContextSource`.

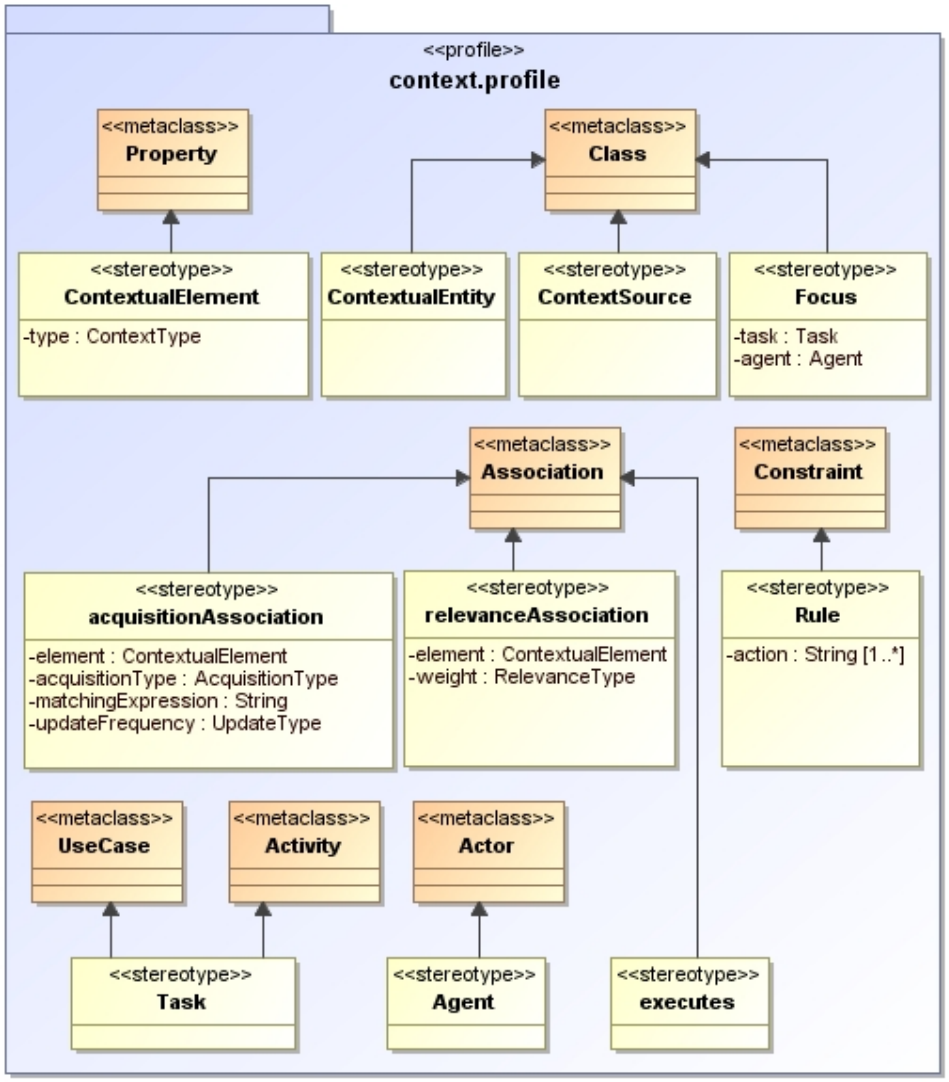

Figure 5-6 Context Profile Stereotypes and Tag Definitions

The stereotypes defined to enrich the UML Use Cases Model are:

- `<<Agent>>`: extends the metaclass `Actor` to indicate that this actor should be considered to compose a `Focus`. In Figure 5-7, *Professor* and *Student* are identified as <<`Agent>>`;
- `<<Task>>`: extends the metaclasses `UseCase` and `Activity`. In the Use Case Model, it indicates the use cases that should be considered in

a Focus composition. In Figure 5-7, the uses cases *Book Hotel*, *Start a Mission*, *Book Transport*, *Finalize a Mission* and *Request Financial Aid*, are identified as <<Task>>;

- <<executes>>: extends the metaclass Association to explicitly represent an association between an Agent and a Task. This stereotype should be used to link *actors* assigned as <<Agent>> to *use cases* denoted as <<Task>>.

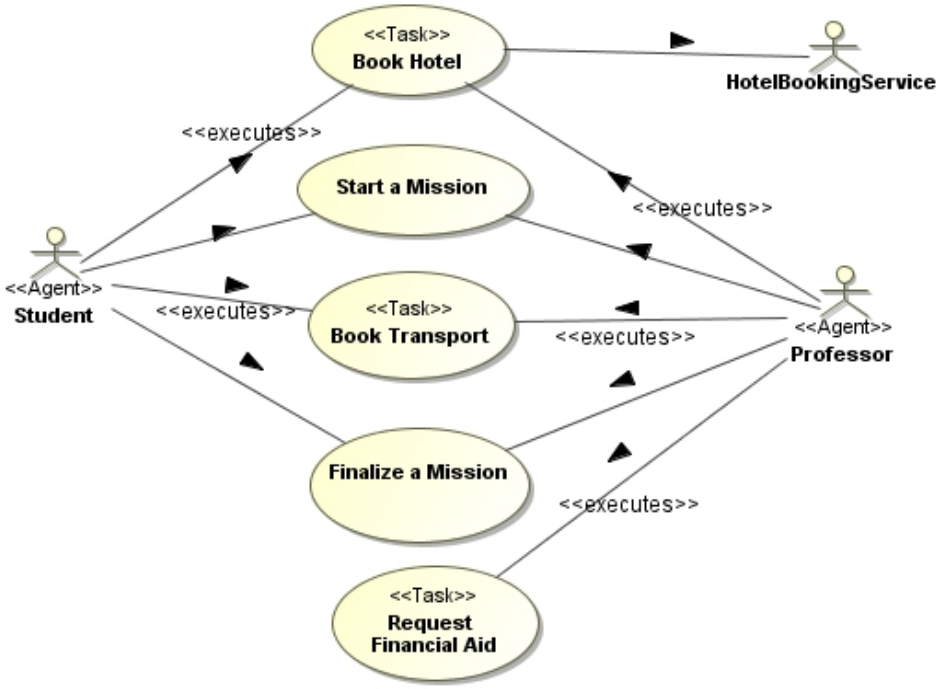

Figure 5-7 Academic Mission Use Cases Diagram, Enriched with the Context Profile Stereotypes

The Context Profile enables to enrich with context semantics the UML Class Model by the following stereotypes:

- <<ContextualEntity>>: extends the metaclass Class, and indicates which classes from the application conceptual model should be considered as contextual entities. In Figure 5-8, two contextual entities were defined: *Person* and *Mission*;
- <<ContextualElement>>: extends the metaclass Property, indicating which attributes or relationships of a contextual entity represent contextual elements. This stereotype has a tag definition type, to indicate the CE category (ContextType). In Figure 5-8,

the contextual entity *Person* has the following CEs: *age*, *eyeColor*, *availability*, *livesIn* and *currentLocation*. The contextual entity *Mission* has the CEs *whoPays* and *occursIn*. The CE *Mission.whoPays* is classified as `type=who` while the CEs *Person.livesIn* and *Mission.occursIn* are classified as `type=where`;

- `<<ContextSource>>`: extends the metaclass `Class`, and represents classes in the context model that designate context sources. In Figure 5-8, two context sources were defined: *MSNAdapter* and *GeoLocationAdapter*;

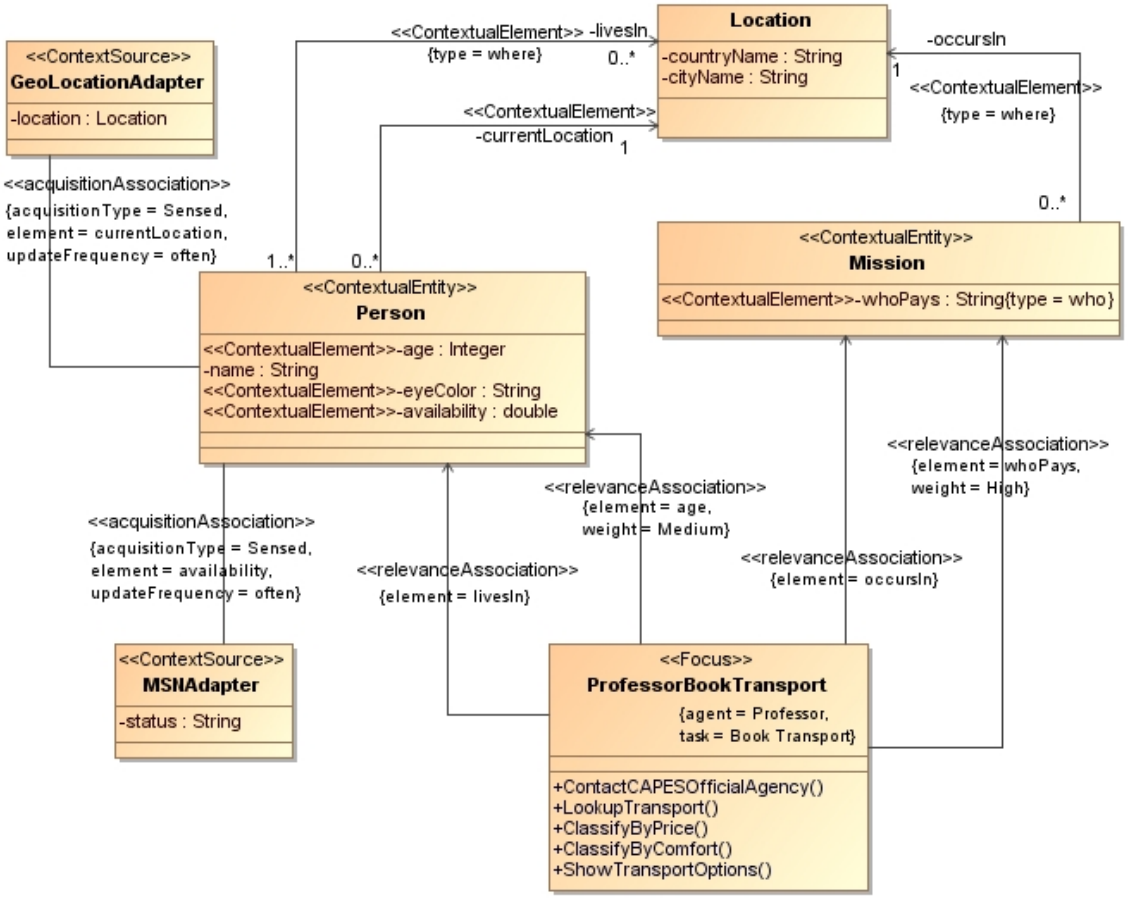

Figure 5-8 Excerpt of the Academic Mission Conceptual Class Diagram Enriched with the Context Profile Stereotypes

- `<<acquisitionAssociation>>`: extends the metaclass `Association` to model the relationship between a `CE` and a `ContextSource`. This stereotype contains four tag definitions: `element`, `acquisitionType`, `matchingExpression` and

updateFrequency, with the same semantics defined in the Context Metamodel (Section 5.3.5). The tag definition element was necessary to overcome a limitation of the UML Metamodel that does not allow associations between a class (ContextSource) and a property (CE). Therefore, the ContextSource must be associated to a ContextualEntity and the corresponding CE is mapped in the element tag. For example, in Figure 5-8 to indicate that the CE *Person.availability* is acquired from the ContextSource *MSNAdapter* we created an acquisitionAssociation between this source and the ContextualEntity *Person* and we assigned the tag element = availability;

- <<Focus>>: extends the metaclass Class and contains two tag definitions: task and agent. In Figure 5-8, the Focus *ProfessorBookTransport* represents that a task *Book Transport* is executed by an agent *Professor*;
- <<relevanceAssociation>>: extends the metaclass Association. It enables to model the relevance relationship between a Focus and a CE. It has two tag definitions: element and weight. The tag element indicates a reference to the corresponding CE. The tag weight indicates how relevant the CE is to the Focus, following the classification defined in the RelevanceType. In Figure 5-8 the CEs identified as relevant to the Focus *ProfessorBookTransport* were: *Mission.occursIn*, *Mission.whoPays*, *Person.livesIn* and *Person.age*;
- <<Rule>>: extends the metaclass Constraint to indicate rules defined for the context model. Since a Constraint already has the definition of conditions, the Rule is extended with a tag definition action to indicate the action that should be taken when the conditions are satisfied.

5.5.2 CxG Profile

The CxG Profile enables a CSS designer to model the application behavior using the UML Activity Diagram with the semantics defined in the Contextual Graphs. The stereotypes defined for the CxG Profile are illustrated in Figure 5-9. An example of contextual graph built using the CxG Profile stereotypes is illustrated in Figure 5-10. This example is related to the Focus: agent *Professor* executes task *Book Transport*, in the Academic Mission Support System.

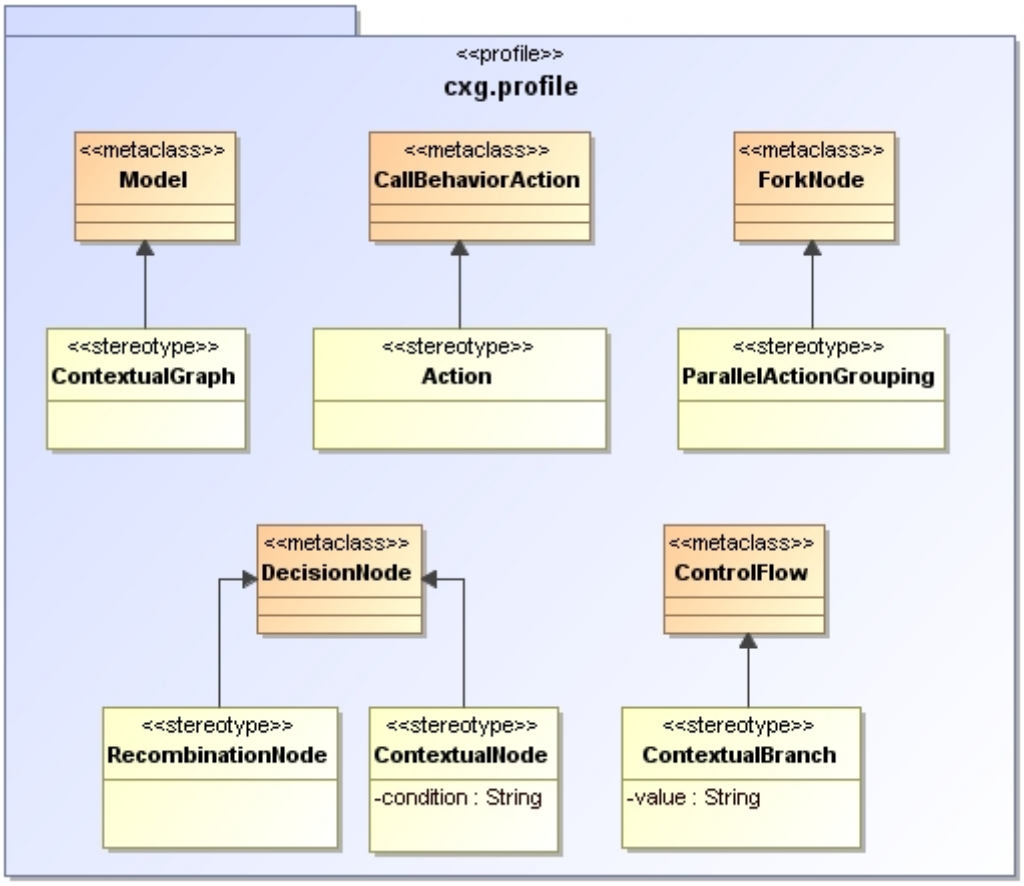

Figure 5-9 CxG Profile Stereotypes and Tag Definitions

- `<<ContextualGraph>>`: extends the metaclass `Model`, and indicates a model package containing a contextual graph elements;
- `<<Action>>`: extends the metaclass `CallBehaviorAction`, indicating a new CxG action node. For example, in Figure 5-10 an action is "*Contact CAPES Official Agency*";
- `<<ParallelActionGrouping>>`: extends the metaclass `ForkNode`, indicating actions that can be executed in parallel;

- `<<ContextualNode>>`: extends the metaclass `DecisionNode`, indicating a CxG contextual node. This stereotype has a tag definition `condition` that receives a string indicating the condition to be tested in the contextual node. For example, in Figure 5-10 a contextual node named CE2 and its `condition` is the CE "*Mission.whoPays*";

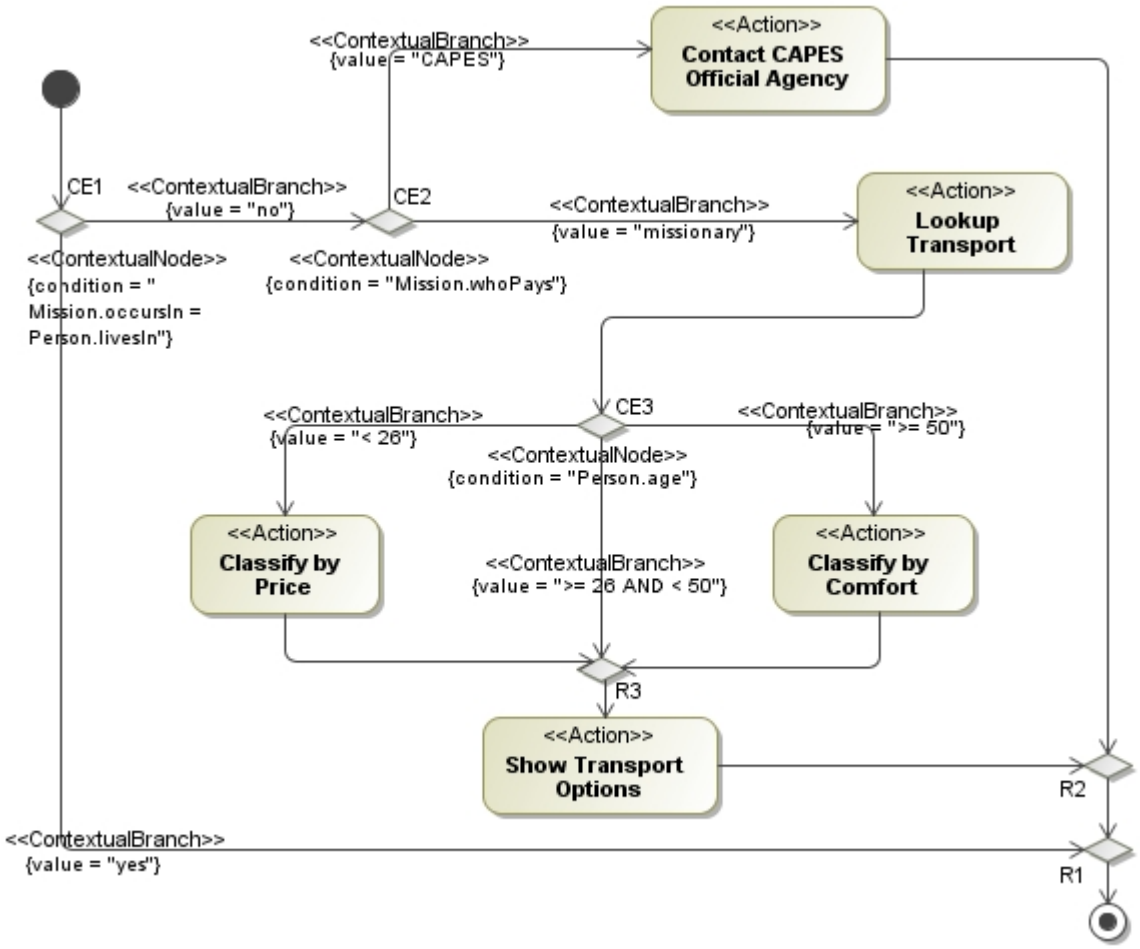

Figure 5-10 Contextual Graph for the Focus *ProfessorBookTransport*

- `<<RecombinationNode>>`: extends the metaclass `DecisionNode`, indicating the deactivation of the condition tested in the corresponding contextual node. According to the CxG definition, each contextual node should have a corresponding recombination node;
- `<<ContextualBranch>>`: extends the metaclass `ControlFlow`, and represents an association from a contextual node to: another contextual node, an action or a parallel action grouping. This branch contains a tag definition `value` that indicates the CE value that should

be validated according to the condition specified in the contextual node. For example, in Figure 5-10 the contextual node CE2 has `condition`="*Mission.whoPays*" and two contextual branches for the two possible values for this CE: `value`="CAPES" or `value`= *"missionary"*.

The advantages of using the CxG Profile to model contextual graphs are twofold: we can design a contextual graph using any UML-based tool with the advanced modeling features provided by these tools; the CSS context behavior model can be more easily integrated with the context conceptual model.

5.5.3 Using the CxG Profile to Model Behavior Variation

Considering the contextual graph illustrated in Figure 5-10, the `condition` in CE1 ("*Mission.occursIn=Missionary.livesIn"*) verifies if the mission is carried out in the same city where the missionary lives, indicating two possible contextual branches: the first tests if `value`="*yes*". In this case, no transport is necessary and thus the contextual graph points to the end of the task execution. If `value`="*no"*, it will activate another contextual node CE2.

The contextual node CE2 refers to the `condition`= *Mission.whoPays*. In this case, two branches are considered. In the first case (`value`="*CAPES*") the activated action is "*Contact CAPES Official Agency*". In the second case, when `value`= "*missionary*", another contextual node CE3 is activated.

CE3 verifies `condition`= *Person.age*. Three contextual branches are related to this contextual node. The first (`value`="*>=50*") tests if the person's age is more than 50, and in this case the <<`Action`>> "*Classify by Comfort*" will be activated, indicating that older people give preference to comfort conditions when traveling. The third branch (`value`="*<26*") triggers the <<`Action`>> "*Classify by Price*", and finally the second branch (`value`="*=26 AND < 50*"), do not execute any action and moves to the recombination node of this CE and to the next action to be executed: "*Recommend Transport*". This action execution conducts to the final node, indicating the end of the task.

The behavior variation in the contextual graph is indicated by the different flows of actions that are triggered according to conditions associated

to the contextual nodes. For example, the action *Lookup Transport Types* will be executed when the following conditions are satisfied:

```
CE1: [Mission.occursIn = Person.livesIn] = yes

CE2: [Mission.whoPays] = “missionary”
```

Each path in the contextual graph contains the rationale used to execute the task in the focus. It contains the sequence of the triggered actions, the conditions activated for each action, and the CEs related to each condition. These paths can be converted to compose the Rules in the context conceptual model (Figure 5-4). For example, considering the contextual graph of Figure 5-10, examples of rules are:

```
Rule1:
   Conditions
      not (Mission.occursIn==Person.livesIn)
      Mission.whoPays=“CAPES”
   Actions
      CallBehavior(“Contact CAPES Official Agency”)

Rule2:
   Conditions
      not (Mission.occursIn = Person.livesIn)
      Mission.whoPays = “missionary”
      Person.age < 26
   Actions
      CallBehavior(“Lookup Transport Types”)
      CallBehavior(“Classify by Price”
      CallBehavior(“Recommend Transport”)
```

5.6 Concluding Remarks

This chapter presented a Context Metamodel to guide the creation of context models in a domain-independent manner. It also presented the proposed UML Profiles [OMG, 2007a] related to the Context Metamodel: Context Profile (for structure) and CxG Profile (for behavior).

Visual languages play an important role in software engineering because graphical models are better readable and understandable by human beings. Benefits of using a context metamodel include: to achieve a common vocabulary that increases the understanding about context peculiarities; and to

provide a guide for the identification of the elements to be designed in a context model.

To exemplify the advantages of using such a metamodel, we can analyze the original conceptual model for the Academic Mission System (Figure 5-2) and the extended version of this model (Figure 5-8) enriched with the stereotypes defined in the Context Profile. We can observe that the enriched conceptual model presents much more semantics in comparison to the original model. This semantic enrichment can ease the design and evaluation of the context concepts in the CSS and the evolution and maintenance of the CSS functionalities.

The design decisions related to the specification of the metamodel concepts were made by observing weaknesses on reviewed context models (discussed in Section 3.3). In doing so, we noticed that, in general, the analysed approaches:

1) do not make a clear distinction between the concepts of Context and Contextual Element, calling everything as context. Also, they do not consider explicitly the dynamic aspect of the context in comparison with the static aspect of the contextual element;

2) do not consider a separation between the concepts of Context and Entity. There are cases (e.g. [Simons and Wirtz, 2007]) where an entity is indicated as being a context, instead of assuming that, in fact, context should be built by analysing specific properties associated to the entities, not the entity as a whole;

3) lack support for reusing existing models in the CSS context model. Most approaches propose the context model as a new model and not as an extension of existing models. There is no clear separation and differentiation between the context modeling and the application modeling parts. This work argues against this practice and claims that reusing existing models is a way to diminish the complexity in building CSS.

Another contribution of our approach is a concrete application and instantiation of the conceptual context model proposed by Brézillon and

Pomerol [Brézillon and Pomerol, 1999] through the creation of a UML Profile to support modeling Contextual Graphs. In their model they indicate the need to consider different knowledge states according to the context (external knowledge, contextual knowledge and proceduralized context) and indicate that the focus is what enables this knowledge state changing. However, they do not specify how a context model designer should effectively represent the context and the focus. In other words, they do not integrate the context model structure definition to their proposal of a context behavioral model.

Existing UML profiles for context modeling (e.g. [Ayed et al., 2007, Simons and Wirtz, 2007]) identify the concept equivalent to our Contextual Element (named, respectively, *context* and *context item*) as an extension of the `Class` metaclass. We propose to model Contextual Element as an extension of the `Property` metaclass, instead. This decision is due to our understanding that entities and contextual elements are two different but related concepts. A contextual element is used to characterize a contextual entity. So, it is not correct to consider, for example, a *Person* as a contextual element, but the *person's age*, or the *person's current location*. Semantically, this corresponds to the attributes associated to a class *Person* (e.g. *age*) or to the associations between that class with another (e.g. *currentLocation*, between classes *Person* and *Location*).

Next chapter presents our proposal for a CSS design process. It supports designers on building context models and designing CSS based on the concepts defined in the Context Metamodel.

CHAPTER

6

A CSS Design Process

Software process is a road map with predictable steps and guidelines related to the development of computer applications [Pressman, 2005]. It aims to support the creation of timely and high-quality products. As discussed throughout this thesis, CSS demands that designers consider new aspects and challenges in comparison with traditional applications. In this sense, it is important to provide CSS developers with a software process that could guide them through the fulfilment of context specific requirements. In this work, we argue that including context entails a different way of thinking about a system's engineering. When designing a CSS, a major emphasis should be given to the analysis of how users interact (or are expected to interact) with the CSS and how these users expect the CSS to act on their behalf.

In this chapter, we describe our proposal for a *Context Process*, which details the main activities related to context specification and the design of CSS, providing a systematic way to execute these activities. It extends the analysis and design phases described in any software development lifecycle (as explained in [Pressman, 2005]) and uses the notation and terminology described in the Software Process Engineering Metamodel (SPEM) [OMG, 2008a].

The chapter is organized as follows: Section 6.1 provides an overview of the Context Process, indicating its main elements and activities which are

detailed in subsequent sections (6.2, 6.3 and 6.4); and, Section 6.5 presents our concluding remarks and a comparison with related work.

6.1 Process Overview

The Software Engineering literature (e.g. [Pressman, 2005]) indicates that a software process, in general, comprises the following main phases: analysis, design, code generation, testing and maintenance. As illustrated in Figure 6-1 the main activities identified in the Context Process, and the corresponding phases in a software process, are: *Context Specification* (analysis), *Context Management Design* (design), *Context Usage Design* (design), *Code Generation*, *Testing* and *Evaluation* (maintenance). This last activity indicates that a CSS should undergo constant evaluations with final users to adjust adaptation functionalities. Currently, we concentrate on the activities related to analysis and design. The other activities are out of the scope of this thesis and will be considered in further work.

Three main *Process Roles* are considered:

- *System Architect* is the person or team responsible for designing the system's architecture [IEEE, 2000];
- *System Analyst* is the person or team responsible to identify users' needs and to translate business requirements into software specifications; and
- *Context Designer* is the person or team responsible to identify context-related requirements and to design context-sensitive solutions. The context designer should have multidisciplinary expertise on subjects related to human cognition, acquisition technologies (e.g. sensors), artificial intelligence, software development and software usability.

We chose to follow the terminology, diagrams and notation provided by the SPEM 2.0 (Software Process Engineering Metamodel) specification [OMG, 2008a]. SPEM is the OMG (Object Management Group) adopted standard for modeling software processes. It is a MOF-compliant metamodel and has an associated UML Profile. We modeled the Context Process using the SPEM Profile and two SPEM diagrams: *Workflow* (to illustrate how activities interact

with each other and the order they should be performed in) and *Activity Detail* (to present internal details of each activity, such as input/output artifacts and guidances). Next sections detail each main activity.

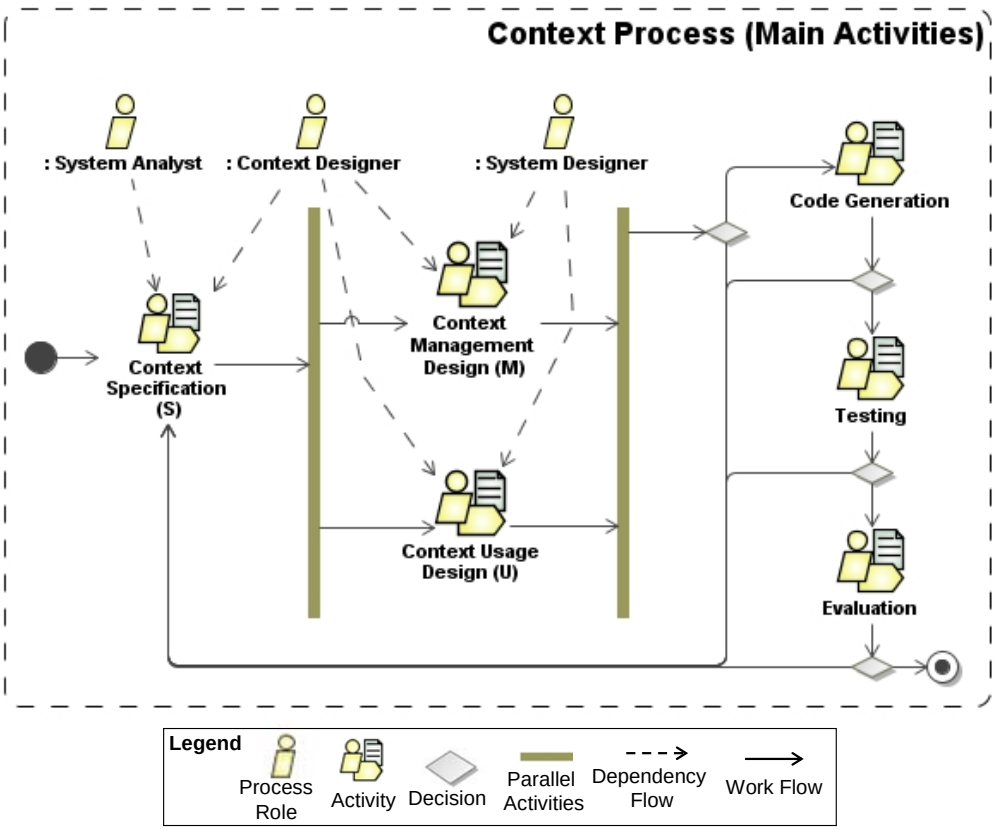

Figure 6-1 *SPEM Workflow Diagram*: Context Process Main Activities

6.2 Context Specification

This activity has the objective to identify the context requirements based on the business requirements and to create the context conceptual model. As illustrated in Figure 6-2, the *Context Specification* activity comprises the following sequential activities: *Identify Focus* (S1), *Identify Behavior Variations* (S2), *Identify Domain Entities and CEs* (S3) and *Verify CEs Relevance* (S4). The activities are executed incrementally, meaning that while executing a given activity it may be necessary to go back to a previous one. For example, after executing S3 one may decide to go to the next activity (S4), to go back to the previous activity (S2) for requirements review, or to start a new interaction, going back to S1 (this flow of activities should be repeated for each identified focus).

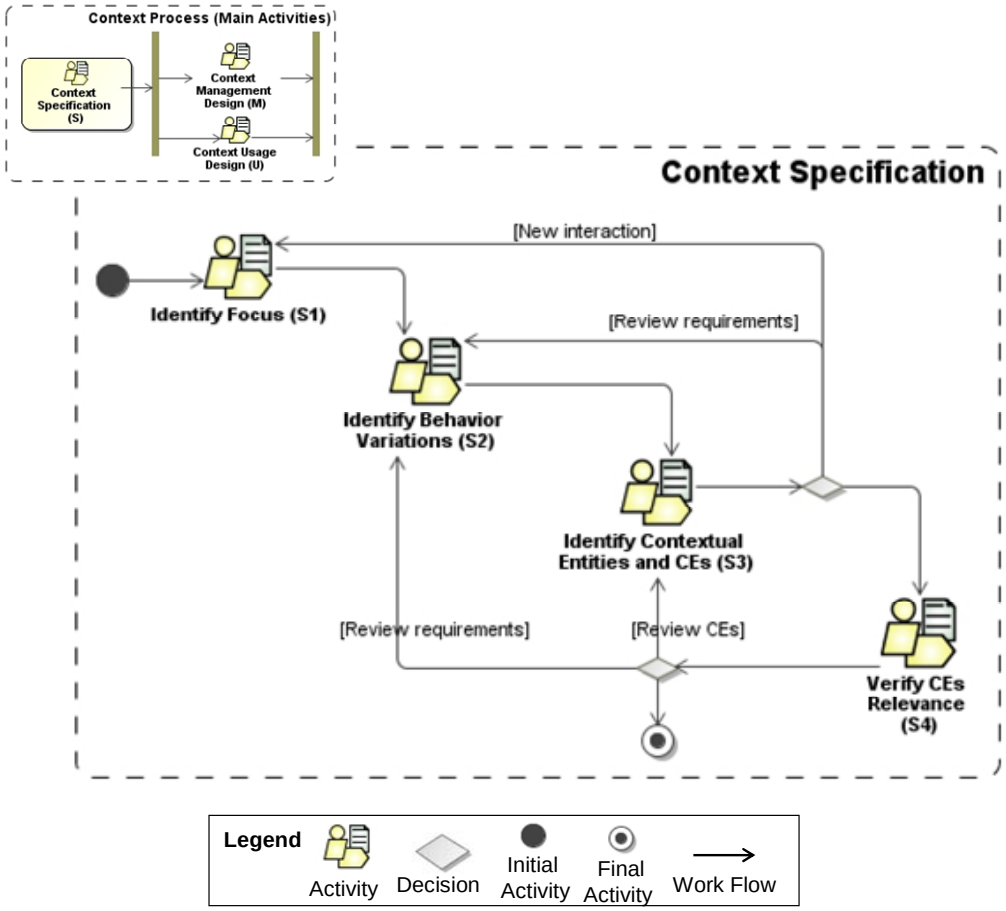

Figure 6-2 *SPEM Workflow Diagram*: Context Specification Activities

6.2.1 Identify Focus (S1)

The *objective* of the *Identify Focus* activity is to recognize, from the application business requirements, which tasks and agents should be considered to compose the CSS Foci. Figure 6-3 presents the *SPEM Activity Detail Diagram* for this activity. It is *performed* by the Context Designer in collaboration with the System Analyst. It is *guided* by the specifications defined in the Context Metamodel and the Context Profile stereotypes. It takes as *input* the application's original Use Cases Model, previously built by the System Analyst, which contains the main business requirements. It produces as *output* an extended version of the Use Cases Model enriched with the Focus identification.

As defined in the Context Metamodel, a Focus is composed by the association between an Agent and a Task, indicating that this Agent uses the CSS to execute that Task. These elements (*Task* and *Agent*) correspond, respectively, to *Use Case* and *Actor* in the Use Cases Model. The stereotypes

<<Agent>>, <<Task>> and <<executes>> defined in the Context Profile (Section 5.5.1), enable to identify which actors and use cases, should be considered to compose the CSS Foci, from the Use Case Model. For example, in the Mission Support System, examples of foci (illustrated in Figure 5-7) are the tuples: <Student, Book Hotel>, <Professor, Request Financial Aid>.

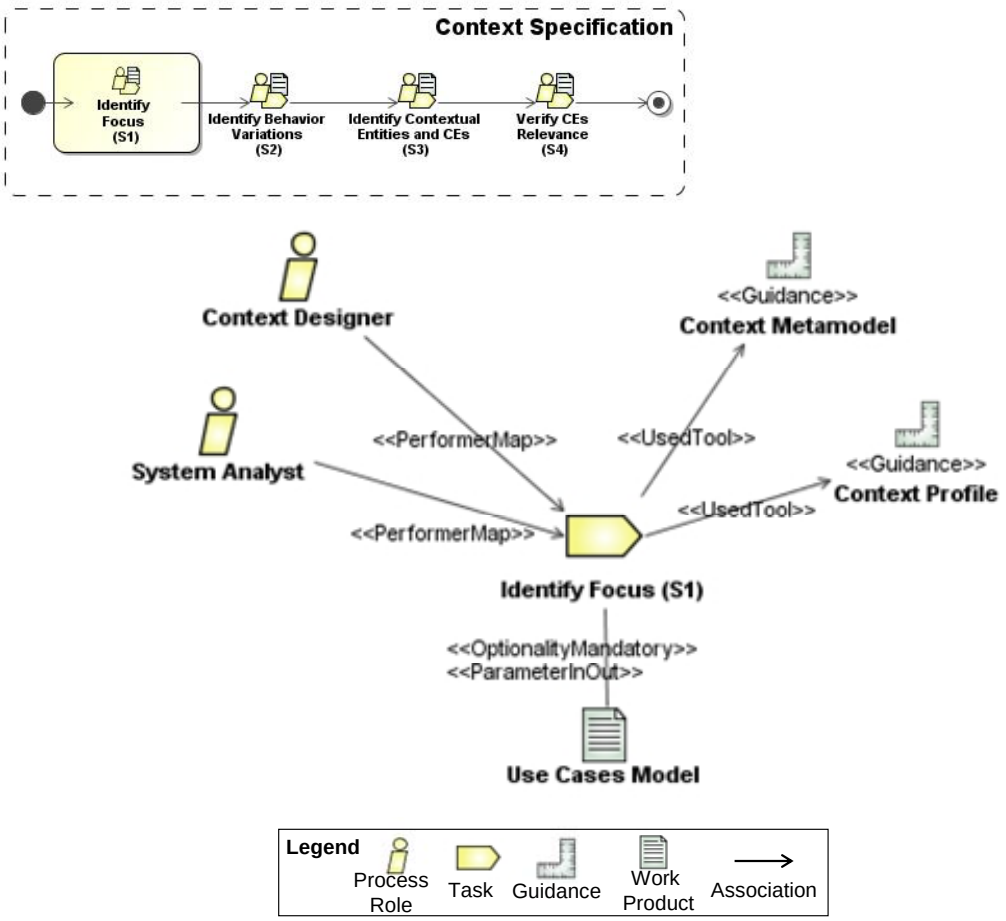

Figure 6-3 *SPEM Activity Detail Diagram*: Identify Focus

6.2.2 Identify Behavior Variations (S2)

Behavior variations indicate different actions, related to a Focus, that the CSS may execute. The *objective* of this activity is to identify, given a focus, which variations are expected in the CSS behavior, and which factors affect these variations. The characteristics of this activity (illustrated in Figure 6-4) are: it is *performed* by the Context Designer and System Analyst*;* it uses as *input* the extended Use Cases Model, which resulted from activity S1; it produces the Context Requirements document as *output*; and it uses, as *guidance*, a *Context Requirements Guidelines* document.

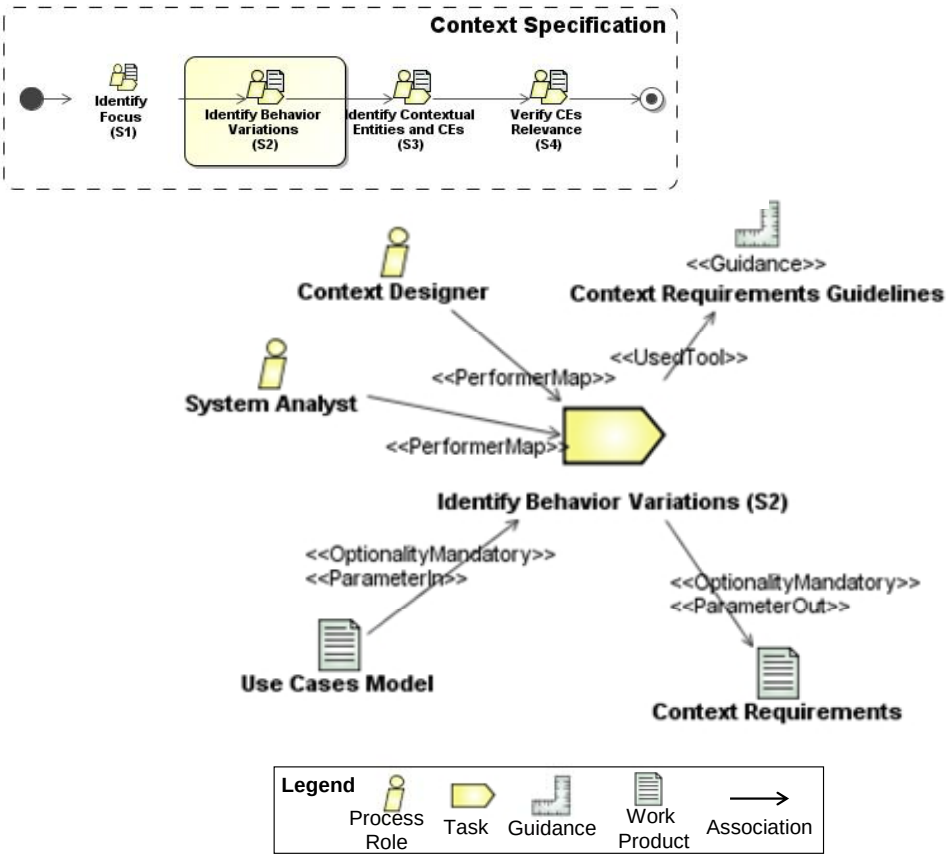

Figure 6-4 *SPEM Activity Detail Diagram*: Identify Behavior Variations

For the Context Requirements Guidelines, we identify some categories of requirements where context, generally, may interfere on a CSS behavior:

1) when the CSS should keep the Agent aware of contextual information related to the Task s/he is executing;
2) when some kind of adaptation or personalization should be provided by the CSS (e.g. change information presentation format, adapt interfaces, enable/disable services, and filter or classify information);
3) when the CSS functions as a recommender system; and
4) when there is a need to enrich managed knowledge with associated contextual elements (e.g. creation date, author, rationale behind a decision made).

This list is not exhaustive and context may affect the CSS behavior in other ways. Ideally, CSS designers should, incrementally, increase and share

this *Context Requirements Guidelines* document, reporting experiences and previously made decisions.

The produced *Context Requirements* document will contain a list and a description of the identified behavior variations. This document could follow proposed standards for requirements specification (e.g. [IEEE, 1998]).

In our example of the Academic Mission Support System, as discussed in Section 5.5.3, some requirements that affect the system behavior in the Focus <Professor, Book Transport> include:

R1 Recommend options of transportation for the mission location;

R2 Consider CEs related to the agent and the mission to categorize the available transports.

6.2.3 Identify Contextual Entities and CEs (S3)

After identifying a focus and the expected behavior variations for that focus, the next activity is to identify the domain entities related to the focus and the characteristics from those entities that influence each variation. This is the *objective* of activity S3, illustrated in Figure 6-5. This activity uses as *input* artifacts: the Context Requirements Document, produced in activity S2, and the domain Conceptual Model, previously specified by the System Analyst. The *output* artifact is the Context Conceptual Model. As *guidance*, it uses the Context Metamodel and Context Profile. Optionally, other *guidances* can support this activity, such as domain ontologies and existing context models (developed by other CSS designers).

The Context Conceptual Model, produced in this activity, is composed as an extension of the Domain Conceptual Model, taking into account the definitions identified in S2 and registered in the Context Requirements Document. The stereotype <<ContextualEntity>>, defined in the Context Profile, should be used to enrich the semantics of the Conceptual Model, explicitly indicating the entities (from the Conceptual Model) that should be considered in the Context Conceptual Model.

The Context Designer should verify, for each identified contextual entity, which of its properties influences the behavior variations (described in the

Context Requirements Document). These properties should be identified in the Context Conceptual Model with the <<ContextualElement>> stereotype. The context requirements may demand the specification of new properties for the domain entities, or the specification of new domain entities (not yet contemplated in the domain Conceptual Model).

An example of this activity is illustrated in Figure 5-8. The identified contextual entities are *Person* and *Mission* and the CEs are *Person.livesIn*, *Person.age*, *Mission.whoPays* and *Mission.occursIn*.

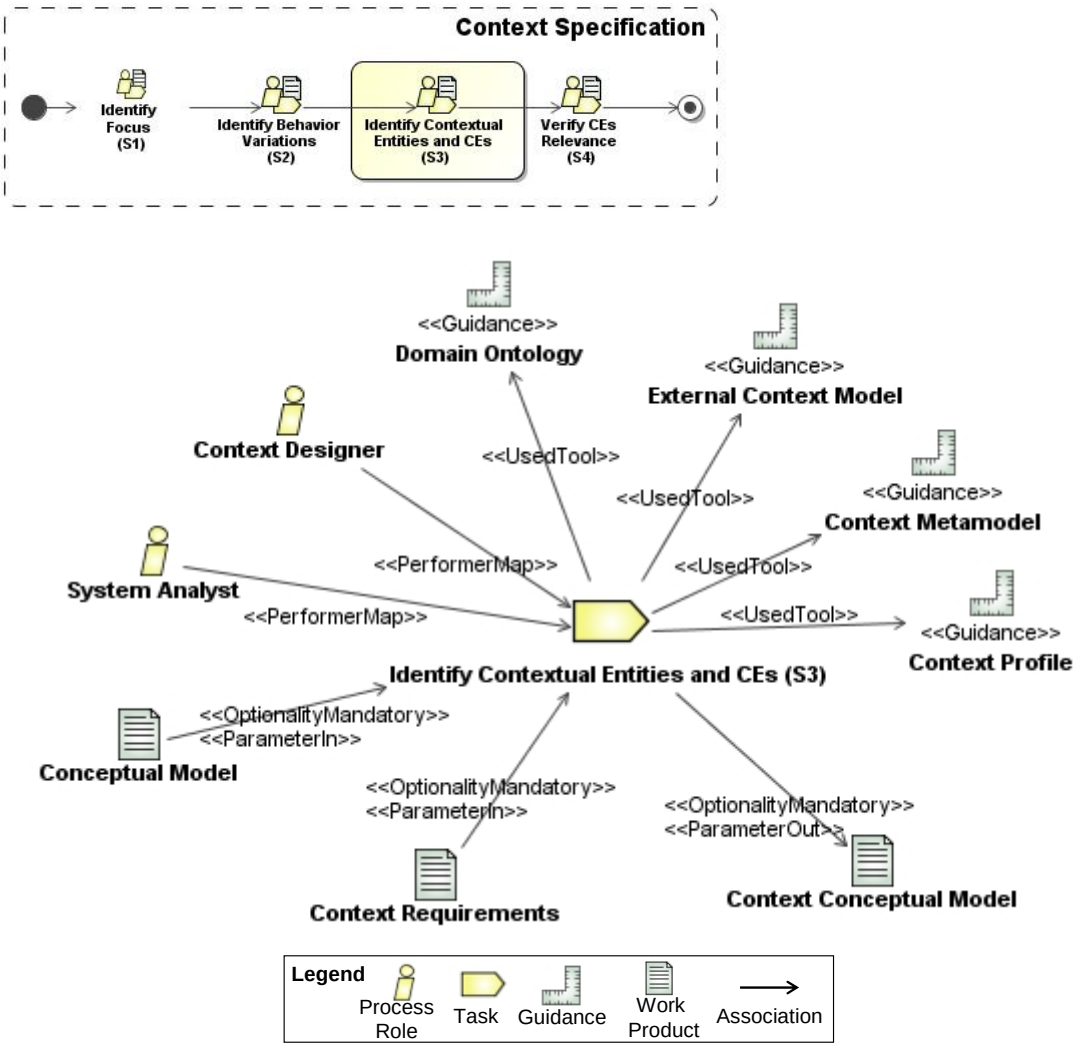

Figure 6-5 *SPEM Activity Detail Diagram*: Identify Contextual Entities and CEs

6.2.4 Verify CEs Relevance (S4)

The next step is to evaluate if the CSS final users and designers have the same understanding about the relevance of the identified CEs to the Foci, and if the

defined behavior variations reflect users' expectations. This is the *objective* of activity S4. As illustrated in Figure 6-6, the activity is *performed* by the Context Designer, it receives as *input* the Context Conceptual Model (produced in S3) and the Context Requirements document (from S2), and produces as *output* a Relevance Evaluation document. It may also produce, as *output*, updated versions of the Context Conceptual Model and Context Requirements Document. As *guidance* it may use Evaluation Guidelines (e.g. questionnaire samples).

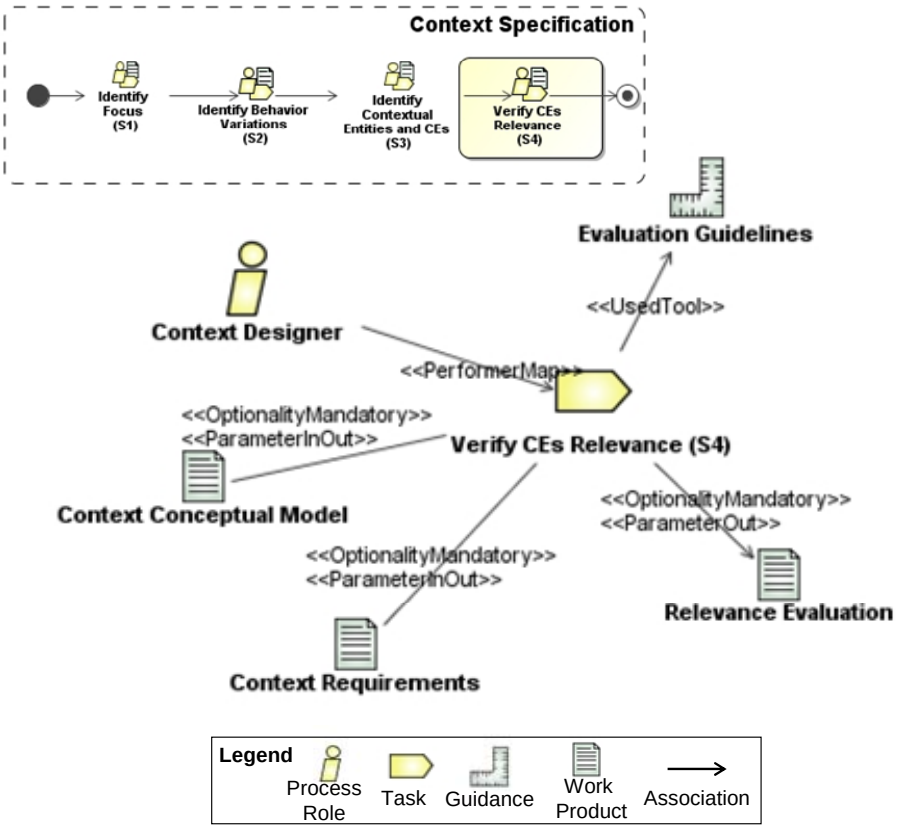

Figure 6-6 *SPEM Activity Detail Diagram*: Verify CEs Relevance

As stated in [Brézillon, 2007b], practices are strongly influenced by the context. Practices are related to the real usage of the CSS by its final users. To identify some of these practices, the Context Process suggests that a preliminary investigation should be conducted with a group of (potential) real users. The Evaluation Guidelines may contain indications about how to conduct this investigation.

One approach, suggested in [Greenberg, 2001], is to use ethnographic methods to observe and analyze many contextual episodes in real life. This

observation will enable the designer to recognize how people really work. It could be possible, then, to determine the contextual elements that are relevant, and how they are acquired and identified.

Another more direct approach is to elaborate survey questionnaires and face-to-face interviews. They can be applied to (potential) CSS users to investigate which CEs they consider relevant (and how relevant) when executing the task defined in the focus. Data provided on these surveys can be analyzed using data mining techniques to identify association patterns between users' characteristics and their perception of CEs relevance.

In our example of the Mission System (Section 5.1), we could perform surveys and interviews with students and professors from distinct universities to investigate: how they usually plan their missions and what elements influence their decision when choosing accommodation and transportation. This information along with personal information about each participant (e.g. age, gender, social status and living location) could support identifying behavior patterns that could be used to compose the Mission System contextual rules.

6.3 Context Management Design

Once the Context Specification is defined, the Context Designer has to investigate how the specified CEs should be acquired and processed. For projects where multiple context consumers are considered, the Context Designer should also define how the managed CEs will be disseminated to the different consumers. The *Context Management Design* activity (illustrated in Figure 6-7) comprises the following sub-activities: *Specify Context Acquisition* (M1), *Design Acquisition Module* (M2), *Design Processing Module* (M3) and *Design Dissemination Module* (M4). These last three are independent from each other, and can be executed in parallel or following any order.

6.3.1 Specify Context Acquisition (M1)

The *objective* of this activity is to specify the acquisition parameters for each identified CE. As illustrated in Figure 6-8, it is *performed* by the Context Designer and the System Designer. It receives as *input* the Context Conceptual Model and the Context Requirements document. It produces as *output* an

updated version of the Context Conceptual Model and an Acquisition Configuration document. It uses the Context Metamodel and the Context Profile as *guidance*s.

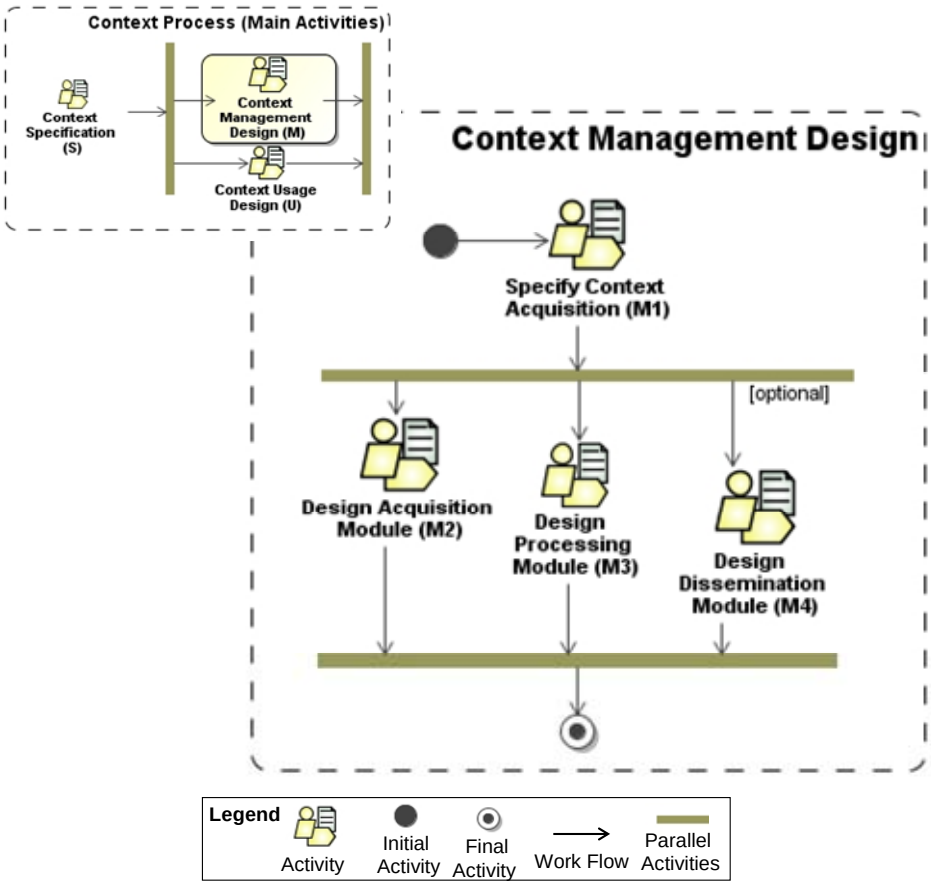

Figure 6-7 *SPEM Workflow Diagram*: Context Management Design Activities

Following what is specified in the Context Metamodel (Section 5.3), the Acquisition Configuration document should contain, for each CE: a reference to the context source that will provide its values; the acquisition mode (e.g. *sensed*, *profiled*, *userDefined*, *queried* or *derived*); the update frequency (*never*, *occasionally*, *often* and *always*); and, optionally, a matching expression indicating any convertions that should be done in the information coming from the context source to match with the CE expected format.

For example, in the Academic Mission System (presented in Figure 5-8), the CE *Person.availability* is filled by the context source *MSNAdapter*. The acquisition type is *Sensed*, the update frequency is *often*, and a matching expression is *{if Status='Online' then availability=0.8; if Status='Busy' then*

availability=0.5; if Status='Away' or Status='Offline' then availability=0.2)}, where *Status* is the information provided by the MSNAdapter.

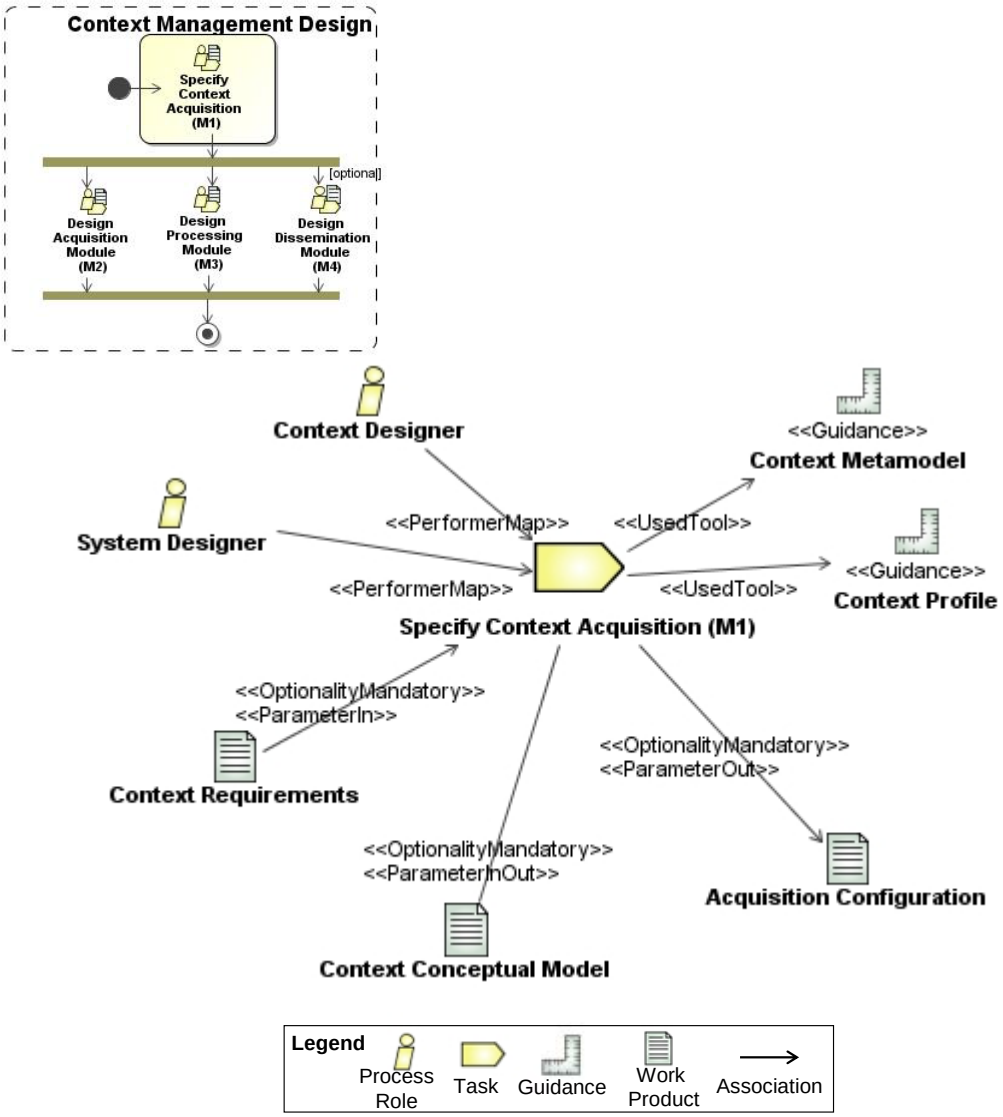

Figure 6-8 *SPEM Activity Detail Diagram*: Specify CE Acquisition

6.3.2 Design Acquisition Module (M2)

The *objective* of this activity is to design the elements responsible for the context acquisition (e.g. context sources APIs and adapters), indicating how the context acquisition should be implemented. As illustrated in Figure 6-9, it is *performed* by the Context Designer and the System Designer, receives as *input* the Acquisition Configuration document and produces as *output* the Acquisition Module Specification. It uses the Context Architecture (defined in Section 4.4) as *guidance*.

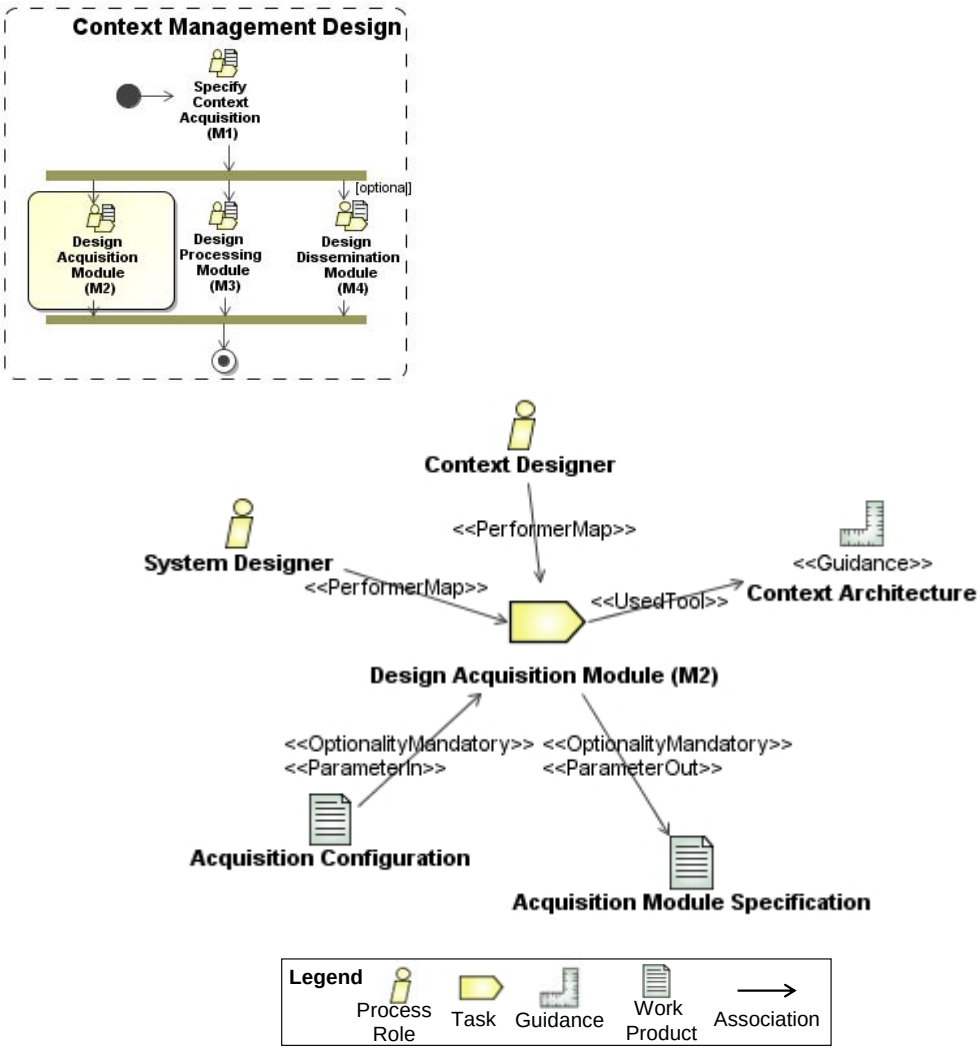

Figure 6-9 *SPEM Activity Detail Diagram*: Design Acquisition Module

According to the Context Architecture (presented in Section 4.4), each context source should have: an API (Application Programming Interface) to allow the access to its internal functionalities, and an *Adapter* to enable the communication between the context source and the context management module. In this sense, for each identified context source (activity M1), the Context Designer should execute one of the following tasks:

1) *Design a Context Source Adapter* in the case that there is already an API to access the context source contents and it is only necessary to translate the information from the context source to the CSS;

2) *Design a Context Source API and Adapter*, when the context source exists, but there is not a modularized way to access its internal information; or

3) *Design a new Context Source*, when no context source was found to provide that CE.

For example, in the Academic Mission System (illustrated in Figure 5-8), two adapters were designed for two existing context sources: MSNAdapter and GeoLocationAdapter; and two new context sources were designed: *Missionary Profile* and *Mission Form*.

A CE can be acquired using one of the following three ways:

1) explicitly informed by the *user*, which answers, on demand, questions about her/his context through a user interface;

2) automatically using *physical* or *logical sensors*, which monitor and collects information either from a physical (e.g. indoor or outdoor places) or virtual (e.g. workspace area, running applications, network) environment;

3) semi-automatically, from *shared repositories* (e.g. applications' models, filled profiles and preferences forms, organizational memory, domain ontologies).

An important characteristic of a CSS is that they have the potential to diminish the interaction between systems and users by acquiring, as much information as possible, from different sources others than the user. For example, a system may ask directly the user about his/her current location. However, a context-sensitive system will try to acquire this information using location sensors combined with reasoning mechanisms. In this light, a CSS is expected to interact with several different and, possibly heterogenous context sources.

Context acquisition services should be constantly available. These services should, preferably, be implemented independently from the CSS, so different applications could share information without worrying about how the CE values are obtained. By doing so, a community of CSS designers and context sources developers could publish the assets related to their context sources (e.g. documentation, source code, software components, models, CEs catalog) in a *Context Sources Yellow Pages* service. Such a service could guide

other CSS designers on specifying their acquisition modules. This service is out of the scope of this work.

6.3.3 Design Processing Module (M3)

This activity has the *objective* to specify and design the elements related to CE processing, i.e. CE knowledge base, inference rules and inference engine (as indicated in the Section 4.4.2). As shown in Figure 6-10, this activity: is *performed* by the Context Designer and the System Designer; receives as *input* the Context Conceptual Model, the Context Requirements document and, optionally, the Context Behavior Model; produces as *output* a document with the defined Contextual Rules, a document with the specification of the Processing Module elements, and an updated version of the Context Behavior Model. It uses as *guidance* the Context Architecture (defined in Section 4.4).

In this activity, the Context Designer should identify, for each CE, whether it is necessary to perform any processing or inferencing to determine its value. Behaviors defined in the Context Behavior Model should be translated into contextual rules. The designer should identify what kind of inferencing will be needed and in this case what existing inference engine better applies. Contextual rules should be translated into the format required by the inference engine. Besides, the inference engine should be configured to match the CSS processing requirements.

In the Academic Mission System, for instance, some contextual rules were identified from the modeled contextual graph (illustrated in Figure 5-10). An example of contextual rule is:

```
Rule1:
   Conditions
      not (Mission.occursIn==Person.livesIn)
      Mission.whoPays=“CAPES”
   Actions
      CallBehavior(“Contact CAPES Official Agency”)
```

In order to be used by an inference engine, this rule must be translated to an appropriate format. Considering, for instance, the JEOPS (Java Embedded Object Production System) inference engine notation [Figueira Filho and Ramalho, 2000] the rule will have the following specification:

```
rule Rule1 {
```

```
        declarations
             Person p;
             Mission m;
             BookTransport bt;
        conditions
             m.getOccursIn() == p.getLivesIn();
             m.getWhoPays() == “CAPES”;
        actions
             bt.recommendTransport(false);
             bt.showMessage(“Contact   CAPES   Official
             Agency”);
    }
```

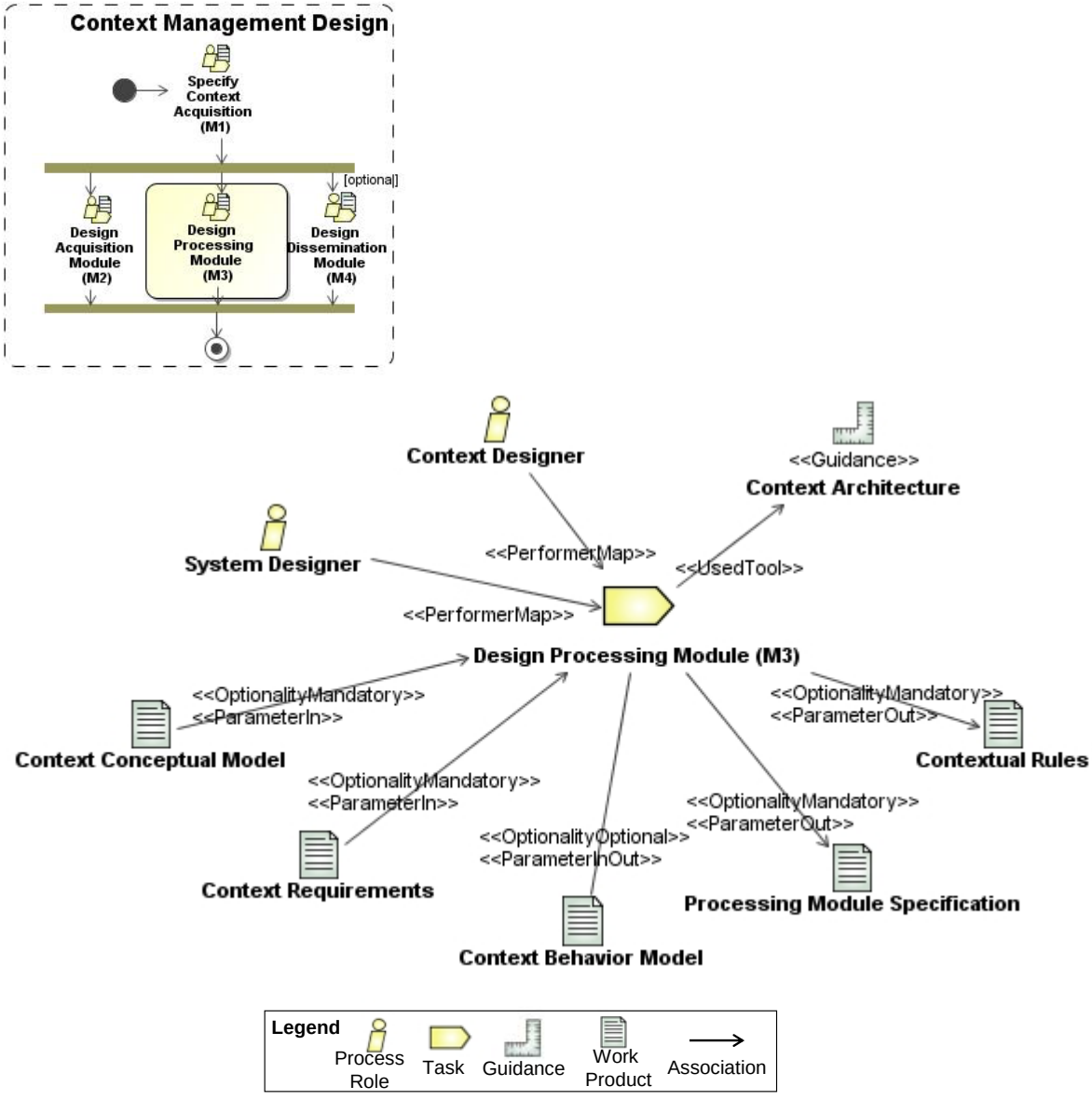

Figure 6-10 *SPEM Activity Detail Diagram*: Design Processing Module

6.3.4 Design Dissemination Module (M4)

The *objective* of this activity is to design the elements responsible for disseminating CEs to different context consumers. Figure 6-11 shows the details of this activity: it is *performed* by the Context Designer and the System Designer, receives as *input* the Context Conceptual Model and the Context Requirements document, and produces as *output* the Dissemination Module Specification. Its *guidance* is the Context Architecture.

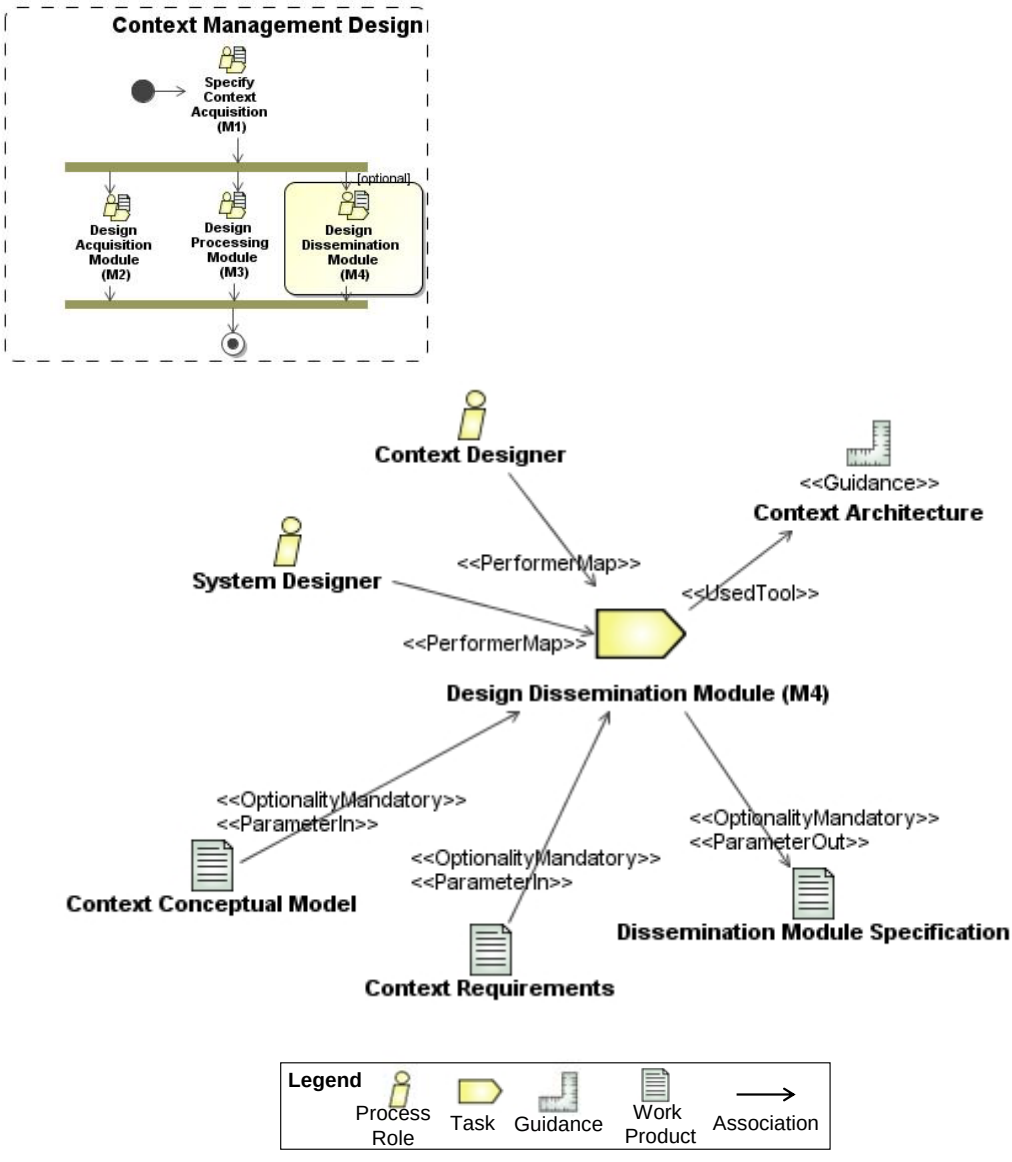

Figure 6-11 *SPEM Activity Detail Diagram*: Design Context Dissemination

When different context consumers are considered, it is necessary to design how context dissemination should be performed. The Context Architecture indicates that each context consumer should have an adapter to

allow its communication with the CSS context management module. The adapter indicates how the consumer desires to be notified when the CE is updated.

6.4 Context Usage Design

This activity has the *objective* to design how context will be effectively used in the CSS. It is composed by three main activities: *Design Context Behavior Model* (U1), *Design Context Adaptation* (U2) and *Design Context Presentation* (U3). The task *Get Focus* indicates that these activities should be performed for each Focus identified in the Context Conceptual Model. We consider two main usages for context in a CSS: to support behavior adaptation (of any kind); to enrich a CSS agent´s cognition with contextual information managed by the CSS. In this light, activities (U2) and (U3) are presented as optional in the process, and the desiger should decide which activity is necessary given the CSS requirements.

6.4.1 Design Context Behavior Model (U1)

This activity has the *objective* to produce the Context Behavior Model corresponding to the identified focus, as well as to design the associations between the CEs and the behavior variations. The activity (as shown in Figure 6-13) is *performed* by the Context Designer and System Designer. Its *input* is the Context Conceptual Model and the Context Requirements document. Its *output* is the Context Behavior Model for the focus. As *guidance*, it uses the CxG Profile (described in Section 5.5.2).

The Context Behavior Model is a representation of the correspondent contextual graph in the Focus (see, for example, Figure 5-10). The contextual graph expects to receive, at least, the actions that should be triggered by the CSS and the conditions that constrain these actions. These conditions can be identified from the contextual rules defined in activity M3.

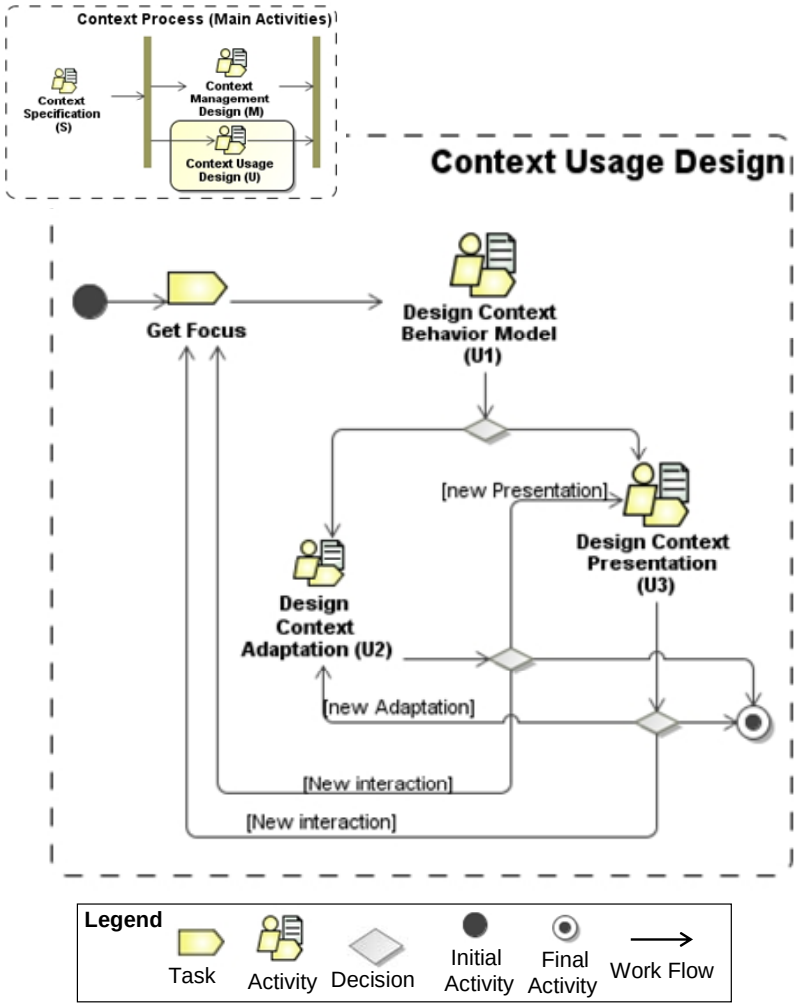

Figure 6-12 *SPEM Workflow Diagram*: Context Usage Design Activities

6.4.2 Design Context Adaptation (U2)

This activity *aims* to specify how the CSS should adapt to the context. As illustrated in Figure 6-14 it is *performed* by the Context Designer and System Designer. It uses as *input* the Context Conceptual Model, the Context Requirements document, and the Context Behavior Model. As *output* it generates the Adaptation Module Specification. To *guide* this activity, the designer may use specifications provided by the Context Architecture and guidelines with directives related to Adaptation and Usability Aspects.

As discussed in Section 2.2.3, misinterpretations may entail undesired behaviors, which will make the CSS annoying and disturbing instead of useful. In this sense, issues related to human aspects such as intelligibility and accountability (e.g. non-intrusiveness, user control, privacy, and security)

should be taken into account (as also discussed in [Bellotti and Edwards, 2001]).

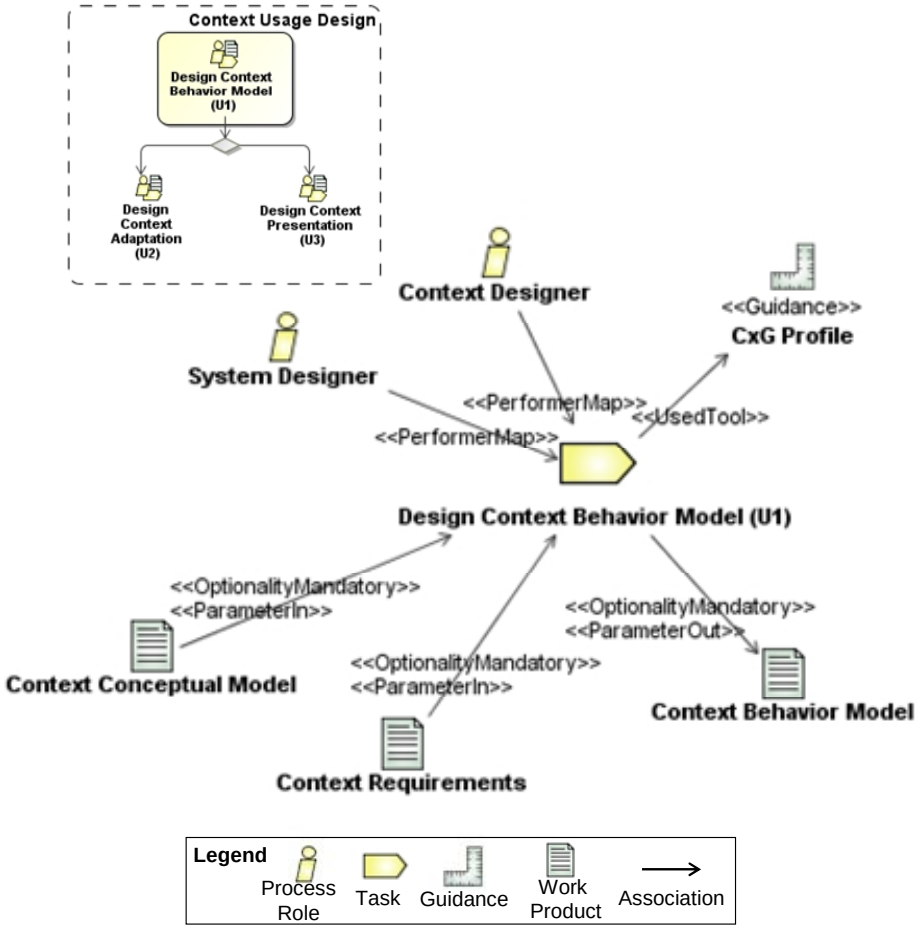

Figure 6-13 *SPEM Activity Detail Diagram*: Design Context Behavior Model

The way the CSS interacts with its users and how users can provide feedback about the executed adaptations should be also specified. This feedback can aid the system to learn how to conduct future adaptations.

6.4.3 Design Context Presentation (U3)

The *purpose* of this activity is to design the presentation of the managed CEs to the CSS Agents in order to enrich their knowledge about the Task being executed. Details of this activity are illustrated in Figure 6-15: it is *performed* by the Context Designer and the System Designer. The *inputs* are: the Context Conceptual Model, the Context Requirements Document, and the Context Behavior Model. As *output* it generates the Presentation Specification document. To *guide* this activity, the designer may use specifications provided

by the Context Architecture and guidelines with directives related to Interface and Usability Aspects.

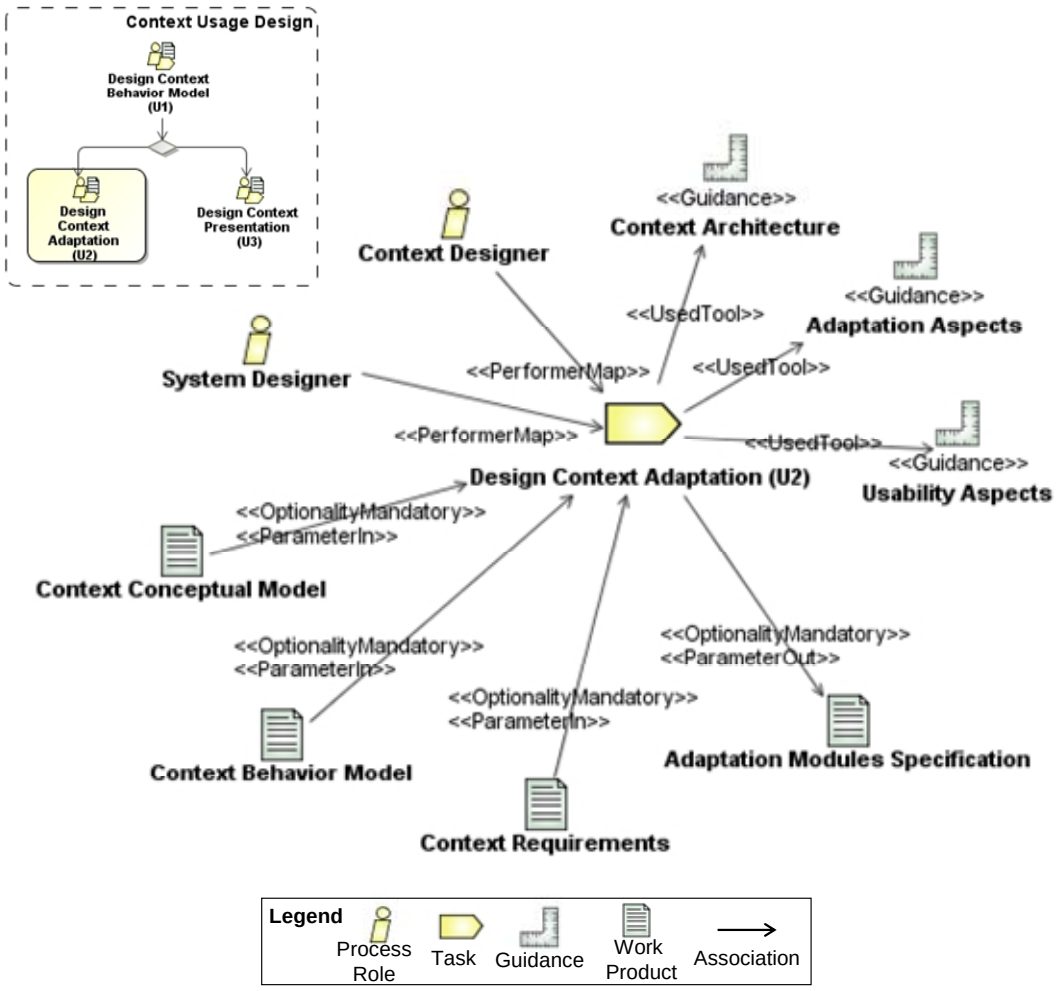

Figure 6-14 *SPEM Activity Detail Diagram*: Design CSS Adaptation

The Context Designer should decide which CEs should be presented, when and how the presentation should occurs. Issues related to human cognition should be considered, to avoid intrusiveness or information overload. The literature of information awareness provides many examples about how to present contextual information to improve cognition (e.g. [Vieira et al., 2004, Byrne, 2004, Gross and Prinz, 2003, Gutwin and Greenberg, 2002]).

6.5 Concluding Remarks

This chapter presented a software process to support the context specification and the design of CSS. The main contributions of the Context Process are:

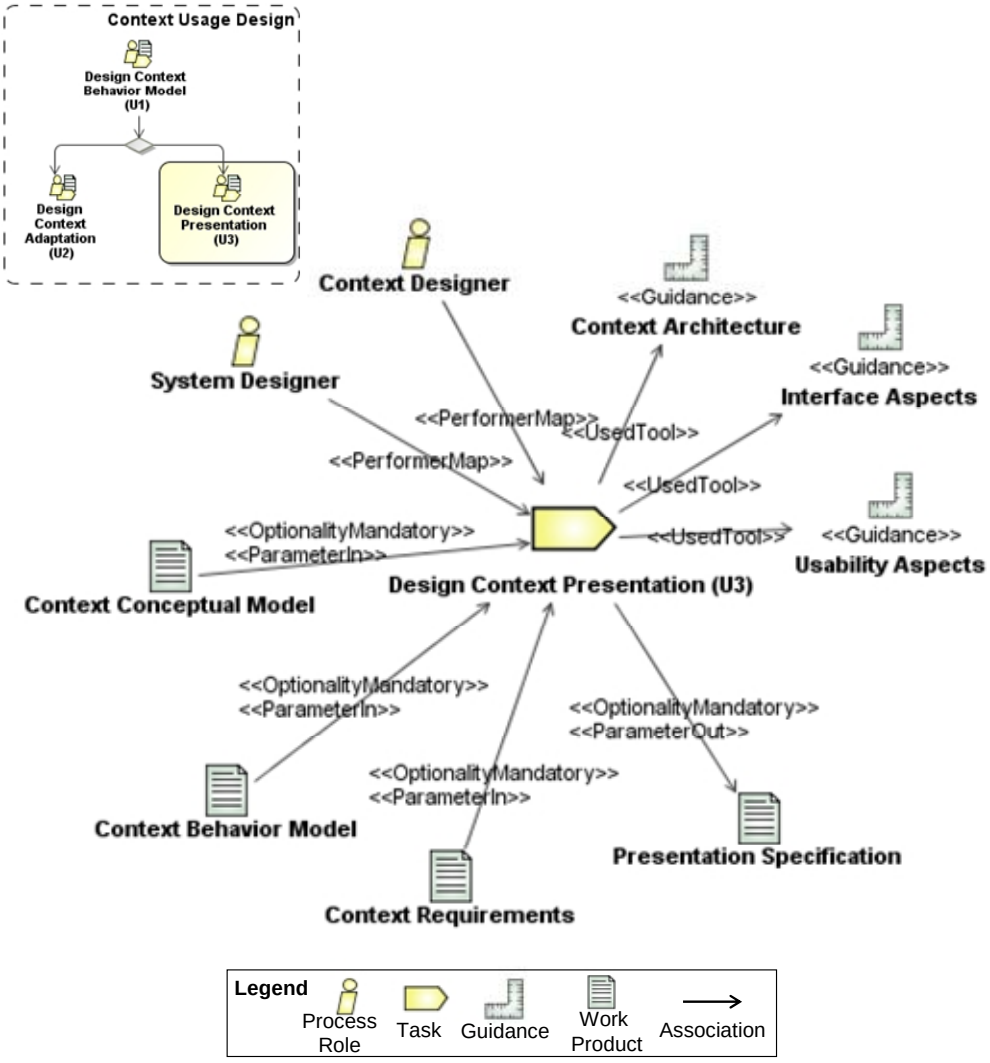

Figure 6-15 *SPEM Activity Detail Diagram*: Design Context Presentation

- It proposes a clear separation of the context-related activities, creating a new role in the software development team, the *context designer*;
- It emphasizes the need to work with existing artifacts when designing a CSS (e.g. requirements, conceptual models, business logic), instead of starting from scratch;
- It covers the main activities related to a CSS design, providing guidelines, indicating input/output artifacts and a systematic way to execute each activity;
- It illustrates how the context metamodel and profiles (described, respectively in Section 5.3 and Section 5.5) and the context

architecture (described in Section 4.4) can be integrated when designing a CSS.

As discussed in Section 3.3, few software processes were proposed to support designing CSS (e.g. [Bulcão Neto et al., 2006, Henricksen and Indulska, 2004]). The POCAp process [Bulcão Neto et al., 2006] is also based on the SPEM notation and describes, in a high level, the activities related to building a CSS, as an instantiation of a common software process. Its main drawback is that it assumes that the CSS is an ontology-based solution. In this sense, guidelines are specifically associated to the aspects related to ontologies manipulation. The high-level process proposed in [Henricksen and Indulska, 2004] contains only a flow of activities that should be followed to use their context modeling language and CSS programming abstractions. It does not mention the artifacts (input and output work products, guidances, process roles) related to the activities, neither provides guidelines explaining how to perform each one. Both processes ([Bulcão Neto et al., 2006, Henricksen and Indulska, 2004]) do not go into details about the activities related to Context Specification and CSS Design. In particular, they do not indicate how existing business models could be reused and extended to generate the context models.

Next chapter describes a case study performed to investigate the feasibility of the proposed ideas, and presents an instantiation of the Context Process.

CHAPTER

7

Case Study

This chapter describes a case study carried out to verify the feasibility of our proposal. To verify the generality of the framework to support applications in distinct domains, we created two design projects for different applications: the *Academic Mission Support System*, introduced in Section 5.1 and described throughout this thesis, and an *Expert Recommender System*, named ICARE.

ICARE (Intelligent Context Awareness for Recommending Experts) [Petry et al., 2008, Petry et al., 2006] is an Expert Recommender System (ERS) that considers contextual information about users[1] and experts when processing recommendations. An ERS is a system that returns references to individuals identified as experts in a requested domain and that can be used to connect human actors [Reichling et al., 2005]. When performing a task, solving a problem or making a decision, people can save time and effort if they interact with others with the necessary knowledge and experience in the task at hand.

To be effective, the ERS should consider the context of people involved in the recommendation (i.e. the user requesting the recommendation and the recommended expert). Context can support ERS to provide better recommendations since it can inform who the user is, the role s/he performs in the organization, the knowledge areas s/he has worked with, the subjects s/he is

[1] In ICARE, *user* refers to the person who is requesting the recommendation.

an expert in. For example, it may be important to consider whether the expert is available to interact, if s/he has a good reputation, if the user already knows the recommended expert, otherwise if they have common friends that could introduce them or if user and expert have already worked together.

In this chapter, we present a case study related to the modeling and design of context in ICARE. The chapter is organized as follows: Section 7.1 presents the ICARE's preliminary requirements and conceptual model, without considering context; Section 7.2 illustrates how the *Context Process* was applied to design ICARE and detail the execution of each activity defined in the process; Section 7.3 describes a functional prototype of ICARE and discusses its evaluation by final users; and Section 7.4 presents some concluding remarks.

7.1 Preliminary Requirements and Conceptual Model

Figure 7-1 illustrates the use case diagram for ICARE, which indicates its main features and interactions with external entities. Three *actors* are considered: the *User* who requests the recommendation, an *External System* who requests the recommendation (when ICARE is plugged as an additional service inside another application), and the *Expert* who is being recommended.

The following *use cases* are specified: (i) the user should *register a profile* before starting using the system; (ii) the user should *log in* to start searching for experts; (iii) the user or the external system can start a *search for an expert* by providing keywords; (iv) ICARE generates and sends a classified list of experts to the external system; and (v) ICARE shows the classified list of experts to the user.

Analysing the task *Search Experts*, we established the following preliminary requirements:

R1 Users must provide one or more keywords to search an expert;

R2 To produce better results, these keywords should be mapped into concepts defined in a domain ontology;

R3 The subjects of expertise should also have a correspondence with the concepts specified in the domain ontology;

R4 Experts corresponding to the user's request will be identified by matching the concepts related to the keywords with the subjects of expertise associated to the experts;

R5 A list of experts classified by their expertise degree in the identified subjects should be presented to the user.

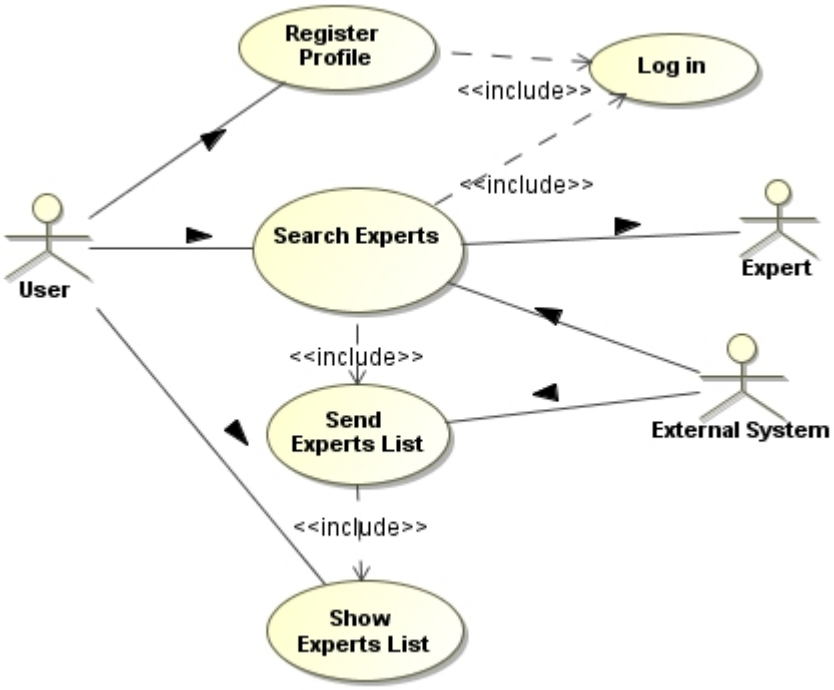

Figure 7-1 ICARE's Use Cases Diagram

Figure 7-2 illustrates ICARE's Conceptual Class Diagram. The *User* is classified as a *Person* and the *Expert* is also a *User*. A *Person* has personal information (*name*, *birthDate*, *photoURL*, *homepage, address*), professional information (*academicDegree*, *yearsExperience*, *organizationLevel* in the *Organization* s/he *worksIn*), indication of *Subjects* of *interest* and *Communication Contacts* (e.g. *email*, *phoneNumber*, *msnId*). An *Expert* also has an indication of *expertiseLevel* in one or more *Subjects*.

7.2 Applying the Context Process to ICARE

This section illustrates how the Context Process was used to guide the context specification and the design of context functionalities in ICARE. Activities are performed following the sequence suggested in the process.

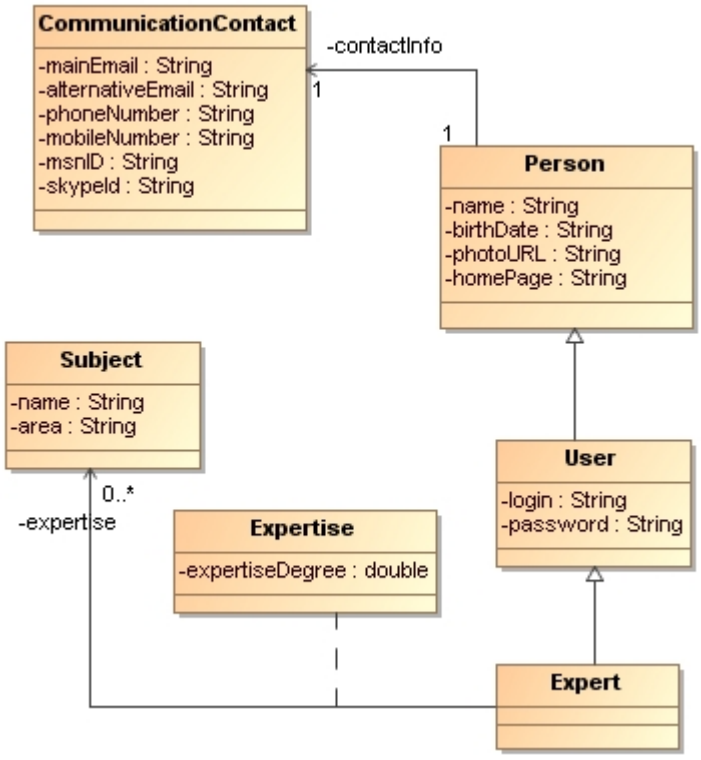

Figure 7-2 ICARE's Preliminary Conceptual Class Diagram

7.2.1 Context Specification

The first activity in the Context Process comprises four sub-activities: *Identify the Focus*, *Specify Behavior Variations, Identify Contextual Elements* and *Verify CEs Relevance*. This section describes how these activities were performed in ICARE.

- *Identify the Focus (S1)*

By analyzing ICARE's Use Cases Diagram (Figure 7-1) we consider as Focus the association between the <<Agent>> *User* and the <<Task>> *Search Experts* (Figure 7-3). Our interpretation is that users come to ICARE mainly to find experts that match their needs (by providing keywords). The other requirements are complementary and intend to support this task. Another focus that could be considered is the association between the <<Agent>> *External System* and the <<Task>> *Search Experts*. This is relevant when ICARE is installed as a plugin in another application (e.g. Text Editor or Email Reader), and the experts search is activated by that application. However, this functionality was not considered in the current version of the application. Figure 7-3 illustrates the

extended version of the Use Cases Model, enriched with the Context Profile stereotypes (presented in Section 5.5.1) is the output product for this activity.

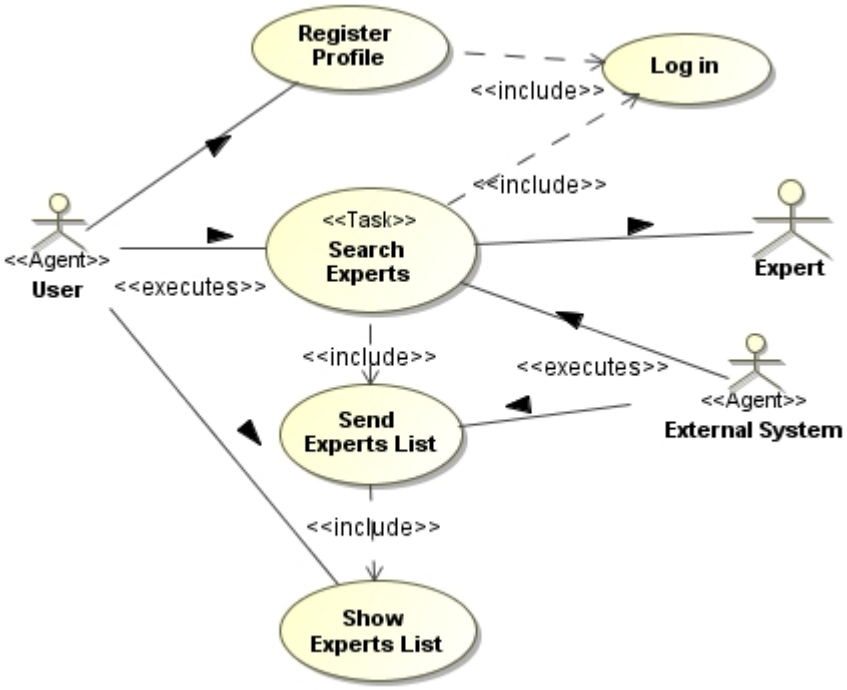

Figure 7-3 ICARE's Use Case Diagram Enriched with Context Profile Stereotypes

- *Specify Behavior Variations (S2)*

Next step in the Context Process is to specify the expected variations in ICARE behavior influenced by the context.

The conventional behavior of ICARE (as described in Section 7.1) is to receive as input a list of keywords and to return a list of experts ranked by their expertise level in the subjects that correspond to those keywords.

The context-sensitive behavior defined for ICARE is to consider, additionally, the users' and experts' context to prioritize experts that better fit not only the informed keywords but also the user's context. In this sense, ICARE receives as input a list of keywords and the User CEs and returns a list of experts ranked by their CEs. This list of experts is increased with information about each Expert CEs.

In this light, the following context-related requirements were identified:

R6 Identify CEs associated to the User who is performing the search;

R7 Use the User's CEs to improve the matching between keywords and ontological terms;

R8 Identify the CEs to categorize the Experts;

R9 For each retrieved Expert, identify the current value of her/his CEs and include these CEs in the list generated in the search;

R10 The CEs associated to the Experts should be matched to CEs identified for the User to improve the classification of the experts' list.

This list of requirements composes the *Context Requirements* document specified in the process as the output product for this activity.

- *Identify Domain Entities and CEs (S3)*

Analyzing the context requirements (produced in *S2*) and the ICARE's original Conceptual Model (Figure 7-2), we identified the need to specify new CEs to better characterize users and experts. These new elements are illustrated in Figure 7-4 through the stereotype <<ContextualElement>>. The domain entities *Person*, *User* and *Expert* are identified by the stereotype <<ContextualEntity>>. The new defined CEs are explained below:

- *availability* (User, Expert): indicates how busy the user or the expert is. An available expert is more likely to accept to engage in a new interaction;
- *knows* and *socialDistance* (User, Expert): indicates, respectively, a social relation between two people and the number of people that socially separates them. According to existing studies (e.g. [Galegher et al., 1990, Granovetter, 1973]), colleagues and friends are more likely to assist and interact with each other than strangers;
- *currentLocation* (User, Expert): refers to the physical location the person is currently in. Knowing that an expert is at the same place that s/he is, the user may choose to contact her/him face-to-face;
- *contactInfo* (Expert): informs how a person can be contacted. Knowing how to contact an expert, the user may choose to contact

her/him immediately (synchronous communication) or at another time (asynchronous communication);

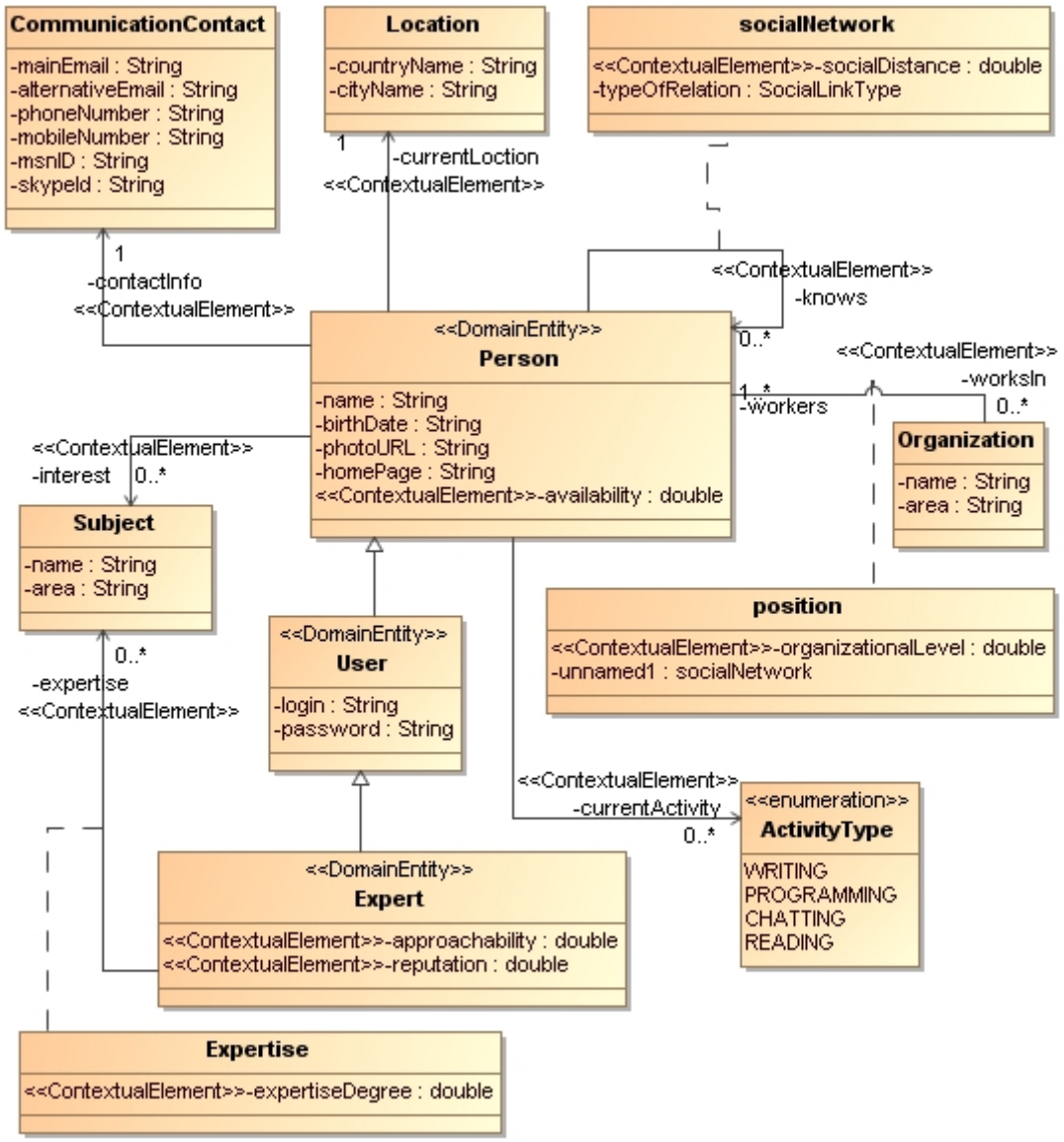

Figure 7-4 ICARE Conceptual Class Diagram enriched with Context Profile Stereotypes and new CE Definitions

- *worksIn* and *organizationalLevel* (User, Expert): identifies the work relation between a person and an organization and indicates the position of the person in the organization. Users with a high organizational level may demand the support of the best expert in the subject, whether or not the expert is available. Moreover, a user that occupies a low position may face difficulties in establishing interactions with high position experts;

- *currentActivity* (User, Expert): represents the activity the person is currently performing. The user's current activity may support the resolution of the provided keywords into ontological terms;
- *interest* (User, Expert): identifies the subjects a person has interest in. It may help identifying mutual affinities between users and experts;
- *expertise* and *expertiseDegree* (Expert): indicates the level of expertise of an expert in a subject;
- *approachability* (Expert): denotes how easy it is to contact the expert. An expert that is frequently online, even with a busy signal, is more accessible than another one that is regularly disconnected;
- *reputation* (Expert): points out the expert's overall quality as judged by users who contacted her/him.

The extended version of the ICARE's Conceptual Model is the Context Conceptual Model (Figure 7-4), which is the output for this activity.

- *Verify CEs Relevance (S4)*

To evaluate the CEs identified for ICARE, we designed an experiment that was conducted with fifty (50) participants from different research and development organizations in Informatics (presented in [Petry et al., 2008]). A questionnaire was applied presenting some of the CEs identified in *S3* to the participants. People were asked whether or not they would consider those CEs when filtering and ranking the searched experts. They were also asked to rank the CEs by importance, according to their view. This investigation refers to the following CEs, related to the experts ranking: *availability*, *reputation*, *expertise degree*, *organizational level* and *social distance*.

Figure 7-5 shows the results of the CEs relevance evaluation: the *availability* of the recommended expert was considered relevant by 94% of the participants; 92% agreed with considering her/his *reputation*; 88% thought that is relevant to take into account the *level of expertise* of the recommended expert in the subject area; and 54% approved taking into account the differences in the *organizational level* between the user and the recommended experts. An interesting result of this investigation is that most of the participants (64%) did not consider relevant the *social relationship* between users and experts. This

result contradicts what we initially thought; we believed that the recommendation could be more effective (resulting in more opportunities for collaboration) if the user already knows the recommended expert or if they have common friends or colleagues.

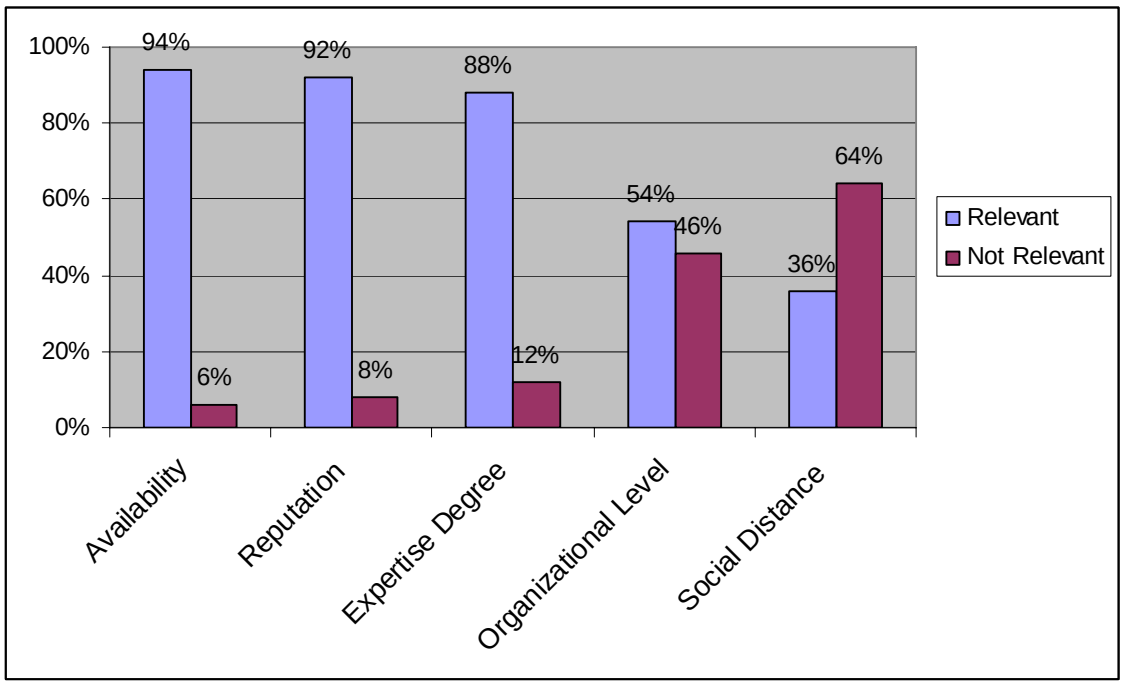

Figure 7-5 Evaluation of CEs Relevance

Users were also asked to rank the CEs by their level of relevance. The results are shown in Figure 7-6, which presents the CE and their associated weights: *High*, *Medium* and *Low*. For example, the CE *availability* has a high relevance for ranking experts according to 58% of the participants, 22% answered that the *availability* has a medium relevance in the experts ranking, while 20% stated that its relevance is low.

7.2.2 Context Management Design

The next activity in the Context Process is to design the architectural elements related to the three main context management concerns: acquisition, processing and dissemination (Section 4.2). Since ICARE is the only context consumer in this project, we did not consider the Design Dissemination Module activity. The other activities are described in the following.

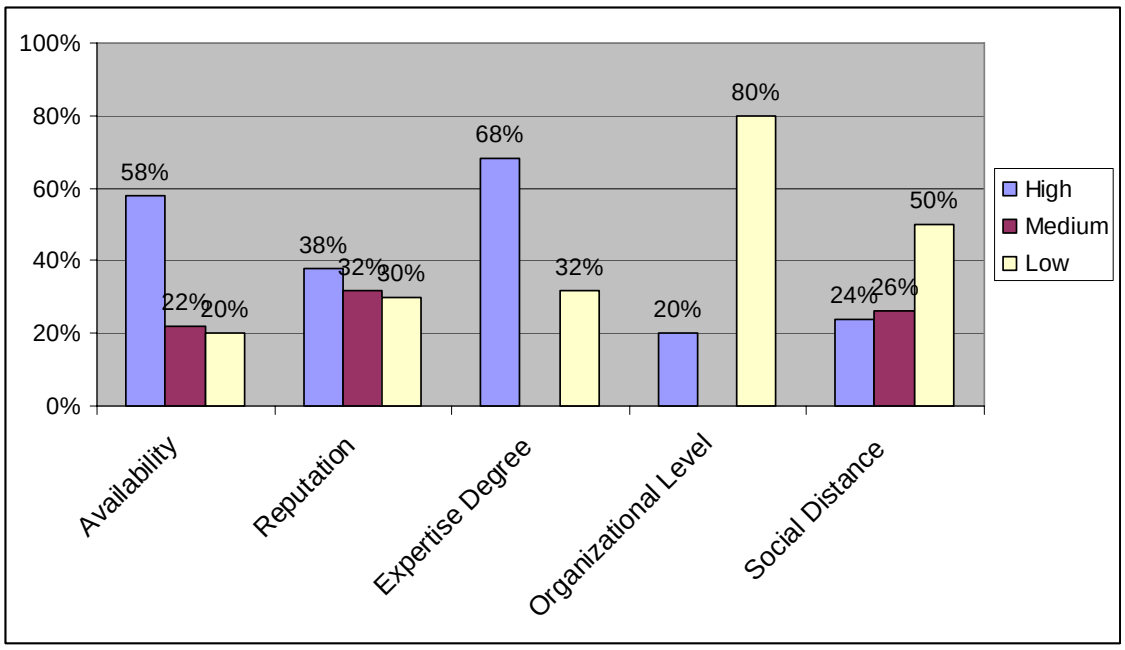

Figure 7-6 Relevance Weight Assigned to the CEs

- *Specify Context Acquisition (M1)*

To accomplish this activity we analyzed each relevant CE to verify how its acquisition should be performed.

Three external context sources were identified:

- *Lattes Database*[1]*:* is a curricula and institutions database of Science and Technology areas in Brazil;
- *GeoLite City*[2]: is an open source database that supports the identification of country, state/*region*, city, latitude and longitude information for IP addresses worldwide;
- *Windows Live Messenger (MSN)* [3]: is an instant messenger application that enables a person to connect instantly to other people. It informs the person's current status (*busy*, *available*, *away*, *offline*), current activity (*chatting*) and people s/he knows (*friends contact list*).

Two internal context sources were defined:

[1] http://lattes.cnpq.br/english/index.htm
[2] http://www.maxmind.com/app/geolitecity
[3] http://get.live.com/messenger/

- *User Profile*: is a form filled by users when they register to ICARE. It collects information about users' communication contacts, professional information and subjects of interest;
- *History Cases*: is a historical base containing previous cases of recommendation in ICARE. Each case contains the provided keywords, the returned experts list and a feedback provided by the user indicating the usefulness of the recommendation.

The acquisition parameters for each CE are summarized in Table 7-1, showing the corresponding context source, acquisition type and update periodicity. Different sources can provide values for the same CE. For example, the CE *interest* can be acquired from the Lattes database but, eventually, the user can update this information in her/his profile.

Table 7-1 Context Acquisition Parameters for ICARE

Contextual Element	Context Source	Acquisition Type	Update Frequency
currentLocation	GeoLite	Sensed	Often
availability	MSN	Sensed	Often
currentActivity	MSN	Sensed	Often
knows	MSN	Queried	Occasionally
socialDistance	MSN	Derived	Occasionally
expertise	Lattes	Queried	Occasionally
expertiseDegree	Lattes	Derived	Occasionally
interest	Lattes	Queried	Occasionally
	User Profile	Profiled	Occasionally
contactInfo	User Profile	Profiled	Occasionally
worksIn	User Profile	Profiled	Occasionally
organizationalLevel	User Profile	Profiled	Occasionally
approachability	History Cases	Derived	Occasionally
reputation	History Cases	Derived	Occasionally

- *Design Acquisition Module (M2)*

The objective of this activity is to design the interaction between ICARE and its context sources. To access the external context sources the following decisions were made:

- *Lattes Database:* we identified a tool, presented in [Ribeiro Jr, 2005], that allows the identification of a person's interests and expertise by

applying information retrieval techniques over specified fields in the person's Lattes curriculum;

- *GeoLite City*: this database provides a lookup API[1] that receives an IP address as input and returns the corresponding location information as output. We specified an adapter to translate the location information from the *GeoLite City* format into ICARE expected format;
- *WhatIsMyIP*[2]: since the *GeoLite City* expects an IP in order to provide a location, it was necessary to identify a context source that could provide this information. We designed, then, an adapter for the *WhatIsMyIP* service;
- *Windows Live Messenger (MSN)*: the Messenger API Type Library[3] is a set of interfaces for objects related to the MSN client that expose events and enable to query information from a MSN client. We designed an Adapter to manipulate information from MSN clients using the Messenger API.

The specification for ICARE's context sources is illustrated in Figure 7-7, using the UML Class Diagram notation. The figure also includes the *ICARE's* internal context sources (*User Profile* and *History Cases*). We used the design pattern Façade [Gamma et al., 1995], which allows to isolate the internal functionalities of the context manager from the context sources, thus each context source can change without impacting its usage in the context manager. The communication between each context source and the context manager is done using the façade (*FacadeContextSource*). A supertype was defined (*ContextSourceAdapter*) to uniformize the design of each particular context source adapter. The *FacadeContextSource* is composed by one or more *ContextSourceAdapter*. The adapter for each context source is an instantiation of the *ContextSourceAdapter* class.

[1] http://www.maxmind.com/app/java
[2] http://whatismyip.com/automation/n09230945.asp
[3] http://msdn.microsoft.com/en-us/library/ms630961(VS.85).aspx

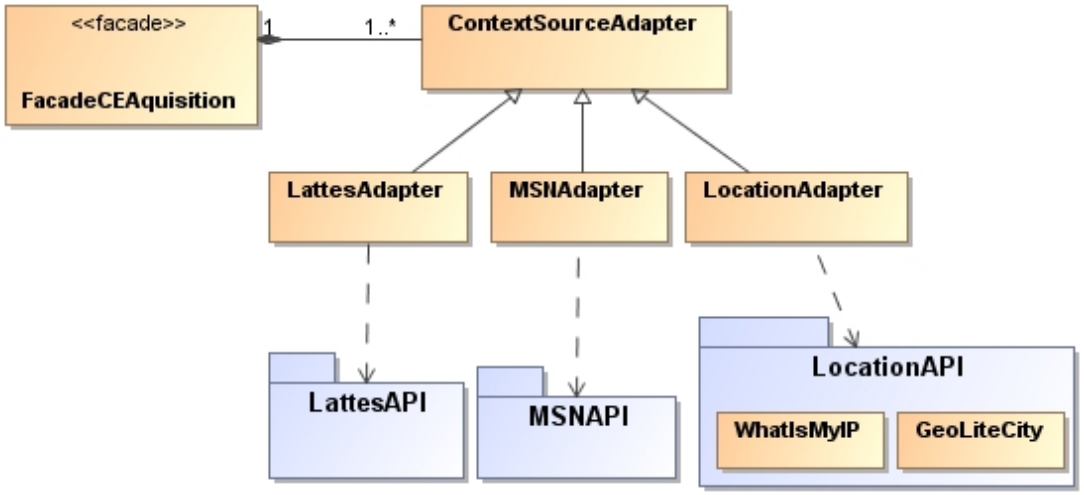

Figure 7-7 UML Class Diagram for CE Acquisition in ICARE

- *Design Processing Module (M3)*

ICARE uses the JEOPS (Java Embedded Object Production System) inference engine [Figueira Filho and Ramalho, 2000] to process the information in the Contextual Elements Knowledge Base (CEKB). JEOPS considers production rules using a forward chaining inference mechanism. It is used to process the CEs and to adjust the relevance weights assigned to the CEs. Every time ICARE receives a recommendation request, the User's CEs are asserted to the CEKB. Jeops then infers and sets the appropriate weights to be considered for the Expert's CEs. A sample rule is the following:

```
<conditions>
     User.organizationalLevel < 0.5
     User.availability < 0.3
<actions>
     Expert.expertiseDegreeWeight = 0.8
     Expert.socialDistanceWeight = 0.2
```

This rule indicates that if the user occupies a low position in the organization and is not very accessible, the recommendation should favor experts with higher expertise degree giving less importance to the social distance.

To identify the contextual rules that assign the relevance weights, we used a data mining software called Weka[1]. The questionnaire data (collected in activity S4 – *Verify CEs Relevance*) was used as input to Weka. The algorithm *Farthest First* was employed to identify association patterns (called clusters) between the users CEs and their perception about CEs relevance (the CE relevance weight). The identified clusters were transformed into contextual rules to express these associations. The contextual rules follow the pattern:

```
<conditions>
     contextual condition
<actions>
     CE relevance weights setting
```

The ICARE's Processing Module specification is illustrated in Figure 7-8, using the UML Class Diagram notation. The RankingWeights class contains the relevance weights for the CEs considered in the experts ranking. The *Retriever* class is the responsible for activating the inference engine, asserting the User CEs, and to rank the experts.

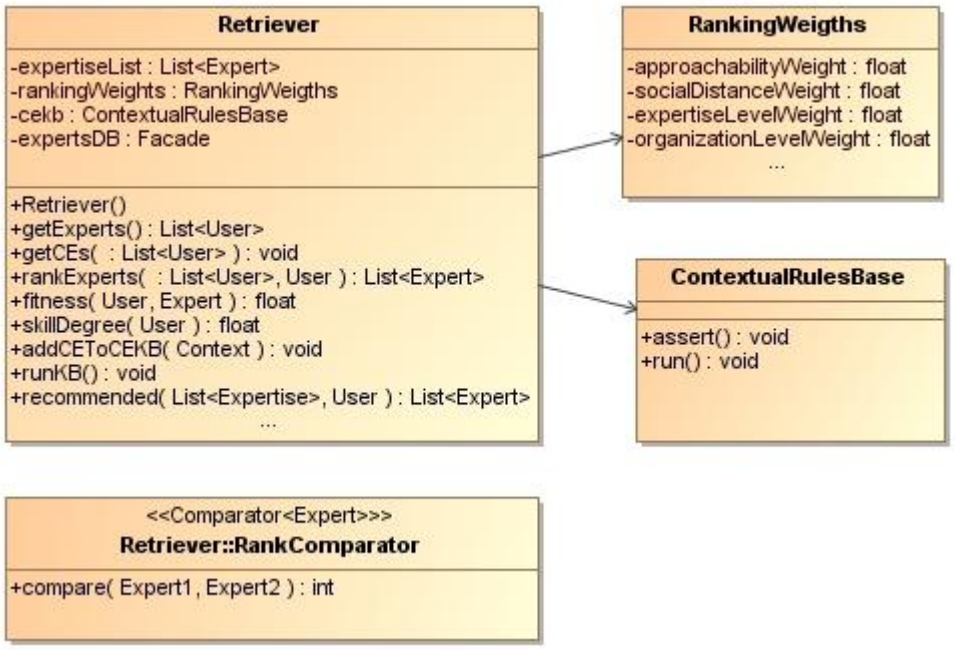

Figure 7-8 UML Class Diagram for CE Processing in ICARE

[1] http://www.cs.waikato.ac.nz/ml/weka/

7.2.3 Context Usage Design

This section describes how the context usage was designed for ICARE. It discusses the activities Design Context Behavior Model, Design Context Adaptation and Design Context Presentation, as defined in the Context Process.

- *Design Context Behavior Model (U1)*

Figure 7-9 presents the contextual graph correspondent to the focus <User, Search Experts> in ICARE. Context is used to change the relevance weight associated to the CEs used to rank experts. According to the rules identified in activity M3 (Section 7.2.2), the conditions are associated to the CEs *User.availability* and *User.organizationalLevel*.

- *Design Context Adaptation (U2)*

Context is used in ICARE to improve the classification of experts. ICARE adapts the returned experts list by changing the experts' classification according to the *fitness formula*:

$$Fitness(e,u) = \frac{\alpha_1 \times ed_e + \alpha_2 \times (ap_e + av_e) + \alpha_3 \times rep_e}{\alpha_4 \times socialDist(e,u) + \alpha_5 |OL_u - OL_e|}$$

Where:
α_i = relevance weight for each CE;
ed_e = expert's expertise degree;
ap_e = expert's approachability;
av_e = expert's availability;
rep_e = expert's reputation;
socialDist (e, u) = Social distance between the expert and the user;
$|OL_u - OL_e|$ = Difference between the user's (OL_u) and the expert's organizational level (OL_e)

This formula separates the elements that are directly proportional from those that are inversely proportional to the expert's fitness for the recommendation. For instance, the higher an expert's reputation is, the higher her fitness. On the other side, the lower the social distance between the expert and the user is, the higher the expert's fitness.

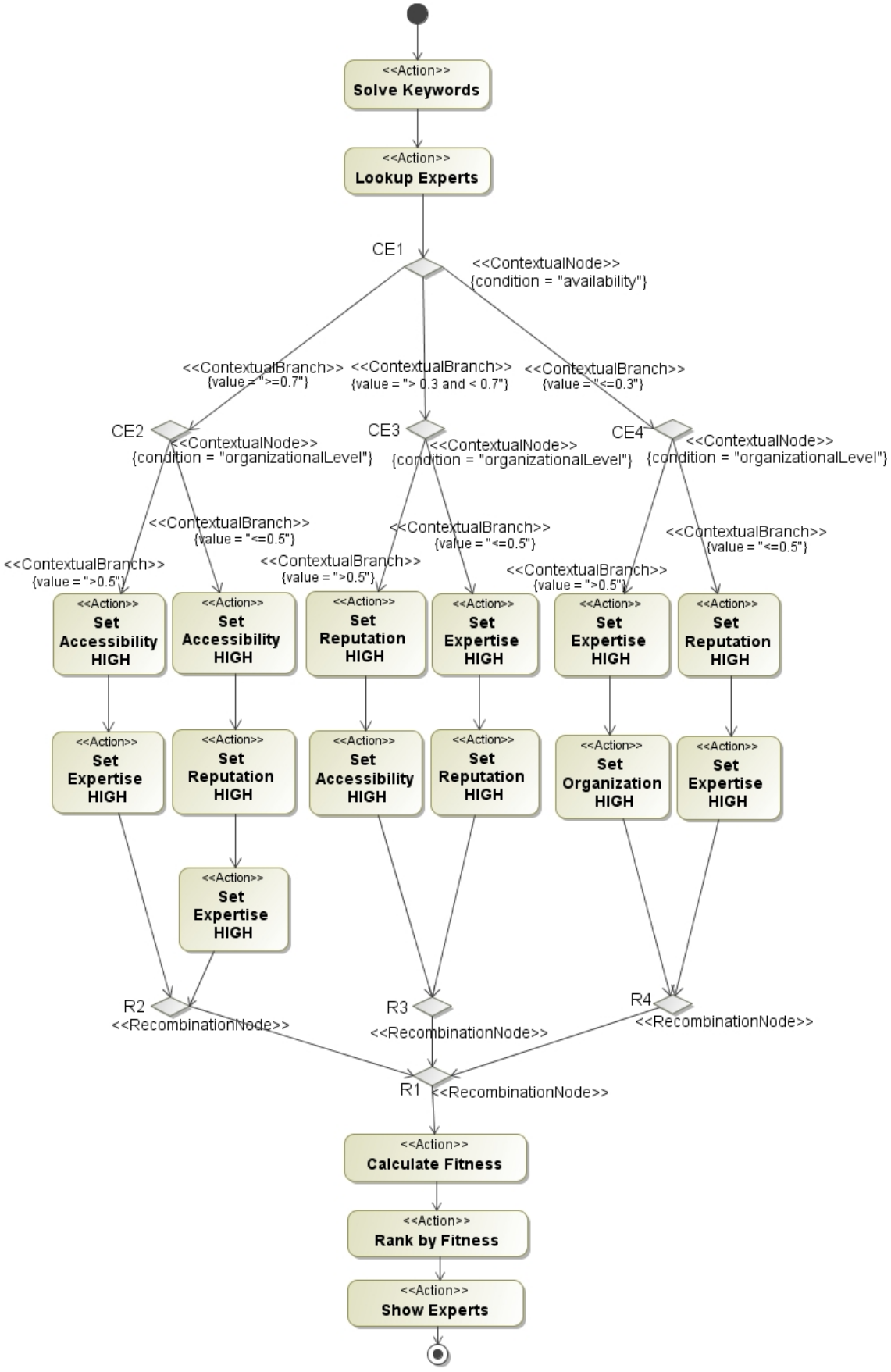

Figure 7-9 Contextual Graph for the Focus Search Experts

To make the expert recommendation fit better the user's expectation, each CE is associated to its corresponding relevance weight. The relevance weights are adjusted using the contextual rules defined in activity *M3 - Design Processing Module*. The fitness measure is computed at run time based on the CEs associated to each expert and to the user asking for the recommendation.

To improve usability and further recommendations, the user can provide feedback about the received recommendation. This feedback will compose a RecommendationCase, which stores the following values:

1) A timestamp with the date/time of the performed search;
2) An identification of the user who is performing the search;
3) A list of provided keywords;
4) A list of identified expertises (ontological terms);
5) A classified list with the returned experts;
6) An indicative of the user's feedback for that search.

The history of recommendation cases can be used to support further searches and to refine the relevance weights contextual rules and the fitness formula. The learning through recommendation cases is planned to be developed in further versions of ICARE.

- *Design Context Presentation (U3)*

Context is also used in ICARE to increase the user's cognition about the recommended experts. We believe that the perception about the appropriateness of an expert may change from user to user. In this sense, if ICARE provides contextual information about the experts, users themselves can make their own identification of which experts better fit what they need.

The following CEs were identified to be presented to the users along with the expert's name and her/his communication contacts: subjects of expertise and corresponding expertise degrees, reputation, availability, organizational level, social distance (from the expert to the user performing the search), current activity and current location.

7.3 ICARE Prototype

A prototype of ICARE was developed to implement the designed functionalities and to serve as a proof of concepts for the elements specified in the CEManTIKA framework. This section describes some implementation issues related to the prototype and the results of its evaluation with final users.

7.3.1 Implementation Issues

To populate an initial Experts Base we processed a set of curricula from researchers and developers in the Computer Science domain using the tool presented in [Ribeiro Jr, 2005]. This tool receives a version in XML of a person's Lattes curriculum and generates a profile for that person containing the identified interests and expertises for that person according to her/his Lattes curriculum. The tool uses information retrieval techniques to extract the interests and expertises from a set of fields in the curriculum. In ICARE, the following fields were used:

1) Title of the undergraduate project;
2) Title of the master degree thesis;
3) Title of the PhD thesis;
4) Name of the working area;
5) Title of the research projects;
6) Title of published papers.

The ICARE client interface is illustrated in Figure 7-10. In the upper part, the user can type the keywords to demand the recommendation. A lateral box enables the user to know the values for the CEs as considered by ICARE. In the case that any CE value is incorrect, the user can update this information before performing the search. The search results are presented in a list. The user can choose an expert from the list to see additional information about her/him. The CEs associated to that expert is shown in the interface lower part.

The prototype of ICARE was developed using the Java language [Java, 2007]. To solve the keywords it uses a domain ontology written in RDF (Resources Description Framework) [Klyne and Carroll, 2004]. This ontology is

manipulated using the Jena framework [Jena, 2006]. The context processing was implemented using the JEOPS [Figueira Filho and Ramalho, 2000] inference engine. Data used by ICARE are persisted in the relational database MySQL 5[1].

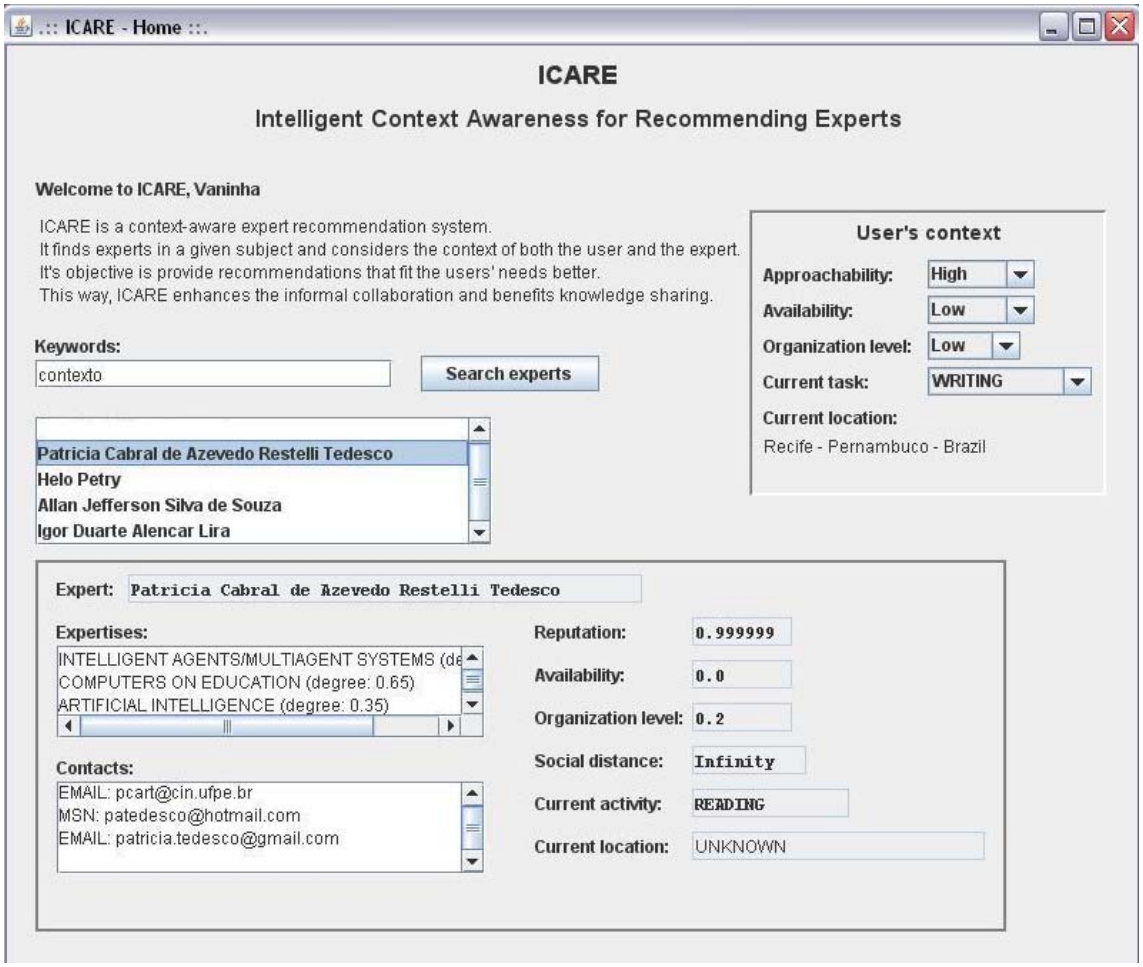

Figure 7-10 ICARE Interface with the parameters used in the Recommendation

7.3.2 Evaluation of ICARE Prototype

With the objective to verify the acceptance of ICARE and the adequacy of its recommendations, we performed two experiments with a total of 46 participants. Participants were asked to inform their current context and to execute five recommendations. Afterwards, they answered a questionnaire that aimed at evaluating the aforementioned goals. The results found on these experiments are summarized in the following.

[1] http://www.mysql.com/

When asked if they found relevant the CEs used to rank experts, about 80% answered “yes”. The participants that answered “no” specified that, in their opinion, the irrelevant elements were: social relationship, organizational level and expertise degree. About 80% of the participants agreed that the weights of the CEs should vary.

When asked if ICARE has identified adequate ontological concepts according to the informed keywords, 60.4% answered “yes”, 26% answered “no” and 23.6% of them said that the concepts were correctly identified, but incorrectly ordered.

When asked if the recommended experts were adequate, 50.5% of the participants agreed with the received recommendation and considered that the experts’ curricula were consistent with the executed request. Amongst the participants that disapproved the recommended experts, 27.1% of them also classified the resolution of the keyword into ontological concept as inadequate (in the same recommendation). Therefore, the system could not find the correct experts if the expertise was not identified correctly first.

In summary, the experiment showed an acceptance of ICARE’s chosen CEs. Also, the CEs’ weights and the identified ontological concepts were, in general, adequate. Further details about the experiment can be found in [Petry et al., 2008].

7.4 Concluding Remarks

This chapter presented a case study reporting how the CEManTIKA framework elements, guided by the Context Process, were used to model a context-sensitive Expert Recommender System named ICARE. To perform this case study, we extended and adapted the proposal of ICARE presented in [Petry, 2007]. We remodelled all the system by clearly separating the part related to context manipulation from the part related to ICARE business functionalities. By doing this, we have built a more modular, easier to maintain and to evolve prototype. Another contribution of the work performed in this thesis to the ICARE prototype is the inclusion of context sources that allows the acquisition of Sensed CEs (which changes frequently).

Some lessons learned during the development of ICARE, and verified in the experiments realized with some final users, is that the application of relevance weights when classifying experts is an interesting approach. This is due to the fact that a user may find one identified contextual element more relevant than another. For instance, a user might prefer to contact the more accessible experts than the ones with good reputation. Thus, we can provide answers that are more adequate to users with different profiles since the user's context is used to adjust the contextual elements' weights in the contextual rules. The relevance weight allows considering these user's preferences in the performed search.

Next chapter presents the conclusions of this work, discussing its main contributions and perspectives for further work.

CHAPTER

8

Conclusions

Context is a concept that is starting to gain evidence in researches related to several disciplines in Computer Science. With the ever-increasing use of context in computer systems there is also an increase in the need to support designers on building their applications to include the concept of context. Although some works address specific challenges involved in developing CSS (e.g. [Hirschfeld et al., 2008, Yang et al., 2006]), most solutions are proprietary or restricted to specific application domains (particularly, to Ubiquitous Computing).

Another problem, observed in a preliminary experiment (described in the Appendix A), is that designers have difficulties to understand and define what exactly to consider as context and how to design a CSS. This is due to the lack of consensus in the literature regarding the terminology, characteristics and specificities related to context and context-sensitive systems. There is a need for solutions that guide CSS designers on performing activities related to the context specification, management and usage in an integrated and domain-independent way.

This work was motivated by these issues and had the objective to specify a framework for designing CSS in different application domains. The CEManTIKA framework is composed by four main elements: (i) a generic

context management architecture; (ii) a domain-independent *context metamodel*, which guides context modeling in different applications; (iii) a set of *UML profiles* to account for context structure and context-sensitive behavior; and (iv) a *context process* with guidelines that cover activities related to the context specification, and the design of context management and usage.

The chapter is organized as follows: Section 8.1 discusses the contributions achieved with this research; Section 8.2 presents some lessons learned during the development of this work, together with some difficulties that were encountered; Section 8.3 indicates some directions in which the presented research could be extended; and Section 8.4 presents some concluding remarks.

8.1 Thesis Contributions

Context is not a mature concept, and the community that investigates the particularities of its usage in computer systems is still quite small, and mostly associated to the Ubiquitous Computing area. The research presented in this thesis represents a step towards the definition of a terminology related to context and CSS, in a larger sense, indicating the concepts and activities involved in including context into any computer application. This work has contributions in two axes: *conceptual*, representing an advance in the literature about context and CSS; and *practical*, with the conception and specification of the CEManTIKA framework and its usage in a case study.

8.1.1 Conceptual Contributions

In the conceptual part, we carried out a review and a critical analysis of the context usage in computer systems and the proposed techniques for context representation (detailed in [Vieira et al., 2006a]). We investigated the specificities related to context and context management in order to identify the dependence relation between context and domain knowledge (discussed in {Vieira, 2007 427 /id}). Moreover, we performed a comparative study of existing approaches related to context modeling and management {Vieira, 2006 370 /id}. The produced bibliographic material can be used as a reference for further researches involving context usage in computer systems.

8.1.2 The CEManTIKA Approach

In the practical part, the principles adopted for the *CEManTIKA Framework* included the request for integrating solutions related to the static part of a CSS (i.e. the context structure modeling) and its dynamic part (i.e. dealing with context dynamics to support behavior variations). To achieve that, we proposed the integration of two different views about context. The view largely adopted in the literature of Context-Aware Computing, as stated in the Dey's definition {Dey, 2001 83 /id} and the view related to the Cognitive Science area, as indicated in the works conducted by Brézillon and colleagues {Gonzalez, 2008 589 /id}{Brézillon, 2007 590 /id}. Our challenge here was to investigate how to integrate these two views diminishing the gap between them.

This investigation produced our *working definition of context* that considers, in the conceptual part, the dynamics of context as conceived in {Brézillon, 1999 38 /id}, and in the implementation part the materialization of context into contextual elements, which are more manageable and static units of information {Vieira, 2007 427 /id}.

From the investigation about context management we observed that the context usage in computer systems can be considered from two points of view: there is a part that strongly depends on particularities of a knowledge domain and there is another part that can be considered in a domain-independent way {Vieira, 2007 427 /id}. The former is related to the *context specification* and its *usage* in a computer system, and the latter is related to the *context management*, i.e. the mechanics of manipulating contextual elements and handling the context dynamics.

8.1.3 Context Architecture

The conducted study on context management served as the theoretical basis for our classification of the tasks related to CSS design and the definition of the *Context Architecture* (Section 4.4). This architecture along with the CSS tasks classification (Section 4.2) provides the basic architectural elements to be considered when developing any CSS in any domain. The main contributions of this architecture are the generality of the approach along with a clear separation

of the elements specifically related to context management from the modules related to the application business features.

8.1.4 Context Metamodel

To support the activities that are dependent on the domain (context specification and context usage) in a domain-independent way, we investigated how to abstract the concepts related to context representation without falling into the particularities of an application. This study produced the basis for the *Context Metamodel* (formelly called generic context model as presented in [Vieira et al., 2008, Vieira et al., 2007a]).

The *Context Metamodel* supports CSS designers involved in the context specification activity on building new context models for their applications. This is due to the fact that it defines the main concepts to be considered when building a context model, identifying the relationships between those concepts. The *Context Metamodel* covers two context-related aspects of a CSS: its structural part (*Context Metamodel Structure*) and its behavioral part (*Context Metamodel Behavior*). The Context Metamodel Structure concepts were defined in this thesis, and the Context Metamodel Behavior concepts were extended from the Contextual Graphs {Brézillon, 2002 612 /id}.

The Context Metamodel Structure is grounded on the principle that contextual elements can and should be identified from the concepts already modeled in the application domain (e.g. conceptual models and requirements models). This observation simplifies the definition of context models, since it is considered as an extension of a conceptual model instead of a new application model. In this light, theories and tools that support conceptual modeling can be used when modeling context.

To support the usage of the *Context Metamodel* on the design of context models, we propose the usage of UML extensions {OMG, 2007 603 /id}. UML was chosen due to its popularity among software designers and its ability to integrate the structural and behavioral models of an application. Two *UML Profiles* were defined: the *Context Profile* (for the Context Metamodel Structure) and the *CxG Profile* (for the Context Metamodel Behavior). These profiles are lightweight extensions of UML, meaning that they can be used on

any existing UML Case tool. The context models produced using these profiles are UML-compliant, i.e. they are themselves UML models. The Context Metamodel and its corresponding UML Profiles represent an original contribution of this work.

8.1.5 Context Process

Based on the knwoledge about context and CSS acquired during the development of this research, we developed a software process to guide a CSS development team on modeling and designing context into any application. This process, called *Context Process*, covers the activities related to Context Specification, Context Management and Context Usage (as defined in our CSS tasks classification). It provides a systematic way to integrate the Context Architecture, Context Metamodel and Context Profiles when designing a CSS. It also proposes the inclusion of a new role into the CSS development team: the Context Designer.

The *Context Process* uses the SPEM specification {OMG, 2008 609 /id}, a standard notation for modeling software processes. The whole process was defined with an overall indication of the sequence of activities to be performed and a detailed description of each activity. This detailed view presents the input and the produced artifacts of each activity, indicating guidelines that should be followed to support the activity execution. The Context Process is useful both for guiding a CSS development team on designing a new application and also as a conceptual foundation to support academic teaching activities on context and CSS. The Context Process with its detailed view about context specification and CSS design activities represent a novelty in the context and CSS literature.

8.1.6 Design of Context-Sensitive Systems

To investigate the feasibility of the ideas discussed in this work, and to provide instantiation examples to support the Context Process, we conducted the design of two applications in distincts domains: a context-sensitive Expert Recommender System, named ICARE, and an Academic Mission Support System.

To analyze implementation issues involved in developing a CSS, we implemented a *functional prototype* of ICARE to evaluate the ideas identified in this thesis. This version extends the one presented in ({Petry, 2008 568 /id}{Petry, 2006 417 /id}). The extended version adapts the proposal of ICARE to the CEManTIKA framework and includes context sources to provide Sensed CEs. By comparing the two versions of ICARE we observed that we have built, in the extended version, a more modular, easier to maintain and to evolve prototype.

8.1.7 Other contributions

During the development of this work, we investigated and experimented different techniques for representing contextual information. In particular, we studied the usage of ontologies and as a result we have created a context ontology for groupware systems, presented in {Vieira, 2005 268 /id}. An architecture for manipulating this ontology and allowing its usage on groupware applications was discussed in {Vieira, 2005 269 /id}. This ontology and architecture served as a ground to the development of an undergraduate project {Zarate, 2006 315 /id}. This project extended a Collaborative Writing System {Vieira, 2005 312 /id} with context-sensitive behavior. The developed work is described in {Vieira, 2006 321 /id}.

In another study, conducted as part of an undergraduate project {Ferraz, 2006 433 /id}, we developed some desktop sensors to support the automatic acquisition of contextual information from a person's workspace.

We have, also, analyzed the applicability of the CEManTIKA approach to different application domains. In particular, we developed a study about context applied to software reuse and how CEManTIKA could be integrated to a context-sensitive reusable assets search tool. This work is presented in {Cruz, 2007 446 /id}.

8.2 Further Work

One of the preliminary objectives of our research was to provide conceptual grounds and infrastructure about context and CSS, in order to support further

and more specific researches. In this sense, this work has uncovered a myriad of problems to be solved, which are listed below:

Context Metamodel

1. *to improve the treatment of relevance between CE and Focus*. Currently, this relevance is indicated through a manually attributed weight. We believe that the relevance association between CEs and Focus is a key factor on building more sophisticated context-sensitive systems. Relevance heuristics and algorithms should be investigated to automatically infer this relevance weight. Some theories related to relevance could support this study (e.g. [Assimakopoulos, 2003, Wilson and Sperber, 2002, Surav and Akman, 1995, Post, 1969]);
2. *to develop a new modeling and implementation language to manage the concepts defined in the Context Metamodel*. In particular, we believe that the concepts of Focus, Contextual Element and Context could demand new language constructs. Related works to this subject can be found in [Hirschfeld et al., 2008, Merril, 1998];
3. *to analyze the extension of the Context Metamodel to consider additional concepts*, such as Situation [Ye et al., 2008, Akman and Surav, 1997]. We define Situation as the interpretation of a set of CEs. In this sense, a Situation should have a name, and a set of conditions related to the CEs that must be satisfied. If all conditions are satisfied the Situation is activated;
4. *to extend the Context Metamodel with other characteristics related to the contextual elements*, for example, quality attributes (e.g. precision, accuracy, freshness), temporal attributes (e.g. created in, modified in, validity) and privacy attributes.

UML Profiles

5. *to implement semantic constraints using OCL* [OMG, 2006b]. The current version of the Context Profile and CxG Profile only cover the presentation of stereotypes and tagged values, and assume that designers will follow the Context Metamodel semantic and constraints when building their context models.

Context Architecture

6. *to investigate the usage of Model-Driven Development* [OMG, 2003] techniques to enable the automatic generation of the implementation modules for context acquisition and processing from information provided in the context model (e.g. acquisition mode, update frequency, matching expression);

Context Process

7. *to specify templates for the documents* indicated in the Context Process; and *to extend the Context Process* to contemplate the activities related to implementation, testing and evaluation;
8. *to evaluate* the Context Process and the other elements of the framework *in more complex and different case studies*.

8.3 Concluding Remarks

This work investigated the specificities related to context from the Conceptual Modeling and Software Engineering perspectives. We explored the idea that it is possible to modularize the development of CSS by separating the elements related to the application business domain from the specificities associated to context manipulation. We argued that this modularization can aid the maintenance and evolution of CSS, diminishing the complexity on building these applications.

Since context is a novel and not yet mature concept, and its applicability to computer systems is not a trivial task, we believe that the proposal presented in this thesis will be incrementally improved. For this aim, it is necessary to conduct more complex projects and experiments, as well as, to investigate technologies that could support this task.

The research described here is targeted, especially, at designers of CSS, particularly those responsible for knowledge engineering, requirement analysis and architecture design.

References

1. Akman, V. Context in Artificial Intelligence: A Fleeting Overview, La Svolta Contestuale, C. Penco, ed. McGraw-Hill, 2002.
2. Akman, V., Surav, M. "The use of situation theory in context modeling", Computational Intelligence, v. 13, n. 3, 1997, pp. 427-438.
3. Assimakopoulos, S. "Context selection and relevance", In: Theoretical & Applied Linguistics Postgraduate Conference, 2003, Edinburgh, UK.
4. Ayed, D., Delanote, D., Berbers, Y. "MDD Approach for the Development of Context-Aware Applications", In: LNAI 4635, 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'07), v. 4635, 2007, pp. 15-28, Roskilde, Denmark.
5. Bardram, J. E. "The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications", In: Proc. of 3rd International Conference on Pervasive Computing, v. LNCS 3468, 2005, pp. 98-115, Munich, Germany.
6. Bazire, M., Brézillon, P. "Understanding Context Before Using It", In: 5th International and Interdisciplinary Conference, CONTEXT 2005, v. LNAI 3554, 2005, pp. 29-40, Springer Verlag, Paris, France.
7. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L. "Web Ontology Language (OWL) Reference, W3C Recommendation", In: http://www.w3.org/TR/owl-ref/, 2004, Access in 03/2008.
8. Belian, R. B. "A Context-based Name Resolution Approach for Semantic Schema Integration", D.Sc. Thesis, Centro de Informática, Universidade Federal de Pernambuco, 2008.
9. Bellotti, V., Edwards, K. "Intelligibility and Accountability: Human Considerations in Context-Aware Systems", Human Computer Interaction, v. 16, n. 2-4, 2001, pp. 193-212.
10. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M. C., Palinginis, A. "Interplay of Content and Context", In: International Conference on Web Engineering, 2004, pp. 187-200, Munich, Germany.

11. Borges, M. R. S., Brézillon, P., Pino, J. A., Pomerol, J.-C. "Dealing with the Effects of Context Mismatch in Group Work", Decision Support Systems, v. 43, n. 4, 2007, pp. 1692-1706.

12. Bouquet, P., Ghidini, C., Giunchiglia, F., Blanzieri, E. "Theories and Uses of Context in Knowledge Representation and Reasoning", Journal of Pragmatics, v. 35, n. 3, 2003, pp. 455-484.

13. Brézillon, P. "CxG Community", In: http://www.cxg.fr/, 2007a, Access in 09/2008a.

14. Brézillon, P. "Context modeling: Task model and model of practices", In: CONTEXT 2007, LNAI 4635, 2007b, pp. 122-135, Roskilde, Denmark.

15. Brézillon, P. "Representation of Procedures and Practices in Contextual Graphs", The Knowledge Engineering Review, v. 18, n. 2, 2003b, pp. 147-174.

16. Brézillon, P. "Context Dynamic and Explanation in Contextual Graphs", In: Proc. of CONTEXT'2003, v. LNAI 2680, 2003c, pp. 94-106, Stanford, California.

17. Brézillon, P. "Focusing on Context in Human-Centered Computing", Human-Centered Computing, 2003a, pp. 2-6.

18. Brézillon, P. "Task Realization Models in Contextual Graphs", In: CONTEXT 2005, LNAI 3554, 2005, pp. 55-68, Paris, France.

19. Brézillon, P., Araújo, R. M. "Reinforcing Shared Context to Improve Collaborative Work", Revue d'Intelligence Artificielle.Special Issue on "Applying Context Management", v. 19, n. 3, 2005, pp. 537-556.

20. Brézillon, P., Borges, M. R. S., Pino, J. A., Pomerol, J.-C. "Context-Awareness in Group Work: Three Case Studies", In: IFIP International Conference on Decision Support Systems (DSS-2004). Decision Support in Uncertain and Complex World., 2004, pp. 115-124, Italia.

21. Brézillon, P., Pasquier, L., Pomerol, J.-C. "Reasoning with Contextual Graphs", European Journal of Operational Research, v. 136, 2002, pp. 290-298.

22. Brézillon, P., Pomerol, J.-C. "Contextual Knowledge Sharing and Cooperation in Intelligent Assistant Systems", Le Travail Humain, PUF, Paris, v. 62, n. 3, 1999, pp. 223-246.

23. Bucur, O., Beaune, P., Boissier, O. "Representing Context in an Agent Architecture for Context-Based Decision Making", In: Proc. of the International Workshop in Context Representation and Reasoning, 2005, Paris, France.

24. Bulcão Neto, R. F. "Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto", Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, 2006.

25. Bulcão Neto, R. F., Kudo, T. N., Pimentel, M. G. C. POCAp: A software process for context-aware computing, Hong Kong, China, Proc. of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2006.

26. Bulcão Neto, R. F., Pimentel, M. G. C. "Toward a Domain-Independent Semantic Model for Context-Aware Computing", In: Proc. of the 3rd IW3C2 Latin American Web Congress, IEEE Computer Society, 2005, pp. 61-70, Buenos Aires, Argentina.

27. Bunningen, A. "Context Aware Querying - Challenges for data management in ambient intelligence", University of Twente, TR-CTIT-04-51, 2004.

28. Byrne, P. "Awareness Mechanisms in Groupware Systems", M.Sc. Dissertation, University of Dublin, 2004.

29. Chaari, T., Ejigu, D., Laforest, F., Scuturici, V. "A comprehensive approach to model and use context for adapting applications in pervasive environments", The Journal of Systems and Software, v. 80, 2007, pp. 1973-1992.

30. Chalmers, M. "A Historical View of Context", Computer Supported Cooperative Work (CSCW), v. 13, n. 3-4, 2004, pp. 223-247.

31. Chedrawy, Z., Abidi, S. S. R. "Case Based Reasoning for Information Personalization: Using a Context-Sensitive Compositional Case Adaptation Approach", In: IEEE International Conference on Engineering of Intelligent Systems, 2006, pp. 1-6, Islamabad, Pakistan.

32. Chen, H. "An Intelligent Broker Architecture for Pervasive Context-Aware Systems", PhD Thesis, Faculty of the Graduate School of the University of Maryland, 2004.

33. Chen, H., Finin, T. "An Ontology for Context-Aware Pervasive Computing Environments", The Knowledge Engineering Review, v. 18, n. 3, 2004, pp. 197-207.

34. Chen, P. "The Entity Relationship Model - Toward a Unified View of Data", ACM Transactions Database Systems, v. 1, n. 1, 1976, pp. 9-36.

35. Cheverst, K., Mitchell, K., Davies, N. "Design of an Object Model for a Context Sensitive Tourist GUIDE", Computers and Graphics, v. 23, n. 6, 1999, pp. 883-891.

36. Christopoulou, E., Goumopoulos, C., Zaharakis, I., Kameas, A. "An Ontology-based Conceptual Model for Composing Context-Aware Applications", In: Sixth International Conference on Ubiquitous Computing (Ubicomp 2004), Workshop on "Advanced Context Modelling, Reasoning and Management", 2004, Nottingham, England.

37. Constanza, P., Hirschfeld, R. "Language constructs for context-oriented programming: an overview of ContextL", In: Proc. of the 2005 Symposium on Dynamic languages, 2005, pp. 1-10, San Diego, California.

38. Costa, P. D. Architectural Support for Context-Aware Applications - From Context Models to Services Platforms, Enschede, The Netherlands, CTIT Ph.D.-Thesis Series, No. 07-108, Telematica Instituut Fundamental Research Series, No. 021, 2007.

39. Cruz, E., Vieira, V., Almeida, E. S., Meira, S., Salgado, A. C., Brézillon, P. "Modeling Context in Software Reuse", In: 4th International Workshop on Modeling and Reasoning in Context, v. Computer Science Research Report #112, 2007, pp. 89-102, Roskilde, Denmark.

40. de Bra, P., Aroyo, L., Chepegin, V. "The Next Big Thing: Adaptive Web-Based Systems", JoDI - Journal of Digital Information, v. 5, n. 1, 2004.

41. Degler, D., Battle, L. "Knowledge Management in Pursuit of Performance: the Challenge of Context", Performance Improvement Journal (EPSS Special Edition), v. 39, n. 6, 2000, pp. 25-31.

42. Degler, D., Battle, L. "Can Topic Maps describe context for enterprise-wide applications?", In: Extreme Markup Languages Proceedings, 2003, Montreal, Quebec.

43. Desmet, B., Vallejos, J., Constanza, P., Meuter, W., D'Hondt, T. "Context-Oriented Domain Analysis", In: CONTEXT 2007, LNAI 4635, 2007, pp. 178-191, Roskilde, Denmark.

44. Dey, A. K. "Providing Architectural Support for Building Context-Aware Applications", PhD Thesis, Georgia Institute of Technology, 2000.

45. Dey, A. K. "Understanding and Using Context", Personal and Ubiquitous Computing, v. 5, n. 1, 2001, pp. 4-7.

46. Dey, A. K., Salber, D., Abowd, G. D. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human Computer Interaction Journal, v. 16, n. Special Issue on Context-Aware Computing, 2001, pp. 97-166.

47. Dourish, P. "What we talk about when we talk about context", Personal and Ubiquitous Computing, v. 8, n. 1, 2004, pp. 19-30.

48. Farias, C. R. G., Leite, M. M., Calvi, C. Z., Pessoa, R. M., Pereira Filho, J. G. "A MOF Metamodel for the Development of Context-Aware Mobile Applications", In: Proc. of ACM SAC'07, 2007, pp. 947-952, Seoul, Korea.

49. Ferrara, A., Ludovico, L. A., Montanelli, S., Castano, S., Haus, G. "A Semantic Web ontology for context-based classification and retrieval of music resources", ACM Transactions on Multimedia Computing, Communications and Applications, v. 2, n. 3, 2006, pp. 177-198.

50. Ferraz, J. "Aquisição de Contexto Baseada em Agentes em um Gerenciador de Contextos Aplicado a Sistemas Colaborativos ", Trabalho Final de Graduação, Centro de Informática - UFPE (in portuguese), 2006.

51. Ferscha, A., Holzmann, C., Oppl, S. "Context Awareness for Group Interaction Support", In: Proc. 2nd ACM International Workshop on Mobility Management and Wireless Access, ACM 2004, 2004, pp. 88-97, Philadelphia, PA, USA.

52. Figueira Filho, C., Ramalho, G. "JEOPS – Java Embedded Object Production System", Lecture Notes in Computer Science, v. 1952, 2000, pp. 53-62.

53. Franklin, D. "The Representation of Context: Ideas from Artificial Intelligence", Law, Probability and Risk, v. 2, 2003, pp. 191-199.

54. Fuchs, F., Hochstatter, I., Krause, M., Berger, M. "A Metamodel Approach to Context Information.", In: PerCom Workshops 2005, 2005, pp. 8-14, Kauai Island, HI.

55. Galegher, J., Kraut, R., Egido, C. Intellectual Teamwork: Social and Technological Bases for Cooperative Work, Hillsdale, NJ: Lawrence Erlbaum Associates., 1990.

56. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing Co., Inc., 1995.

57. Garshol, L. M. "Metadata? Thesauri? Taxonomies? Topic maps! Making sense of it all", Journal of Information Science, v. 30, n. 4, 2004, pp. 378-391.

58. Gauvin, M., Bourry-Brisset, A. C., Auger, A. "Context, Ontology and Portfolio: Key Concepts for a Situational Awareness Knowledge Portal", In: Proceedings of the 37th Hawaii International Conference on System Sciences - Track 4, 2004, p. 40111b.

59. Gehlen, G., Aijaz, F., Sajjad, M., Walke, B. "A Mobile Context Dissemination Middleware", In: Proc. of Fourth International Conference on Information Technology, 2007, pp. 155-160, Las Vegas, Nevada, USA.

60. Giunchiglia, F. "Contextual reasoning", In: Proceedings IJCAI Workshop on Using Knowledge in its Context, 1993, pp. 39-49, Chambery France.

61. Gogolla, M. "Using OCL for defining precise, domain-specific UML stereotypes", In: Proc of the 6th Australian Workshop on Requirements Engineering, 2001, pp. 51-60, Sydney, Australia.

62. Gogolla, M., Sellers, B. H. "Analysis of UML Stereotypes within the UML Metamodel", In: Proc. of the 5th Conf. Unified Modeling Language, 2002, pp. 84-99, Dresden, Germany.

63. Gonzalez, A. J., Brézillon, P. "Integrating two context-based formalisms for improved representation of human tactical behavior", The Knowledge Engineering Review, v. 23, n. 2, 2008, pp. 1-21.

64. Google "Ten things Google has found to be true", In: http://www.google.com/intl/en/corporate/tenthings.html, 2008, Access in 09/2008.

65. Goslar, K., Schill, A. "Modeling Contextual Information Using Active Data Structures", In: Proceedings of the Current Trends in Database Technology - EDBT 2004, Workshops on Pervasive Information Management (PIM), v. LNCS 3268, 2004, pp. 325-334.

66. Granovetter, M. S. "The Strength of Weak Ties", The American Journal of Sociology, v. 78, n. 6, 1973, pp. 1360-1380.

67. Greenberg, S. "Context as a Dynamic Construct", Human Computer Interaction, Special Issue on Context-Aware Computing, v. 16, n. 2-4, 2001, pp. 257-268.

68. Gross, T., Prinz, W. "Awareness in Context: A Light-Weight Approach", In: Proceedings of the Eights European Conference on Computer-Supported Cooperative Work - ECSCW 2003, 2003, pp. 295-314, Helsinki, Finland.

69. Gross, T., Specht, M. "Awareness in Context-Aware Information Systems", In: Mensch & Computer - 1. Fachuebergreifende Konferenz, 2001, pp. 173-182, Bad Honnef, Germany.

70. Gruber, T. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", International Journal Human-Computer Studies, v. 43, n. 5-6, 1993, pp. 907-928.

71. Gu, T., Pung, H. K., Zhang, D. Q. "A Service-Oriented Middleware for Building Context-Aware Services", Elsevier Journal of Network and Computer Applications (JNCA), v. 28, n. 1, 2005, pp. 1-18.

72. 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, 2004, San Diego, California, USA.

73. Guarino, N. "Formal Ontology and Information Systems", In: Proc. of the International Conference on Formal Ontology in Information Systems, 1998, Trento, Italy.

74. Guizzardi, G. "Ontological Foundations for Structural Conceptual Models", PhD Thesis, University of Twente, The Netherlands, 2005.

75. Gutwin, C., Greenberg, S. "A Descriptive Framework of Workspace Awareness for Real-Time Groupware", In: Computer Supported Cooperative Work, v. 11(3-4), 2002, pp. 411-446, Special Issue on Awareness in CSCW, Kluwer Academic Press.

76. Gutwin, C., Greenberg, S., Blum, R., Dyck, J. "Supporting Informal Collaboration in Shared-Workspace Groupware", The Interactions Lab, University of Saskatchewan, Canada, HCI Technical Report 2005-01, 2005.

77. Halpin, T. "ORM - Object Role Modeling", In: http://www.orm.net/index.html, 2006, Access in 09/2008.

78. Harvel, L. D., Liu, L., Abowd, G. D., Lim, Y.-X., Scheibe, C., Chatam, C. "Context Cube: Flexible and Effective Manipulation of Sensed Context Data", In: Proc. 2nd International Conference on Pervasive Computing, LNCS 3001, 2004, pp. 51-68, Linz/Vienna, Austria.

79. Held, A., Buchholz, S., Schill, A. "Modeling of Context Information for Pervasive Computing Applications", In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002), 2002, Orlando, FL, USA.

80. Henricksen, K. "A Framework for Context-Aware Pervasive Computing Applications", PhD Thesis, School of Information Technology and Electrical Engineering, The University of Queensland, 2003.

81. Henricksen, K., Indulska, J. "Developing Context-Aware Pervasive Computing Applications: Models and Approach", Pervasive and Mobile Computing Journal, v. 2, n. 1, 2006, pp. 37-64.

82. Henricksen, K., Indulska, J. "A software engineering framework for context-aware pervasive computing", In: 2nd IEEE International Conference on Pervasive Computing and Communications, 2004, pp. 77-86, Orlando, Florida, USA.

83. Henricksen, K., Livingstone, S., Indulska, J. "Towards a Hybrid Approach to Context Modelling, Reasoning and Interoperation", In: UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, 2004, pp. 54-61, Nottingham, UK.

84. Hirschfeld, R., Costanza, P., Nierstrasz, O. "Context-Oriented Programming", Journal of Object Technology, v. 7, n. 3, 2008, pp. 125-151.

85. IEEE "IEEE Recommended Practice for Software Requirements Specifications", IEEE Std 1471, In: http://ieeexplore.ieee.org/iel4/5841/15571/00720574.pdf, 1998, Access in 09/2008.

86. IEEE "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", IEEE Std 1471, In: http://ieeexplore.ieee.org/iel5/7040/18957/00875998.pdf, 2000, Access in 09/2008.

87. Jang, S., Ko, E. J., Woo, W. "Unified User-Centric Context: Who, Where, When, What, How and Why", In: Proc. of the International Workshop ubiPCMM05, 2005, pp. 26-34, Tokyo, Japan.

88. Java "The Source for Java Technology", In: Disponível em: http://java.sun.com, 2007.

89. Jena "Jena - A Semantic Web Framework for Java", In: http://jena.sourceforge.net/, 2006, Access in 09/2008.

90. Kashyap, V., Sheth, A. "Semantic and schematic similarities between database objects: a context-based approach", The VLDB Journal, v. 5, 1996, pp. 276-304.

91. Kiniry, J. R. "Formalizing the User's Context to Support User Interfaces for Integrated Mathematical Environments", Electronic Notes in Theoretical Computer Science, v. 103, 2004, pp. 81-103.

92. Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J., Martin, H. "Context-Aware Filtering for Collaborative Web Systems: Adapting the Awareness Information to the User's Context", In: ACM Symposium on Applied Computing, 2005, pp. 1668-1673, Santa Fe, New Mexico.

93. Klemke, R. "Context Framework - an Open Approach to Enhance Organisational Memory Systems with Context Modelling Techniques", In: Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM 2000), 2000, Basel, Switzerland.

94. Klyne, G., Carroll, J. J. "Resource Description Framework (RDF): Concepts and Abstract Syntax", 2004.

95. Kobryn, C. "Modeling Components and Frameworks with UML", Communications of the ACM, v. 43, n. 10, 2000, pp. 31-38.

96. Kokinov, B. "Dynamics and Automaticity of Context: A Cognitive Modeling Approach", In: Proc. of the CONTEXT'99, 1999, pp. 200-213, Trento, Italy.

97. Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H., Malm, E. "Managing Context Information in Mobile Devices", IEEE Pervasive Computing, v. 2, n. 3, 2003, pp. 42-51.

98. Kramer, R., Modsching, M., Schulze, J., Hagen, K. "Context-Aware Adaptation in a Mobile Tour Guide", In: Proc. of the 5th International and Interdisciplinary Conference, CONTEXT 2005, LNCS3554 , 2005, Paris, France.

99. Leite, L. E., Lima, O., Filho, G., Meira, S., Tedesco, P. "Uma Arquitetura de Serviço para Avaliação de Contextos em Redes de TV Digital", In: 25th Simpósio Brasileiro de Redes de Computadores (SBRC'2007), 2007, pp. 1-13, Belém, Pará.

100. McCarthy, J. "Notes on Formalizing Contexts", In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993, pp. 555-560, San Mateo, California.

101. Merriam-Webster "Merriam-Webster Online Search: Dictionary and Thesaurus", In: http://www.merriam-webster.com/dictionary, 2008, Access in 09/2008.

102. Merril, D. "Knowledge Objects", CBT Solutions, v. March/April, 1998, pp. 1-11.

103. Mylopoulos, J. "Conceptual modeling and Telos", In: Loucopoulos, P. and Zicari, R., Conceptual modeling, databases, and CASE, chapter 2, pp. 49-68, Wiley, 1992.

104. Nguyen, J. V., Gonzalez, A. J. "Contextual Graphs for a Real World Decision Support System", In: Proc. of the FLAIRS'06, 2006, pp. 643-648, Florida, USA.

105. Ning, K., Gong, R., Decker, S., Chen, Y., O'Sullivan, D. "A Context-Aware Resource Recommendation System for Business Collaboration", In: CEC-EEE 2007, 2007.

106. Nunes, V. T., Santoro, F. M., Borges, M. R. S. "Capturing Context about Group Design Processes", In: 11th International Conference on Computer Supported Cooperative Work in Design, 2007, pp. 18-23, Melbourne, Australia.

107. Nurmi, P., Floréen, P. "Reasoning in Context-Aware Systems", Position Paper, Helsinki Institute for Information Technology, In: http://www.cs.helsinki.fi/u/ptnurmi/papers/positionpaper.pdf, 2004, Access in 09/2006.

108. OMG "UML Semantics, Version 1.1", In: ftp://ftp.omg.org/pub/docs/ad/97-08-04.pdf, 1997, Access in 07/2008.

109. OMG "MDA Guide Version 1.0.1", In: http://www.omg.org/mda/, 2003, Access in 06/2008.

110. OMG "Meta Object Facility (MOF) Core Specification v. 2.0", In: http://www.omg.org/spec/MOF/2.0/PDF/, 2006a, Access in 09/2008a.

111. OMG "Object Constraint Language (OCL) Specification v.2.0", In: http://www.omg.org/technology/documents/formal/ocl.htm, 2006b, Access in 09/2008b.

112. OMG "Unified Modeling Language: Superstructure, Version 2.1.2", In: http://www.omg.org/spec/UML/2.1.2/, 2007a, Access in 09/2008a.

113. OMG "XML Metadata Interchange (XMI)", In: http://www.omg.org/spec/XMI/2.1.1/, 2007b, Access in 09/2008b.

114. OMG "Software & Systems Process Engineering Meta-Model (SPEM) Specification v.2.0", In: http://www.omg.org/technology/documents/formal/spem.htm, 2008a, Access in 09/2008a.

115. OMG "UML Resource Page", In: http://www.uml.org/, 2008b, Access in 06/2008b.

116. Pan, W., Wang, Z., Gu, X. "Context-Based Adaptive Personalized Web Search for Improving Information Retrieval Effectiveness", In: IEEE International Conference on Wireless Communications, Networking and Mobile Computing, 2007, pp. 5427-5430, Shanghai, China.

117. Park, D., Hwang, S., Kim, A., Chang, B. " A Context-Aware Smart Tourist Guide Application for an Old Palace", In: International Conference on Convergence Information Technology, v. 00, 2007, pp. 89-94, Gyeongju, Republic of Korea.

118. Petry, H. "ICARE: Um Sistema de Recomendação de Especialistas Sensível a Contexto", Dissertação de Mestrado, Centro de Informática-UFPE, Brasil, 2007.

119. Petry, H., Tedesco, P., Vieira, V., Salgado, A. C. "ICARE: A Context-Sensitive Expert Recommendation System", In: Proc. of the Workshop on Recommender Systems (ECAI'08), 2008, pp. 53-58, Patras, Greece.

120. Petry, H., Vieira, V., Tedesco, P., Salgado, A. C. "Um Sistema de Recomendação de Especialistas Sensível ao Contexto para Apoio à Colaboração Informal", In: Proc. of

the Simpósio Brasileiro de Sistemas Colaborativos (in portuguese), 2006, pp. 38-47, Natal, RN.

121. Pomerol, J.-C., Brézillon, P. "About some relationships between Knowledge and Context", In: Proceedings of the International Conference on Modeling and Using Context (CONTEXT-01). Springer Verlag, 2001, pp. 461-464, Dundee, UK.

122. Post, P. B. "A Lifelike Model for Associative Relevance", In: Proc. of the International Joint Conference on Artificial Intelligence. Available at: http://dli.iiit.ac.in/ijcai/IJCAI-69/PDF/027.pdf, Access in 04/2008, 1969, pp. 271-280, Washington, D.C., USA.

123. Power, R. "Topic Maps for Context Management", In: Proc. of the Workshop on Adaptive Systems for Ubiquitous Computing, 2003, pp. 199-204, Dublin, Ireland.

124. Pressman, R. S. Software Engineering: A Practitioner's Approach, 6th ed., McGraw Hill, 2005.

125. Preuveneers, D., den Bergh, J. V., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., de Bosschere, K. "Towards an Extensible Context Ontology for Ambient Intelligence", In: Ambient Intelligence: Second European Symposium, EUSAI 2004, LNCS 3295, 2004, pp. 148-159, Eindhoven, The Netherlands.

126. Ranganathan, A., Al-Muhtadi, J., Campbell, R. H. "Reasoning about Uncertain Contexts in Pervasive Computing Environments", IEEE Pervasive Computing, v. 3, n. 2, 2004, pp. 62-70.

127. Ranganathan, A., Lei, H. "Using Context Information to Improve Human Communication", IEEE Computer, v. 36, n. 4, 2003, pp. 90-92.

128. Raz, D., Juhola, A. T., Fernandes, J. S., Galis, A. Fast and Efficient Context-Aware Services, Hoboken, USA, John Wiley & Sons, 2006.

129. Reichling, T., Schubert, K., Wulf, V. "Matching Human Actors Based on their Texts: Design and Evaluation of an Instance of the ExpertFinding Framework", ACM SIGGROUP Conference on Supporting Group Work, 2005, pp. 61-70.

130. Ribeiro Jr, L. C. "Definição Automática de Perfis de Usuários de Sistemas de Recomendação", Escola de Informática. Universidade Católica de Pelotas, 2005.

131. Rittenbruch, M. "Atmosphere: Towards Context-Selective Awareness Mechanisms", In: Proc. of the 8th International Conference on Human-Computer Interaction, 1999, pp. 332-328, Munich, Germany.

132. Riva, O. "A Context Infrastructure for the Support of Mobile Context-Aware Services", In: http://www.cs.helsinki.fi/u/kraatika/Courses/f4fMC/WS1/Riva.pdf, 2005, Access in 09/2008.

133. Roque, L. "Context Engineering and Modelling Challenges", In: Proc. of Workshop on Context Modeling and Decision Support, 2005, Available at: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-144/01_roque.pdf, Accessed in 02/2008.

134. Rosa, M. G. P., Borges, M. R. S., Santoro, F. M. "A Conceptual Framework for Analyzing the Use of Context in Groupware", In: Proc. of CRIWG'03, v. LNCS 2806, 2003, pp. 300-313, Springer-Verlag Berlin, Heidelberg.

135. Russell, S., Norvig, P. Artificial Intelligence – A Modern Approach, 2 ed., New Jersey, Prentice Hall, 2003.

136. Ryan, N. "ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server", In: http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html, 1999, Access in 03/2008.

137. Sacramento, V., Endler, M., Rubinsztejn, H. K., et al. "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users", ACM/IFIP/USENIX International Middleware Conference, In: Available at http://www-di.inf.puc-rio.br/~endler/paperlinks/DSOnline.htm, 2004, Access in 07/2008.

138. Santoro, F. M., Brézillon, P., Araújo, R. M. "Context Dynamics in Software Engineering Process", International Journal of Advanced Engineering Informatics., v. (to appear), 2005.

139. Sato, K. "Context-sensitive approach for interactive systems design: modular scenario-based methods for context representation.", J Physiol Anthropol Appl Human Sci., v. 23, n. 6, 2004, pp. 277-281.

140. SBC "Grandes Desafios da Pesquisa em Computação no Brasil (2006 - 2016)", Relatório sobre o Seminário realizado em 8 e 9 de maio de 2006, In: http://www.sbc.org.br/index.php?language=1&content=downloads&id=272, 2006, Access in 09/8 A.D.

141. Schilit, B., Adams, N., Want, R. "Context-Aware Computing Applications", In: Proc. of the Workshop on Mobile Computing Systems and Applications, v. 85, 1994, pp. 90-, Santa Cruz, CA.

142. Seyler, F., Taconet, C., Bernard, G. "Context Aware Orchestration Meta-Model", In: Proc. of the 3rd International Conference on Autonomic and Autonomous Systems, 2007, pp. 17-25, Athens Greece.

143. Sheng, Q. Z., Benatallah, B. "ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services", In: Proc. of the International Conference on Mobile Business, 2005, pp. 206-212.

144. Siebra, S. "Contextual analysis of users interactions in collaborative learning environments", D.Sc. Thesis, Centro de Informática, Universidade Federal de Pernambuco, 2007.

145. Simons, C., Wirtz, G. "Modeling context in mobile distributed systems with the UML", Journal of Visual Languages and Computing, v. 18, 2007, pp. 420-439.

146. Souza, D., Belian, R. B., Salgado, A. C., Tedesco, P. "CODI - A Contextual Ontology for Data Integration", In: Proc. 4th Workshop on Ontologies-based Techniques for DataBases (ODBIS), VLDB'08, 2008, Auckland, New Zealand.

147. Stefanidis, K., Pitoura, E., Vassiliadis, P. "On Supporting Context-Aware Preferences in Relational Database Systems", In: Proc. of 1st International Workshop on Managing Context Information in Mobile and Pervasive Environments, 2005, Ayia Napa, Cyprus.

148. 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, 2004, Nottingham/England.

149. Surav, M., Akman, V. "Contexts, Oracles, and Relevance", In: AAAI-95 Workshop on Formalizing Context, 1995, pp. 23-30, Boston, USA.

150. Swarts, L. "Why People Hate the Paperclip: Labels, Appearance, Behavior, and Social Responses to User Interface Agents", B.Sc. Thesis, Symbolic Systems Program, Stanford University, 2003, http://xenon.stanford.edu/~lswartz/paperclip/.

151. Tedesco, P. "Mediating Meta-Cognitive Conflicts in Group Planning Situations", PhD Thesis, The University of Leeds, Computer Based Learning Unit and School of Computing, 2001.

152. Truong, K. N., Abowd, G. D., Brotherton, J. A. "Who, What, When, Where, How: Design Issues of Capture & Access Applications", In: Proceedings of the International Conference on Ubiquitous Computing, 2001, pp. 209-224.

153. Vajirkar, P., Singh, S., Lee, Y. "Context-Aware Data Mining Framework for Wireless Medical Application", In: DEXA'2003 - Lecture Notes in Computer Science (LNCS), Volume 2736, Springer-Verlag, 2003, pp. 381-391, Prague, Tcheque, Republique.

154. Van den Bergh, J., Coninx, K. "Using UML 2.0 and Profiles for Modeling Context-Sensitive User Interfaces", In: Proc. of the MoDELS'05 Workshop on Model Driven Development of Advanced User Interfaces, 2005, Montego Bay, Jamaica.

155. van Setten, M., Pokraev, S., Koolwaaij, J. "Context-aware Recommendations in the Mobile Tourist Application COMPASS", In: Proc. 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, LNCS3137, 2004, pp. 235-244, Eindhoven, The Netherlands.

156. Vieira, V. "Gerenciamento de Contexto em Sistemas Colaborativos", Exame de Qualificação e Proposta de Tese de Doutorado, Centro de Informática-UFPE, Brasil, 2006.

157. Vieira, V. "Ariane: Um Mecanismo de Apoio à Percepção em Bases de Dados Compartilhadas", Dissertação de M.Sc., Programa de Engenharia e Sistemas - COPPE - UFRJ, 2003, Rio de Janeiro, Brasil.

158. Vieira, V., Brézillon, P., Salgado, A. C., Tedesco, P. "Towards a Generic Contextual Elements Model to Support Context Management", In: Proc. of the 4th International Workshop on Modeling and Reasoning in Context, v. Computer Science Research Report #112, 2007a, pp. 49-60, Roskilde, Denmark.

159. Vieira, V., Brézillon, P., Salgado, A. C., Tedesco, P. "A Context-Oriented Model for Domain-Independent Context Management", Revue d'Intelligence Artificiel, v. 22, n. 5, 2008.

160. Vieira, V., Lucena, B., Rocha, F. "CoWS - Collaborative Writing through Shared Spaces", In: http://www.cin.ufpe.br/~vvs/cows, 2005a, Access in 04/2006a.

161. Vieira, V., Mangan, M. A. S., Werner, C. M. L., Mattoso, M. L. Q. "Ariane: An Awareness Mechanism for Shared Databases", In: Proceedings of the X

International Workshop on Groupware, CRIWG2004, San Carlos, Costa Rica, 2004, pp. 92-104.

162. Vieira, V., Souza, D., Salgado, A. C., Tedesco, P. "Uso e Representação de Contexto em Sistemas Computacionais", In: Cesar A.C.Teixeira, Clever Ricardo G.de Farias, Jair C.Leite, and Raquel O.Prates.(Org.)., Tópicos em Sistemas Interativos e Colaborativos, pp. 127-166, São Carlos: UFSCAR, 2006a.

163. Vieira, V., Tedesco, P., Salgado, A. C. "Representação de Contextos em Ambientes Colaborativos Usando Ontologia", In: Anais do II Workshop Brasileiro de Tecnologias para Colaboração, WCSCW2005, 2005b.

164. Vieira, V., Tedesco, P., Salgado, A. C. "Towards an Ontology for Context Representation in Groupware", In: Proc. of the 11th International Workshop on Groupware, v. LNCS3706, 2005c, pp. 367-375, Porto de Galinhas, Brasil.

165. Vieira, V., Tedesco, P., Salgado, A. C., Brézillon, P. "Investigating the Specificities of Contextual Elements Management: The CEManTIKA Approach", In: CONTEXT 2007, LNAI 4635, 2007b, pp. 493-506, Roskilde, Denmark.

166. Vieira, V., Zarate, D., Tedesco, P., Salgado, A. C. "An Ontology-Based Reasoning Mechanism for Context Management in Groupware", In: www.cin.ufpe.br/~vvs/cows/2006_cows_jena.pdf, 2006b, Access in 06/2007b.

167. Wang, X. H., Gu, T., Zhang, D. Q., Pung, H. K. "Ontology based context modeling and reasoning using OWL", In: Proc. of the 1st Workshop on Context Modeling and Reasoning, 2004, Orlando, Florida.

168. Weiser, M. "The computer for the 21st century", Scientific American, v. 265, n. 3, 1991, pp. 66-75.

169. Wilson, D., Sperber, D. "Relevance Theory", In: Ward, G. and Horn, L., Handbook of Pragmatics, pp. 607-632, Oxford: Blackwell, 2002.

170. Yang, S. J. H., Huang, A. F. M., Chen, R., Tseng, S.-S., Shen, Y.-S. "Context Model and Context Acquisition for Ubiquitous Content Access in U-Learning Environments", In: Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing, v. 2, 2006, pp. 78-83.

171. Yau, S. S., Huang, D., Gong, H., Yao, Y. "Support for Situation-Awareness in Trustworthy Ubiquitous Computing Application Software", Software-Practice & Experience, v. 36, n. 9, 2006, pp. 893-921.

172. Ye, J., Coyle, L., Dobson, S., Nixon, P. "Representing and Manipulating Situation Hierarchies using Situation Lattices", Revue d'Intelligence Artificielle, v. (accepted, waiting for publication), 2008.

173. Zacarias, M., Caetano, A., Pinto, H. S., Tribolet, J. M. "Modeling Contexts for Business Process Oriented Knowledge Support", In: WissensManagement 2005, Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure, 2005, pp. 389-396.

174. Zarate, D. "Aplicação de Contexto ao CoWS, uma Ferramenta de Escrita Colaborativa", Trabalho Final de Graduação, Centro de Informática - UFPE, 2006, Recife - Brasil.

175. Zimmermann, A., Lorenz, A., Oppermann, R. "An Operational Definition of Context", In: CONTEXT 2007, LNAI 4635, 2007, pp. 558-571, Roskilde, Denmark.

176. Zimmermann, A., Lorenz, A., Specht, M. "Applications of a Context-Management System", In: Proc. of the CONTEXT-2005, v. LNCS3554, 2005a, pp. 556-569, Paris, France.

177. Zimmermann, A., Specht, M., Lorenz, A. "Personalization and Context Management", User Modeling and User-Adapted Interaction, v. 15, n. 3-4, 2005b, pp. 275-302.

APPENDIX

A

Preliminary Study

Based on the studies of Cawsey [1992] and Mark and Greer [1993], Tedesco [, 2001] argues that a *formative* evaluation of a practical system should be used in the system's development, to refine and improve different aspects, based on the feedback of the system's users. A preliminary *formative* evaluation was conducted with system designers consisting of face-to-face interviews with the purpose to refine the ideas elaborated to the CEManTIKA framework with the observation of real requirements. This section describes this experiment.

A.1 Objectives

The objectives of this preliminary experiment were:

1) to identify different real scenarios of computer systems where CEManTIKA could be applied;
2) to elicite the perception of the interviewеds about CEManTIKA applicability; and
3) to identify their view about what context is and how it could be implemented in their systems.

By doing this experiment, we expected to identify insights that could help us to refine the concepts defined in the context metamodel and the activities defined in the context process, providing means to enable its use in different domains.

A.2 Design and Execution

The experiment was performed through pre-scheduled, face-to-face, interviews, following an interview guide (shown in Appendix A – in Portuguese). Each interview took around 30 minutes. Before starting the inquest, a brief explanation about the research objectives was given. Planned questions were conducted more or less in the order presented in the guide. Two distinct sets of questions were elaborated according to the declared use of context in the system. If the system was classified as context-unsensitive, the questions were directed to investigate: potential variations in the system behavior that could be implemented; how the developer imagines that these variations could be designed; and the reasons why they were not implemented. For declared context-sensitive systems, the questions were related to investigate how designers carried out the context modeling and how the context-sensitivity functionalities were designed and implemented.

A.3 Participants

The subjects of the experiment were systems' designers involved in the development of (potentially) context-sensitive systems. We interviewed six (6) people, working on academic and commercial projects. About the participants' profile, four of them are PhD students with about ten years of experience working with computer systems. One has a master degree and 2 years of experience in system development and the sixth was an undergraduate student with three years of experience in system development.

About their knowledge about context, one declared no interest in context research and indicated having no knowledge about context applied to computer systems. Two of them declared high knowledge degree in context in theoretical terms and medium experience developing context in computer systems. The other three indicated medium knowledge about context in theory and medium experience implementing context-sensitive systems. From those who declared interest in context, two desire to apply the concept in Ubiquitous Computing systems, one in services for Digital Television, one in Recommender Systems and one in Collaborative Writing System.

A.4 Observed Results

All participants agreed that providing adaptation in the system's behavior, according to changes in the context, is a desired functionality for their systems. Three participants indicated that context was already considered in their projects as an essential requirement (we will call them Group A). The other three participants declared their systems as context-unsensitive (Group B).

Group A participants declared to use the following context sources: user profile (3[14]), device profile (2), location provider middleware (1), user dialog interface (2) and external database (1). All participants declared to have faced difficulties in developing their projects due to the lack of knowledge, understanding and experience about context and the development of context-sensitive systems. They found very difficult to identify what to be considered as context and how to model it and how to model behavior variation. They all pointed out the lack of practical projects and tools to support novel developers in building these systems. One participant declared the use of an existing middleware. Her experience is that the supporting tool is still immature and it only attends partially the problem, since the offered support is mainly on acquisition services related to location and devices' characteristics.

The interview with participants of Group B was directed to identify how they plan to include context in their systems (since in the current version context was not considered in their projects). One participant (the same that declared having no knowledge about context) indicated that it will be interesting for her system to be context-sensitive, but she showed no interest in including this feature at the present. The other two declared that they are already investigating how to extend their systems to consider context, but this is a future work on their projects. When asked about the reason why context was left out of the project scope, since they believe it is an interesting feature, the three of them stated that the reason was limitations in resources and time. This is due to the fact that context is not an essential requirement in these systems, but a desirable optional feature.

[14] Number in parentheses indicate how many people uses the indicated source.

When asked about whether it would be useful to have a framework to support the design of context manipulation in their systems, all participants from Group A and two from Group B answered yes. The participant in Group B with no knowledge about context demonstrated little interest in solutions of any kind for context handling. When people who answered yes were asked about the kind of support they wish to find in such a framework they indicated methodologies, processes and software components.

A.5 Discussion

This preliminary study gave us the opportunity to discuss with researchers and designers involved in real projects about the use of context in their applications. This feedback assured us about the need to have tools, models and methodologies to support developers in including context into their systems. We also conclude that our target audience should be a designer already involved in a context-sensitive project, where context is seen as an essencial requirement. This is due to the observation that, in general, context is seen as an optional, complementary, and expensive feature in a system. In this sense, designers do not feel the need to include these functionalities if they are not specified, from the start, as an essencial requirement. However, since all designers indicated that the adaptation to the context is an interesting feature on theis systems, we believe that as long as the implementation of context becomes easier and less expensive (e.g. with effective framework support), this scenario can change, and designers may consider the benefits of using context greater than its cost.

A.6 Interview Guide (In Portuguese)

Pesquisa sobre o Desenvolvimento de Sistemas Sensíveis ao Contexto

O objetivo desta pesquisa é identificar cenários de uso reais que permitam estabelecer padrões que apóiem a criação de um framework de suporte ao desenvolvimento de sistemas sensíveis ao contexto. Esses sistemas possuem a característica de poder adaptar seu comportamento a diferentes usuários e situações externas. Dessa maneira, pedimos que você nos apresente um sistema que seja sensível ao contexto ou que você acredite que possa se beneficiar de sensibilidade ao contexto para tornar-se mais atraente e útil aos seus usuários, e responda as perguntas a seguir.

1. Qual a sua formação/titulação?

2. Há quanto tempo você atua como profissional de informática?

3. Quais são suas áreas de interesse acadêmico e/ou profissional?

4. Qual é a finalidade do sistema em que está trabalhando?
( ) pesquisa ( ) comercial

5. Quais funções você desempenha no projeto desse sistema?
( ) Programador ( ) Analista/Projetista ( ) Gerente
( ) Consultor ( ) Pesquisador ( ) Outra ? Qual?

6. O sistema está relacionado a que domínio de conhecimento?

7. Como você classificaria o sistema (ex. sistema de recomendação, de suporte à decisão)?

8. De uma maneira geral, quais são as principais funcionalidades do sistema?

9. Em uma escala de 0 (nenhum) a 5 (experiente), qual o seu nível de conhecimento sobre sistemas sensíveis a contexto?

a. Teórico	(0)	(1)	(2)	(3)	(4)	(5)
b. Prático (participação projetos)	(0)	(1)	(2)	(3)	(4)	(5)

10. Você diria que o seu sistema é sensível ao contexto?

*Se a resposta à questão 10 for **Não**, responda as perguntas 11 a 13.*

11. Você acha importante que seu sistema possa se adaptar ao contexto dos seus usuários (ex. antecipando serviços/informações que possam ser úteis ao usuário, mudando o comportamento de acordo com diferentes situações em que o usuário se encontre)? Porque?

12. Você poderia citar algumas situações em que você imagina que o seu sistema deveria se comportar de maneira diferenciada? Que critérios dispariariam os diferentes comportamentos?

13. Porque essas funcionalidades não foram implementadas?

*Se a resposta à questão 11 for **Sim**, responda as perguntas 14 a 18.*

14. Como o contexto é usado em seu sistema? Que critérios você utiliza para determinar os diferentes comportamentos do sistema?

15. Como você captura as informações de contexto que o sistema utiliza?

( ) Usuário cadastra **Perfil** ao se inscrever no sistema
( ) Configuração de **Preferências**
( ) Monitoramento com **Sensores**. De que tipo? _____
( ) Uso de técnicas de **análise e mineração de dados**
( ) **Formulários** de coleta de informações gerados à medida que o sistema precise delas
( ) Outra? Qual? _____

16. Quais dificuldades você enfrentou ao incluir contexto em seu sistema?

17. Você gostaria de contar com um *framework* que te apoiasse no desenvolvimento de funcionalidades de adaptação em seu sistema?

( ) Sim. Porque?
( ) Depende. Do que?
( ) Não. Porque?

18. Que tipo de suporte você esperaria encontrar nesse *framework*?

( ) Modelagem Conceitual
( ) Metodologia / Processo
( ) Componentes de software
( ) Outro? Qual?

APPENDIX

B

Metamodeling and UML Profiles

This appendix gives an overview about the UML extension mechanism, called UML Profiles.

B.1 UML Profiles Definition

UML [OMG, 2008b] is a general purpose visual modeling language for specifying, constructing and documenting the artifacts of systems. It has been widely adopted by both industry and academia as the standard language for describing software systems. The UML Metamodel Specification [OMG, 2007a] provides different and interrelated concepts and diagrams to enable the definition and visualization of separated aspects of a software application. They are classified in three main categories: static application *structure* (e.g. Class Diagram and Object Diagram), general types of *behavior* (e.g. Use Case Diagram, Activity Diagram, and State Machine Diagram); and different aspects of *interactions* (e.g. Collaboration Diagram and Sequence Diagram).

However, the fact that UML is a general purpose notation may limit its suitability for modeling some particular specific domains for which specialized languages and tools may be more appropriate. To allow customized extensions for particular application domains, the UML provides a set of extension mechanisms (*stereotypes*, *tag definitions*, and *constraints*). These

customizations extends elements from the UML Metamodel and are grouped into *UML profiles*.

A *UML profile* can be used to introduce the specific terminology of a particular domain, to specialize the semantics of the UML and to add new semantics to the UML [Gogolla and Sellers, 2002]. Any specific model built upon a UML profile is still a UML model and, in consequence, it can be defined and interchanged with any existing UML modeling tool by XMI (XML Metadata Interchange) [OMG, 2007b]. This is the main benefit of constructing a metamodel as a customization of the UML metamodel. Instead of defining new notations to support the language and building new modeling tools to recognize this notation, one can benefit of the broad variety of features already available in many developed UML modeling tools.

B.2 Elements of a UML Profile

A UML profile is basically a specific kind of package that contains *stereotypes*, *tag definitions* and *constraints* [OMG, 2007a]:

- *Stereotypes*: possess a unique name, which can be used to introduce the domain specific terminology into a modeling language. It extends metaclasses of the UML metamodel and can only be applied to instances of the extended metaclasses. For example, Figure B-1 exhibits the definition of the stereotype “Persistent” that extends the metaclass “Class” to indicate classes in a UML model that should be marked as persistent classes. In the usage example, the classes “Order”, “Customer” and “Product” are identified as <<Persistent>> unlike the class “UserInterface”;

- *Tag definitions*: represent properties of a stereotype and can be used to define additional attributes, which are not provided by the extended metaclass. When a stereotype is applied to a model element, the values of the properties are referred to as tagged values. In the example in Figure B-1, *storedInFile* is a tag definition for the stereotype *Persistent*, and it assumes the value *true* for the class *Order* and the value *false* for *Product* and *Customer*;

- *Constraints*: represent additional semantic information attached to the constrained elements that indicates a restriction that must be satisfied by a correct design of the system. The modeling constraints express the so called *well formedness rules* for the profile [Gogolla, 2001]. A constraint is represented as a statement, enclosed in braces ([Vieira et al., 2007b]), in some formal language (e.g. OCL) or a natural language. OCL (Object Constraint Language) [OMG, 2006b] represents a *condition* as a boolean expression, which can be added to any model element, stating that an instance of the model element must hold that condition.

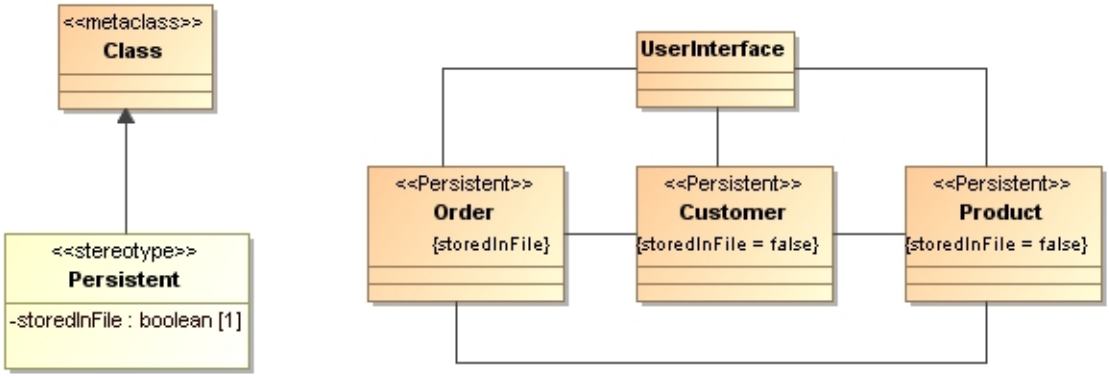

Figure B-1 Example of Stereotype Definition and Use [Gogolla and Sellers, 2002]

To be supported by existing UML based tools, a UML Profile should be defined as a lightweight extension. It means that the specialized semantics of the UML profile must not contradict the semantics defined in the UML Metamodel.