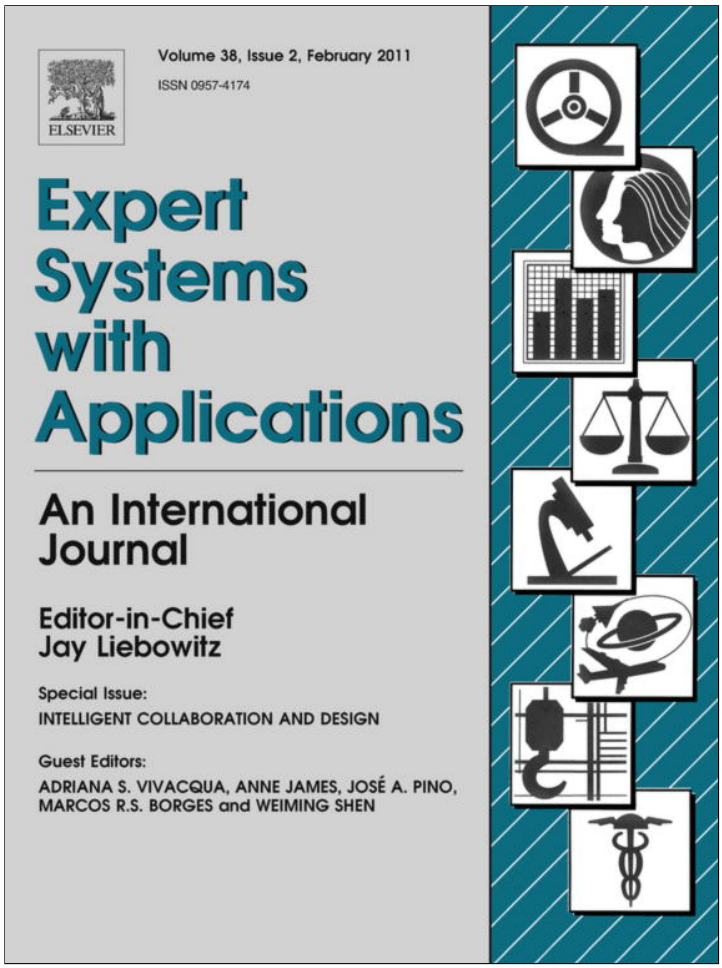

Expert Systems with Applications 38 (2011) 1119–1138

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Designing context-sensitive systems: An integrated approach

Vaninha Vieira [a,*], Patricia Tedesco [b], Ana Carolina Salgado [b]

[a] Computer Science Department, Federal University of Bahia, Salvador-BA, Brazil
[b] Center for Informatics, Federal University of Pernambuco, Recife-PE, Brazil

ARTICLE INFO

Keywords:
Context
Context-sensitive systems
Context management
Metamodeling
Design process

ABSTRACT

Context-sensitive systems (CSS) are computer systems that use context to provide more relevant services or information to support users performing their tasks, where context is any information that can be used to characterize the situation in which something exists or occurs. CSS demand that designers consider new aspects and challenges in comparison to traditional applications. In a preliminary experiment, we observed that developers find it difficult to include the concept of context in their applications. However, only few approaches offer integrated domain-independent support on developing CSS. This paper presents an integrated approach to assist the design of CSS. The originality of this work lies on the proposed way of thinking about context, on the proposed context metamodel and on the specification of a process for designing CSS. The metamodel supports building context models by making explicit the concepts related to context manipulation and by separating the context structure model from the CSS behavior model. The design process details the main activities related to context specification and the design of CSS, providing a systematic way to execute these tasks. This work also advances the state of the art related to the understanding of the concept of context (and its associated concepts). Three experimental studies were conducted to evaluate the proposal: its instantiation in the design of a context-sensitive Expert Recommender System, its usage by distinct designers on their CSS projects, and a qualitative evaluation of the overall proposal by experienced CSS developers. These studies showed a good acceptance of our approach and indicated the feasibility of using it on real projects.

1. Introduction

People are becoming increasingly dependent on computing support for making simple decisions or performing their daily tasks. This new market demand and the dynamic and information-laden environment around us impose that computer systems' developers look for solutions that make applications more attractive, adaptable and 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 filter relevant information from what is available (Kwon & Kim, 2004), to choose relevant actions from a list of possibilities (Hong, Suh, Kiim, & Kim, 2009; Chedrawy & Abidi, 2006), or to determine the optimal method of information delivery (Decouchant, Imaz, Enriquez, Mendoza, & Muhammad, 2009; Pan, Wang, & Gu, 2007). In the literature, we can find several definitions of context (Decouchant et al., 2009). According to the Merriam-Webster dictionary (Merriam-Webster, 2008), context is defined as "the interrelated conditions in which something exists or occurs". As defined in Dey, Salber, and Abowd (2001), context is any information that can be used to characterize the situation of an entity (e.g. person, place, object, user, application). Context is defined in Brézillon and Pomerol (1999) as "what constrains a step of a problem solving without intervening in it explicitly". In a broad sense, context is anything that surrounds a situation, in a given moment, and that enables to identify what is or is not relevant to interpret and understand that situation.

Computer systems that use context to provide more relevant services or information to support users performing their tasks are called context-sensitive systems (CSS). This support can be achieved by improving a user's awareness about the task s/he is performing or by providing adaptations that ease the task execution. For example, in (Hong, Suh, Kiim, et al., 2009), authors propose to use context history to identify user's preference, by automatically extracting the relationship between users' profile and services performed under the same context; thus, personalized services can be provided without users having to input their preference manually.

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.

* Corresponding author. Tel.: +55 71 3283 6342; fax: +55 71 3283 6276.
E-mail addresses: vaninha@dcc.ufba.br (V. Vieira), pcart@cin.ufpe.br (P. Tedesco), acs@cin.ufpe.br (A.C. Salgado).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.05.006

Context is dynamic and contextual information should be interpreted in order to be used. Interpretation always introduces a relevance problem, because different factors should be considered, since what is considered relevant by a person may not be considered as relevant by another one. For example, when planning a touristic trip to Europe, 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 balance relevance issues accordingly.

When designing a CSS one needs to deal with the following issues: which kind of information to consider as context, how to represent this information, how to acquire and process it (considering that it may come from several and heterogeneous sources), how to integrate the context usage in the system and how to present it since the best presentation depends on the recipient. Context is a concept that only recently started to be applied to computer systems. Hence, there is no consensus about definitions, terminologies and related concepts. Including context entails a different way of thinking about a system's engineering. In such cases, an emphasis should be given to the analysis of how users (should) interact with the CSS and how they expect the CSS to act on their behalf.

Despite the many challenges involved in building CSS, most applications handle the context-related issues without taking into account requirements such as modularity, reusability or interoperability (Riva, 2005). 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. Moreover, 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 (Ferrara, Ludovico, Montanelli, Castano, & Haus, 2006; Kramer, Modsching, Schulze, & Hagen, 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 (Fuchs, Hochstatter, Krause, & Berger, 2005; Sheng & Benatallah, 2005; Vieira, Brézillon, Salgado, & Tedesco, 2008) 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 to elaborate their own context models.

Besides, only few approaches offer integrated domain-independent support on developing CSS that combine an extensible architecture, context metamodel and design process (Ayed, Delanote, & Berbers, 2007; Bulcão Neto, 2006; Henricksen & Indulska, 2006). The analyzed approaches do not consider the reuse of existing artifacts and information already modeled and produced by application developers or domain specialists. They do not offer solutions to integrate the structural and behavioral aspects of context modeling. The behavioral aspect of a CSS is usually provided by rules, without graphical modeling support. Despite evidences that developers have difficulties to design CSS, we did not find processes that adequately indicate the activities and steps to be accomplished when designing context modeling and manipulation.

In a preliminary experiment (Vieira, 2008), we interviewed several software developers about their knowledge and understanding about context applied to computer systems. During these interviews, we observed that they have difficulties to imagine how to include the concept of context in their applications. They lack an understanding about what to consider as context, how to represent this special type of knowledge and how to design their applications to use it. These results endorse what we found in the literature and suggest that CSS designers could benefit from software engineering support (e.g. processes, metamodels, and architectural reuse) when developing their CSS.

Our research investigates the concept of context from the Conceptual Modeling and Software Engineering perspectives. 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 them. In this light, we propose 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. This paper describes the CEManTIKA approach discussing its theoretical and practical contributions.

On the theoretical perspective, we investigated the concept of context and the dynamics associated to its usage on different domains and applications. We tried to identify the specificities related to context and context management in order to establish the dependence relation between context and domain knowledge. We used this investigation to establish the principles adopted for the CEManTIKA Framework including the requirement 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). From the investigation about context management, we observed that the context usage in computer systems can be tackled from two different perspectives: there is a part that strongly depends on particularities of a knowledge domain and another part that can be considered in a domain-independent way. 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.

The practical contributions concern the specification of the framework components and their instantiation in different CSS and include: (i) the specification of a *generic architecture* describing the main architectural elements related to a CSS (Vieira, Tedesco, Salgado, & Brézillon, 2007); (ii) a domain-independent *context metamodel* (Vieira et al., 2008) that formalizes the concepts related to context manipulation and guides context modeling in different applications; (iii) a set of UML (Unified Modeling Language) *Profiles* (OMG, 2007) to account for context structure and context-sensitive behavior; and (iv) a *process* for CSS design (Vieira, Tedesco, & Salgado, 2009) with guidelines that cover activities related to context specification and CSS design. To investigate the domain-independent feature of our proposal, we designed applications in different domains. For one of those, a context-sensitive expert recommendation system, we implemented a functional prototype that was evaluated with end users.

This work also advances the state of the art related to the understanding of the concept of context (and its associated concepts). Its originality lies on the proposed way of thinking about context, on the proposed context metamodel and on the process for designing CSS. The remaining of the paper is organized as follows: Section 2 discusses the concepts related to context in computer systems and analyses existing work on context modeling and integrated approaches for supporting CSS development; Section 3 presents an overview of the CEManTIKA approach with its main elements and our working definition of context; Section 4 details the *Context Metamodel* concepts; Section 5 specifies the activities and design decisions involved in the proposed *CSS Design Process*; Section 6 discusses the studies performed to investigate the feasibility of the framework usage and the preliminary results found; and finally, Section 7 states the concluding remarks and further work.

2. Issues on context modeling and support on CSS development

Research involving context in Computer Science can 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 (Park, Hwang, Kim, & Chang, 2007; Siebra, Salgado, & Tedesco, 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 (Gonzalez & Brézillon, 2008; Hirschfeld, Costanza, & Nierstrasz, 2008; Vieira et al., 2008; Yang, Zhang, & Chen, 2008). Results from the latter are applied in researches conducted in the former. Our research is situated in this second group. This section discusses the concept of context and context-sensitive systems and presents existing proposals for integrated support on the CSS, identifying open issues that our research aims to fill.

2.1. Definitions

In Bazire and Brézillon (2005), authors have catalogued more than 150 definitions about the term *context*, and have thus concluded that the definition of what to consider as context varies strongly across 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 et al., 2001). Zimmermann, Lorenz, and Oppermann (2007), in seeking for an operational definition of context, separate the elements that characterize the situation of an entity into five 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).

These definitions see context from a static perspective as elements that can be identified and associated to existing artifacts. Another perspective in this discussion is provided by McCarthy (1993) who attempted to formalize the concept of context and identified the following issues associated to its dynamic aspect: (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. In the same sense, in Kokinov (1999) is presented 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 the concept of context, researchers agree that: (i) context exists only when related to another entity (e.g. task, agent or interaction); (ii) context is a set of items (e.g. concepts, rules and propositions) associated to an entity; and (iii) an item is considered as part of a context only if it is useful to support the problem at hand. To exemplify the last statement, consider the proposition "it is raining". It is considered as part of the context in a traffic jam support system, since rain has implications in visibility, speed and consequently in traffic. However, the same proposition is not a contextual information in a museum guide system.

Following this discussion about the static and dynamic aspects involved in context manipulation, Dourish (2004) discusses 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 emphasize objective functionalities that can be relatively easily tracked and recorded, 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 (2000) identifies some inner characteristics of the contextual information that makes its usage and manipulation difficult: 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 sources* that can be distributed and heterogeneous; *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 propose a hybrid representation, combining aspects of the representational and the interactional views, and proposing an integrated solution that considers the static and the dynamic parts of context manipulation, as we will discuss in our working definition of context (Section 3.1).

The term *context-aware system* is used to refer to systems that use context to provide relevant information and/or services to the user, where relevance depends on the user's task (Dey et al., 2001). Other terms are used as synonyms to designate these systems: *context-sensitive system* (Cheverst, Mitchell, & Davies, 1999; Sato, 2004), *context-oriented system* (Desmet, Vallejos, Constanza, Meuter, & D'Hondt, 2007) and *context-based system* (Kashyap & Sheth, 1996). We adopt the term context-sensitive system (CSS). The reasons for using this term instead of the most popular "context-aware system" are twofold. The first one is that we believe that CSS translates better the semantics of a system that *perceives* changes in its environment *and reacts* to those changes. The second one is that the latter term brings an embedded semantic association with Ubiquitous Computing applications. Since our goal in this work is to consider context usage by any system, we prefer to avoid an explicit association to a particular category of applications.

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, Adams, & Want, 1994) studies how knowledge about context may assist applications to adapt their behavior providing information and services that are closer to users' needs (Decouchant et al., 2009). Developing CSS entails more work in comparison to applications that do not consider context: in the former, one must care for context-related tasks, such as the acquisition, processing, storage and presentation of contextual information. Researchers are demonstrating increasing interest in the subjects of context and context-aware systems, as indicated in the review and classification proposed in (Hong, Suh, & Kim, 2009).

2.2. Related work

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 (Bettini et al., 2010; Strang & Linnhoff-Popien, 2004); (2) building *context models* that enumerate elements to be considered as context in a domain or a set of applications (Souza, Belian, Salgado, & Tedesco, 2008; Ye, Coyle, Dobson, & Nixon, 2008); and (3) guiding the context modeling by providing *generic context models* (Chaari, Ejigu, Laforest, & Scuturici, 2007; Gonzalez & Brézillon, 2008) and metamodels (Fuchs et al., 2005; Sheng & Benatallah, 2005; Vieira et al., 2008), which describe generic concepts related to context.

Current context-sensitive systems base their context model on existing languages, such as OWL (Ferrara et al., 2006; Wang, Gu, Zhang, & Pung, 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 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.

To support the design and development of CSS, researchers investigate the provisioning of *architectural support*, such as frameworks (Bardram, 2005; Henricksen & Indulska, 2006), middleware (Costa, 2007; Gehlen, Aijaz, Sajjad, & Walke, 2007; Gu, Pung, & Zhang, 2005; Sacramento, Endler, & Rubinsztejn, 2004) and toolkits (Dey et al., 2001; Zimmermann, Lorenz, & Specht, 2005). We observed that only few solutions have been reported on Software Engineering for CSS development that combine an integrated support on context modeling and CSS design (Ayed et al., 2007; Bulcão Neto, 2006; Henricksen & Indulska, 2006).

A Software Engineering framework to support CSS is presented in Henricksen and Indulska (2006). To support context modeling they propose a graphical modeling notation called Context Modeling Language (CML), conceived as an extension to the Object-Role Modeling (ORM). To guide the development of CSS by using their tools (CML model and architecture) they propose a methodology composed of five activities: *Analysis and specification* of context fact types; *Design* of triggering mechanisms; *Implementation*; *Customization* of the abstract models; and *Testing*. This approach is interesting since it combines the assistance of an expressive context modeling notation with a modularized architecture and it provides a set of guidelines to support the developer using these artifacts. However, although authors argue in favor of using ORM as the base notation, this is a controversial advantage since ORM is not largely used by developers. Besides, the proposed graphical notation does not have any tool to support context modeling, which makes its usage difficult in practice. The adoption of more widely used notations (e.g. UML) could improve model sharing and reuse. Also, the proposed methodology only presents the high level activities to be performed and provides a flow to indicate the sequence of execution. It does not specify details related to each activity, such as: which artifacts can be used as input or guidance to support the activity execution or which work products are produced at each activity. Moreover, they do not indicate how existing models could be reused to build context models.

Bulcão Neto (2006) proposes a Software Engineering approach to support the development of ontology-based CSS. To support context modeling they specified a generic context model composed by several ontologies responsible to provide semantic descriptions of different dimensions of the contextual information, 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. They also propose a process, based on the SPEM notation (OMG, 2008), that describes the activities related to building a CSS according to their modeling approach. Their process is an interesting proposal since it frames, organizes and details different activities and issues related to the development of CSS. Major drawbacks of this approach are: it is limited to ontology-based CSS, and it does not address the issues related to context manipulation in a broad sense. Thus, it supports challenges associated to creating and manipulating ontologies, with no special treatment for contextual information. Their context model does not consider information about the context dynamics and the CSS behavior.

Model Driven Development (MDD) (OMG, 2003) is a technique for using 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 UML Profiles. They propose a set of steps for developing CSS based on the MDD methodology. It comprises 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. Recent works in the context literature propose to use MDD and UML Profiles to assist the development of CSS (Simons & Wirtz, 2007; Van den Bergh & Coninx, 2005). One advantage of the MDD approach is that code generation follows 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.

We reviewed these approaches using the following comparative criteria: (i) *Domain-independent*, the approach should be intended for usage by different applications in different domains; (ii) *Generic Architecture*, it includes a proposal for a generic architecture for context manipulation; (iii) *Detailed Process*, it includes a proposal for a process to support CSS development, and this process details how the activities should be performed; (iv) *Structure and Behavior Modeling*, the approach includes support for modeling structural and behavioral aspects related to context manipulation; (v) *Reuse of Application Models*, it proposes the reuse of existing application models in order to reduce the complexity of modeling context; and (vi) *Implementation Support*, the proposal includes support for the CSS implementation.

The results found from this investigation are: (i) all approaches are domain-independent; (ii) a generic architecture for context manipulation is detailed only in Henricksen and Indulska (2006); (iii) a process for CSS design and implementation is detailed in Bulcão Neto (2006); however, this process contemplates the aspects mainly related to ontology manipulation instead of indicating the activities that refers to context manipulation; (iv) only the proposal presented in Ayed et al. (2007) indicates solutions for integrating the structural and behavioral aspects of context modeling; the other approaches describe solutions for modeling the structure of the contextual information; (v) the analyzed approaches do not consider the reuse of existing artifacts and information already modeled and produced by application developers or domain specialists; and (vi) the proposals presented in Henricksen and Indulska (2006) and Ayed et al. (2007) also include support for implementation aspects associated to the CSS development; implementation issues are out of the scope of this work.

Although many proposals address specific challenges on developing CSS (architectural support, representation models,

processing and reasoning techniques), there is still a lack of integrated, domain and technology independent solutions, to support the design of CSS and context modeling, considering the static and the dynamic aspects involved in context manipulation. Next section describes our proposal for a domain-independent framework, which is centered on a generic and extensible architecture for CSS, a context metamodel and a design process with guidelines to support the activities related to context specification and CSS design.

3. The CEManTIKA approach

In this research, we followed two axes: a *theoretical* and a *practical* one. In the theoretical axis, we intend to investigate the specificities associated to the concept of context in order to understand the dynamics associated to its usage and the main aspects that differentiate a context-sensitive system from traditional ones. The *practical* axis concerns the specification and instantiation of a framework that could attend three main objectives: (1) to support the design of architectural elements related to context manipulation; (2) to assist context specification and representation in a generic domain-independent manner; and (3) to guide developers on tasks to be performed to design a CSS.

This section presents the theoretical findings of our research, which include a working definition of context, and the main elements defined for our proposal for a framework, named CEManTIKA (*Contextual Elements Modeling and Management through Incremental Knowledge Acquisition*) (Vieira et al., 2008, 2009; Vieira et al., 2007), which offers an integrated support on modeling the static and dynamic aspects of context and on designing context-sensitive systems.

3.1. Our working definition of context

Our working definition of context is a combination of the two classical and complementary definitions proposed in Dey (2000) and Brézillon and Pomerol (1999) and illustrated, respectively, in the lower and upper parts of Fig. 1. The first one, proposed in Dey (2000), states that context is any information used to characterize the situation of an entity (e.g. person, place, object, application). The second one (Brézillon & Pomerol, 1999) 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), the portion of the knowledge that is relevant to the focus; *External Knowledge* (EK), the portion that has no relevance to the focus; and *Proceduralized Context* (PC), or the instantiated portion of knowledge effectively used to support the focus. The transformations from EK to CK and from CK to PC are guided by the focus.

We identified a conceptual gap between these two definitions, since Dey's definition describes context from a static point of view, similar to the representational view proposed by Dourish (2004) (discussed in Section 2.1), while Brézillon and Pomerol's definition describes the dynamic part of manipulating context, similar to Dourish's interactional view. To fill this gap, we extended Brézillon and Pomerol's definition by making the static and structured aspects of the context explicit. This extension intends to turn the context definition into a computable one. The central element of our definition is the Contextual Element (CE). We make a clear separation on the concepts of *context* and *contextual element* as described below:

> A ***contextual element*** (CE) is any piece of data or information that can be used to characterize an entity in an application 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.

As can be seen from the context 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 that compose a context have a relevance relationship with the task being performed by the agent. A CE is the atomic part of what composes the context. The main difference between context and CE is the nature of the information acquisition and definition.

Comparing to Dourish's discussion (Dourish, 2004), presented in Section 2.1, the *CE* matches his representational view. It means that a CE is a form of information that can be known, encoded and represented. It is possible to define what will be considered as a CE in advance, at design time. A CE definition is stable and we can indicate *relevance associations* between a CE and other *entities* (e.g. agent, task, device, place, environment). However, with respect to the concept of *context*, we agree with Dourish's interactional view, which indicates that what will be considered as

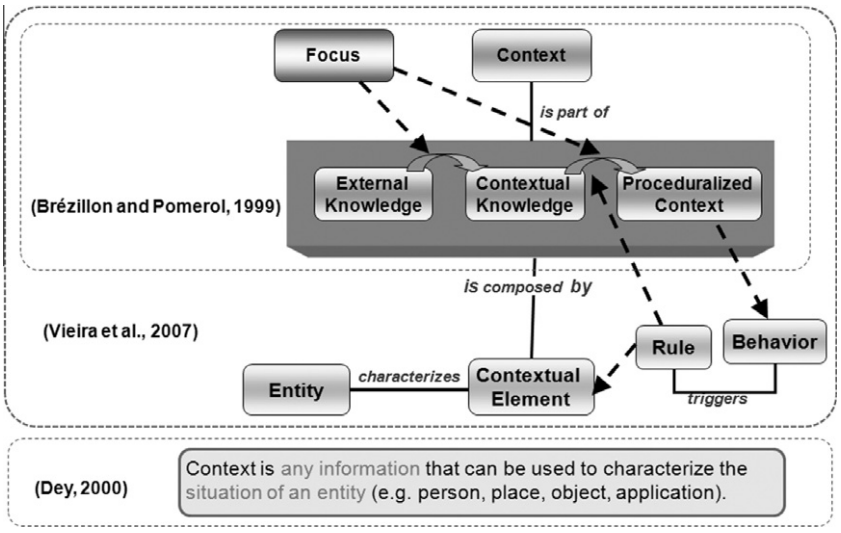

Fig. 1. Illustration of our working definition of context.

relevant in the context will be defined dynamically and will depend on a particular interaction or task execution. Context is dynamic and dependent on a focus; it must be built at runtime, when an interaction occurs. Context is the set of instantiated CEs that are necessary to support the task being performed in a given focus.

Considering Brézillon and Pomerol's definition, 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 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 CSS *triggered Behavior*.

3.2. Managing CEs and dealing with context dynamics

A CSS is an application that uses context to provide relevant information and services to its users. In this sense, a CSS project must consider tasks specifically related to context manipulation. We classify these tasks into three main categories (Vieira et al., 2008):

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.
2. *Context Management* is related to *how* context will be implemented in the system and is defined in terms of the main tasks it comprises: acquisition, storage, processing, dissemination (of CEs).
3. *Context Usage* refers to the use of the specified and managed CEs to guide the variations in the CSS behavior or the context presentation.

We consider that a CSS may be seen from two different perspectives: a part that is *domain-dependent* (context specification and context usage) and a *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 entail different considerations for their usage. On the other hand, *context management* can be modularized and domain-independent, 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., 2007). It is an intermediate layer between context sources and context consumers, and 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. A context manager circumscribes the needed mechanisms to: *acquire* CEs from multiple context sources; *process* and semantically *interpret* the acquired CEs, according to defined and learned rules; *store* the sensed and inferred CEs in a shared knowledge base; and *disseminate* the managed CEs among interested context consumers. The context consumer will use the CEs for different purposes according to its needs.

For example, a context consumer *Phone Forwarding Service* intends to use contextual information to *redirect phone calls* more appropriately, according to the user's situation (e.g. "busy", "in a meeting", "out of office"), and it expects to receive information about a user's *current location*, *availability* and *scheduled activities*. The *current location* can be acquired using a context source *GeoLocation*, the *user's availability* can be inferred from information acquired from a context source *Messenger*,[1] and the user's scheduled activities can be obtained by querying a third context source, for instance *Google Calendar*.[2] The context manager acquires information from these heterogeneous sources, processes them and delivers them to the interested context consumer (i.e. the *Phone Forwarding Service*).

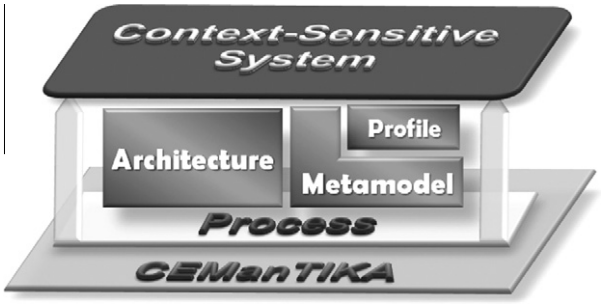

Fig. 2. The CEManTIKA framework elements.

To deal with context dynamics we adopted and extended the ideas proposed by Brézillon and Pomerol (1999) by defining four requirements to manipulate context (Vieira et al., 2007):

1. *CK Construction*: It refers to the representation of the CEs in a domain and the acquisition of CEs from different context sources.
2. *PC Building*: It aims to identify, given a focus, the set of managed CEs that are relevant to that focus and that will compose the context in that focus.
3. *Behavior Triggering*: It entails the binding of CEs to system's functionalities, through inference rules, indicating possibilities of variation in the CSS behavior according to different contexts.
4. *Incremental Knowledge Acquisition*: It is related to enhancing existing knowledge (CEs and rules) by learning new rules, facts and patterns that improve the dynamics associated to context manipulation. Machine learning and user's feedback are techniques that can be used for this purpose.

These four requirements guide the principles of CEManTIKA and are embedded in its proposed elements, which are described in the next section. The Incremental Knowledge Acquisition requirement is not supported by the current version of CEManTIKA and will be considered in further work.

3.3. Main elements of CEManTIKA framework

The CEManTIKA framework comprises comprise four main elements: a CSS reference *Architecture*; a context *Metamodel*, with associated *UML Profiles*; and a *CSS Design Process*. Fig. 2 illustrates the integration between those elements, which are described in the following.

The CSS reference *Architecture* uses the principles defined for managing CEs and dealing with context dynamics (discussed in Section 3.2) and separates a CSS into three main components: *Context Source*, *Context Manager* and *Context Consumer*, as illustrated in Fig. 3. The *Context Manager* is organized in four main

[1] http://www.windowslive.com/messenger/overview.html.
[2] http://www.google.com/calendar/.

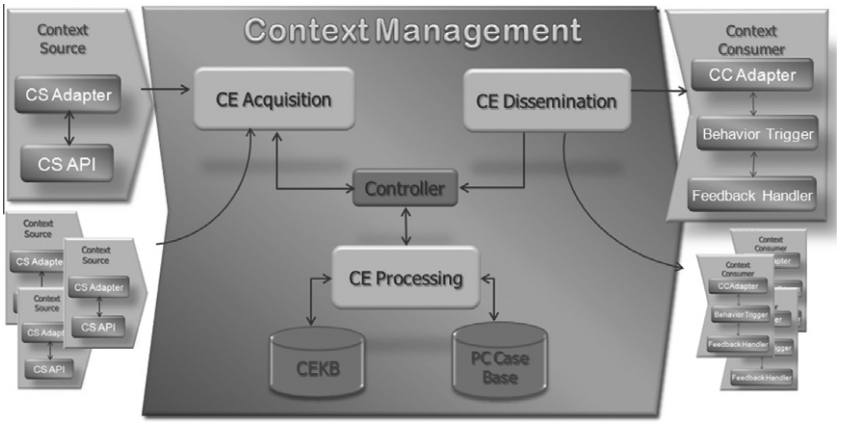

Fig. 3. CSS reference architecture overview.

modules: *Controller*, *CEAcquisition*, *CEProcessing* and *CEDissemination*. The Context Consumer includes the modules: *BehaviorTrigger* and *FeedbackHandler*. *Adapter interfaces* make the communication between the Context Manager and correspondent Context Sources and Context Consumers possible. A repository (Contextual Elements Knowledge Base – *CEKB*) stores CEs currently managed by the system, while another one (Proceduralized Context Cases Base – *PCCB*) keeps historical cases of PC building and Behavior Triggering to support the incremental knowledge acquisition requirement. The architecture separates the tasks related to context management and usage into independent and integrated modules. This division to modules intends to improve reusability and extensibility, by loosely coupling the elements and clearly separating concerns. 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.

Context models, in general, aim to specify concepts used to identify and describe situations in a domain. As discussed in Section 2.2, existing context models usually enumerate domain concepts that can be eventually considered as context in that domain (e.g. user's context, device's context, or location's context). Neither specific formalisms nor their relationship to the context dynamics is considered when structuring these elements. Moreover, much of the information modeled as "context" in existing 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, we propose a context metamodel that abstracts the concepts related to context manipulation. The metamodel is independent from specific application domains and intends to support the creation of context models. To provide support for the use of existing modeling tools, we also propose a set of UML Profiles associated to the metamodel. The context metamodel and its profiles are detailed in Section 4.

CSS demands that designers consider new aspects and challenges in comparison to traditional applications. In this sense, it is important to provide CSS developers with a process that could guide them through the fulfillment of context specific requirements. In this work, we argue that including context entails a different way of thinking about systems' 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. To support developers designing a CSS we propose the *CSS Design Process*, a process that 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, 2008). The process activities and design decisions are detailed in Section 5.

4. A context metamodel for representing structural and behavioral aspects on CSS

Modeling and metamodeling are similar activities, the difference being on the interpretation of both. A model is an abstract representation of a real-world system. A metamodel defines a language and the semantics for specifying models for particular domains or applications. We propose a Context Metamodel that defines concepts related to context manipulation on a high level domain-independent layer. The context metamodel makes explicit the context manipulation in an application and assists the specification of context models, since 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 yet 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. We separate the defined concepts in two main categories: *structure* that describes the concepts related to the conceptual and structural elements of a CSS (Section 4.2), and *behavior* that contains the concepts related to the behavioral aspects of a CSS (Section 4.3). To illustrate the concepts explanation, an example scenario is described in the next section.

4.1. An example scenario

In order to illustrate the context metamodel presentation, this section introduces a CSS 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, training period, lecture or meeting). Missions may have particular characteristics (e.g. duration, location, tasks to be accomplished). Common tasks should be performed to prior to going on a mission,

such as make a registration, request financial aid, book a hotel or transportation, and so on. A person may execute different steps and fulfill specific requirements when planning a mission. For example, in a University, the steps to be executed and the available resources (e.g. financial support) for a professor are different from those available for a student.

A system that assists users planning their missions may have features such as: (i) to identify the steps to be accomplished and requirements to be achieved according to the mission type and the user's profile; (ii) to aid the missionary on booking transport and accommodation; and (iii) to identify other concerns related to the mission that could be useful to the participant (e.g. academic activities, money exchange or tips for touristic plans). Personal history of the missionary in similar missions could also be useful (e.g. to identify preferences or previous decisions). External systems may also provide useful information, such as a hotel booking service.

4.2. Context metamodel structure concepts

Based on our working definition of context as well as on our theoretical findings on context dynamics (discussed in Sections 3.1 and 3.2), we defined the main concepts related to the structural aspects of a CSS as being (Fig. 4): `ContextualEntity`, `ContextualElement`, `ContextSource`, and Focus.

`ContextualEntity`: An entity represents a concrete representation of a real world object that can be both distinctly identified and relevant to describe a domain. It is used to classify individuals with similar characteristics and contains descriptions of these individuals through encapsulated attributes. Examples of entities in the academic mission domain are: *Person*, *Professor*, *Student*, *Mission* and *Hotel*. 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 `CE`.

`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 domain associated to the entity *Student* include: *age*, *sex*, *academicDegree*, *advisor* and *livingLocation*. For the entity *Mission*, examples of CEs are: *location*, *duration*, *type*, *startDate* and *endDate*. `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` are 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.

`Focus`: enables determining what should be considered in a context. When someone is performing some task or activity we say that this person's focus is that task. In Brézillon and Pomerol (1999), focus is defined as "a step in executing a task, in solving a problem, or in making a decision". We extend this definition 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. Thus, a `Focus` 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 system, we can identify that an agent

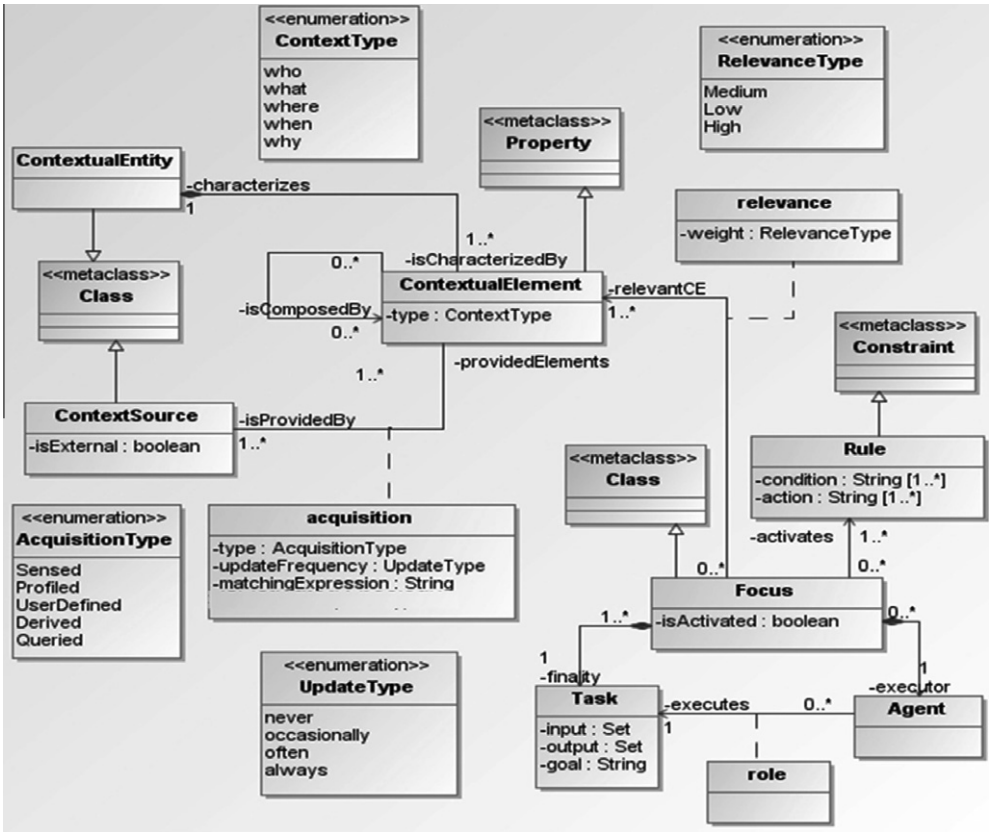

Fig. 4. The Context Metamodel main concepts.

Professor may perform some tasks, such as *Request Financial Aid* or *Book a Hotel*. Each tuple <agent; role; task>constitutes a distinct Focus.

ContextSource: One characteristic of CSS is that context information may originate from heterogeneous and frequently external context sources (e.g. user dialog interfaces, profiles, physical sensors, logical sensors and external databases). 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 CE. For example, the CE *location* may be provided by a GPS device (for outdoor places), a badge identification service (for indoor places) or by an IP (Internet Protocol) locator service (for network connections). Other information can be specified such as the acquisition type (e.g. *Sensed*, *Profiled*, *User Defined*, *Queried* and *Derived*) or the update periodicity for a CE value (e.g. *never*, *occasionally*, *often* and *always*).

4.3. CSS behavior metamodel

The *CSS Behavior metamodel* is related to the dynamic aspect of context manipulation, that is the identification of CEs related to a Focus (CERelevance) and the rules that indicates how the system should act according to context variations (Rule).

CERelevance: 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* can be affected by different agents performing a 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*. However, another agent, for example a *Professor*, could prefer a more comfortable hotel even if it is not the cheapest one. In the Context Metamodel (Fig. 4), this can be indicated through the relevance association between a Focus and a CE and a corresponding relevance weight that can be dynamically updated.

Rule: when processing a CE or identifying the behavior of a CSS, it may be necessary to consider associated rules (e.g. production rules). In the Context Metamodel, 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 (e.g. to *trigger* a system's behavior, to *assign* a new value to a CE, or to *assign* a new relevance weight to an association between a CE and a Focus).

To support modeling behavior concepts, the metamodel proposes the usage of the formalism of *contextual graphs* (Brézillon, 2007). Fig. 5 illustrates a *contextual graph* for a task *Book Transportation* in the academic mission domain. The condition associated to the node CE1 ("*Mission.occursIn = Person.livesIn*") verifies if the mission is carried out in the same city where the person lives. Two paths can be followed according to the allowed values for CE1: yes or No. If CE1.value = "*yes*" no transport is necessary and thus the contextual graph points to the end of the task execution. Otherwise, it will activate the contextual node CE2, which verifies the value of the CE *Mission.whoPays*. In this case, two branches are considered. In the first case (CE2.value = "*CAPES*"), the activated action is "*Contact CAPES Official Agency*", and then the end of the task execution. In the second case (CE2.value = "*missionary*"), an action "*Lookup Transport*" is triggered and, in the sequence, another contextual node CE3 is examined. CE3 verifies the value for the CE *Person.age*. Three branches are identified. For example, the condition (value = "*< 26*") triggers the action "*Classify by Price*" and leads to the next action to be executed "*Show Transport Options*" and, finally, conducts to the final node, indicating the end of the task. Each contextual node has an associated recombination node (labeled R1, R2 and R3, respectively), indicating the convergence of different paths opened by a contextual node. For example, the condition verified by CE3 ends in node R3.

Each path in the contextual graph contains the definition of a production rule that indicates the context influence over the application behavior. Considering the contextual graph of Fig. 3, examples of rules are presented below.

```
Rule 1:
   Conditions
      not (Mission.occursIn = Person.livesIn)
      Mission.whoPays = "CAPES"
   Actions
      CallBehavior ("Contact CAPES Official Agency")
Rule 2:
   Conditions
      not (Mission.occursIn = Person.livesIn)
      Mission.whoPays = "missionary"
      Person.age < 26
   Actions
      CallBehavior ("Lookup Transport")
      CallBehavior ("Classify by Price")
      CallBehavior ("Show Transport Options")
```

Rule 1 indicates that if a person does not live where the mission occurs, and the mission is paid by an agency (CAPES), the behavior that should be triggered is "Contact CAPES Official Agency". Rule 2 states that if a person does not live where the mission occurs, the mission is paid by the missionary and if the missionary is less than 26 years old, the following behaviors should be triggered in the sequence: "Lookup Transport", then "Classify by Price", then "Show Transport Options".

4.4. Instantiating the context metamodel with UML profiles

The *UML profile mechanism* (OMG, 2007) allows customizing the UML Metamodel for specific problem domains. 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 for each category: the structure metamodel (*Context Profile*) and the behavior metamodel (*CxG Profile*), illustrated, respectively, in Figs. 9 and 10. The Context Profile aims to support building context conceptual models based on the context structure metamodel concepts by extending and enriching the *UML Use Cases* and *Class Models* with context semantics by using stereotypes and tag definitions, as explained in Table 1. The CxG Profile allows a CSS designer to model the application behavior using the *UML Activity Model* with the semantics defined in the *contextual graphs*. The stereotypes defined for the CxG Profile are described in Table 2. 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, and we can integrate the CSS behavior model with the context conceptual model.

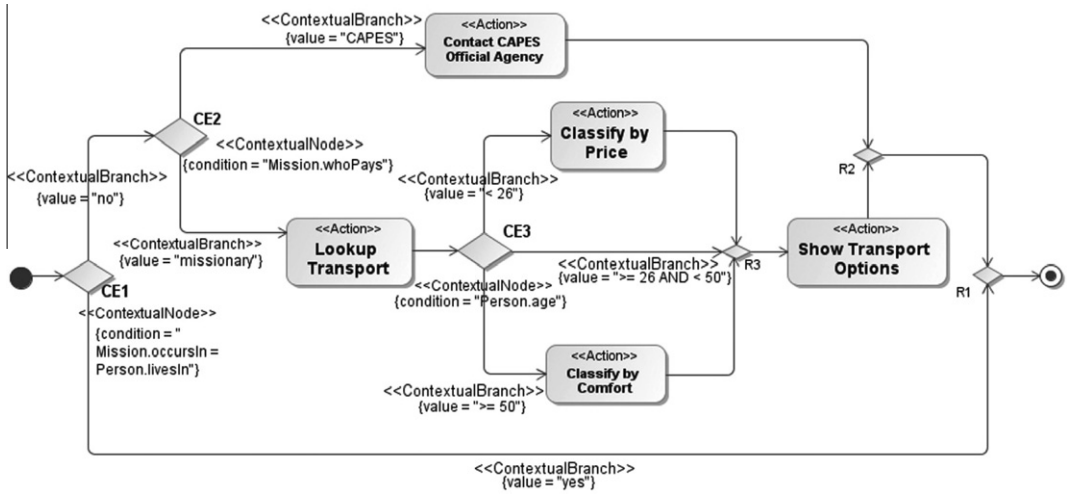

Fig. 5. Contextual graph for the `Focus` <`Person`; `Book Transportation`>.

Table 1
The context profile elements.

UML model	Stereotype	Extended metaclass	Tag definition	Description
Use cases	`<<Agent>>`	`Actor`		Actors associated to compose a `Focus`
	`<<Task>>`	`UseCase`, `Activity`		Use cases to be considered in a `Focus` composition
	`<<executes>>`	`Association`		Describe a `Focus` composition by associating an `Agent` to a `Task`
Class	`<<Contextual Entity>>`	`Class`		Classes that designate contextual entities
	`<<Contextual Element>>`	`Property`	`type`	Attributes or associations of a contextual entity that represent `CEs`
	`<<Context Source>>`	`Class`		Classes that designate context sources
	`<<acquisition Association>>`	`Association`	`element`, `acquisitionType`, `matchingExpression`, `updateFrequency`	Model the relationship between a `CE` and a `ContextSource`
	`<<Focus>>`	`Class`	`task`, `agent`	Classes in the context model that describe a `Focus`
	`<<relevance Association>>`	`Association`	`element`, `weight`	Relevance relation between a `Focus` and a `CE`, with a relevance weight
	`<<Rule>>`	`Constraint`	`action`	Behavior rules. The tag `action` represents what will be executed when all `conditions` (defined in `Constraint`) are satisfied

Table 2
The CxG profile elements as extension of the UML activity model.

Stereotype	Extended metaclass	Tag definition	Description
`<<ContextualGraph>>`	`Model`		Model package containing the contextual graph elements. Each defined `Focus` has an associated contextual graph
`<<Action>>`	`CallBehavior Action`		CxG action node
`<<ParallelAction Grouping>>`	`ForkNode`		Set of CxG action nodes that can be executed in parallel
`<<ContextualNode>>`	`DecisionNode`	`condition`	A contextual node tests the values of a `CE`. Each condition represents a situation associated to that `CE`, which can split the graph in two or more different paths
`<<RecombinationNode>>`	`DecisionNode`		Each contextual node must have a corresponding recombination node to indicate a deactivation point, meaning that the condition analysed in the contextual node is no longer needed in the graph execution
`<<ContextualBranch>>`	`ControlFlow`	`value`	Model an association from a `ContextualNode` to: another contextual node, an action or a parallel action grouping. It represents a new path in the CxG graph, according to the situation analysed in the `ContextualNode`

4.5. Discussion

This section presented our proposal for a Context Metamodel, which guides the creation of context models in a domain-independent manner. It also presented the proposed UML Profiles (OMG, 2007) related to the Context Metamodel: Context Profile (for modeling structural concepts) and CxG Profile (for modeling CSS 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. Besides, the semantics enrichment provided by the stereotypes 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 made when specifying the context metamodel concepts were based on weaknesses of the reviewed context models. We noticed that, in general, the analyzed approaches: (i) 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 to the static aspect of the contextual element; (ii) do not consider a separation between the concepts of *context* and *entity*. There are cases (e.g. Simons & Wirtz, 2007) where an entity is indicated as being a context, instead of assuming that context should be built by analyzing specific properties associated to the entities, not the entity as a whole; (iii) lack support for reusing existing application models in the construction of the 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.

Existing UML profiles for context modeling (Ayed et al., 2007; Simons & Wirtz, 2007) identify the concept equivalent to our `CE` (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, instead of considering, for example, a *Person* as a contextual element, we should identify the *person's age* or the *person's current location* as some `CEs` that characterize the entity *Person*. 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*).

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

5. CSS design process

The Software Engineering literature indicates that a software process, in general, comprises the following main phases: analysis, design, code generation, testing and maintenance (Pressman, 2005). As discussed in Section 3, the main tasks involved in CSS development are: *Context Specification*, *Context Management* and *Context Usage*. We propose a *CSS Design Process*, which details the main activities related to context specification and CSS design, providing a systematic way to execute these activities. The main

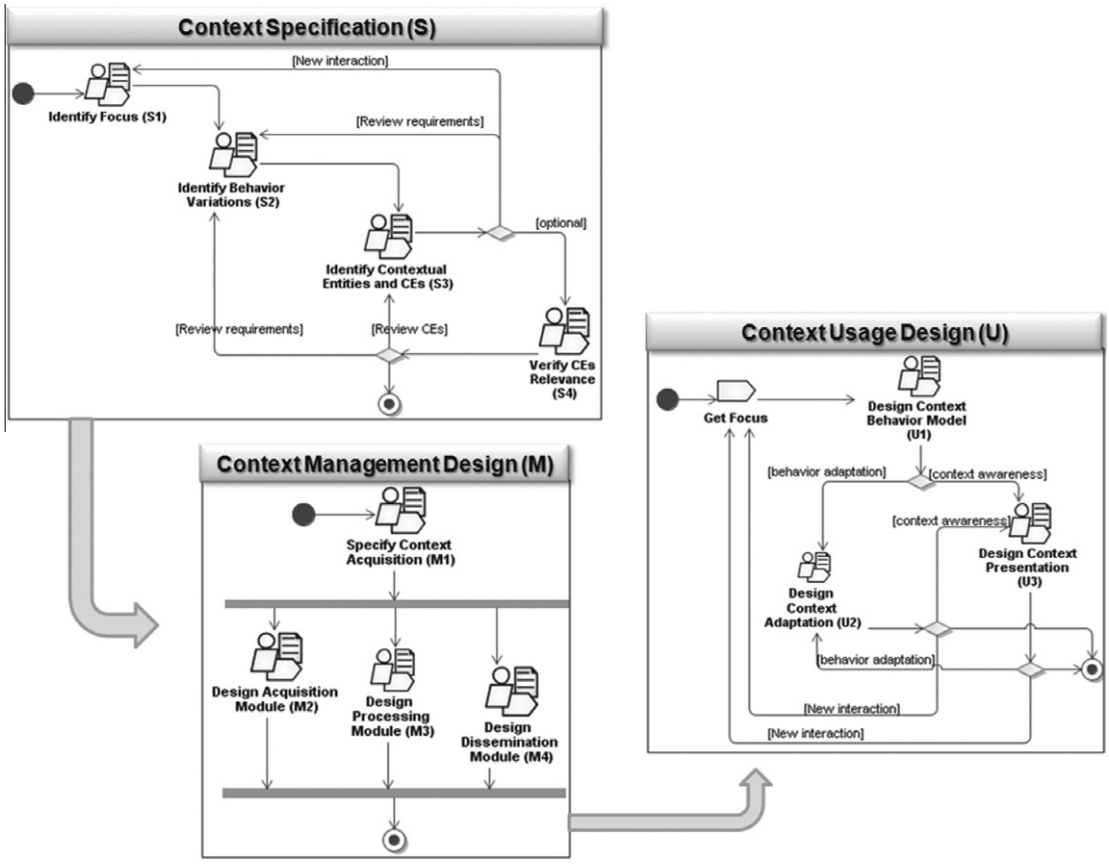

Fig. 6. CSS design process sub-activities for: Context Specification (S), Context Management Design (M), and Context Usage Design (U).

activities identified in the CSS Design 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 *CSS Evaluation* (*maintenance*). This last activity indicates that a CSS should undergo constant evaluations with end users to improve adaptation functionalities. Currently, we concentrate on extending the activities related to analysis and design (see Fig. 6). The other activities are out of the scope of this work and will be considered in the near future.

The CSS Design Process uses the notation and terminology described in the Software Process Engineering Metamodel (SPEM) (OMG, 2008), a standard for modeling software processes. Its goal is to guide a CSS development team on modeling and designing context manipulation. Three main roles are considered: (i) the *System Designer* is the person or team responsible for designing the system's architecture; (ii) the *System Analyst* is responsible for identifying users' needs and for translating business requirements into software specifications; and (iii) the *Context Designer* is responsible for identifying context-related requirements and for designing context-sensitive solutions. The context designer role demands expertise on multidisciplinary subjects related to Human Cognition, Automatic Acquisition technologies, Artificial Intelligence, Software Development and Usability. Next sections detail each main activity of the proposed design process.

5.1. Context specification

This activity has the objective to identify the context requirements based on the business requirements and to create the context conceptual model. This activity comprises four sub-activities as illustrated in Fig. 6 and described in the following:

Identify Focus (S1) aims to recognize from the business requirements which tasks and agents should be considered as foci in the CSS. It is guided by the concepts defined in the Context Metamodel using the Context Profile stereotypes. It takes as input a Use Cases Model with the main business requirements for the CSS. It produces as output an extended version of the Use Cases Model enriched with the focus identification.

Identify Behavior Variations (S2): a behavior variation indicates the different actions, related to a focus, that the CSS may execute, according to distinct contexts. S2 aims to identify, given a focus, which variations are expected in the CSS behavior and which factors affect these variations. It uses the extended Use Cases Model (from S1) and produces a Context Requirements document.

Identify Contextual Entities and CEs (S3): this activity aims to identify the entities related to the focus and the characteristics from those entities that influence each behavior variation. Fig. 7 illustrates the detailed diagram for this activity. Its inputs are the Context Requirements document (S2), and a Conceptual Model from the CSS domain. The output artifact is a Context Conceptual Model. As guidance, it uses the concepts and stereotypes defined in the Context Metamodel and Context Profile. Optionally, other guidances can support this activity, such as domain ontologies and existing context models.

Verify CEs Relevance (S4): the next step is to evaluate if the CSS end users and designers have the same understanding about the relevance of the identified CEs, and if the defined behavior variations reflect users' expectations. S4 uses as input the Context Conceptual Model (S3) and the Context Requirements document (S2), and produces 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).

These activities are all performed by the context designer in collaboration with the system analyst. The activities are executed sequentially. While executing a given activity it may be necessary to go back to a previous one.

5.2. Context management design

Once the Context Specification is defined, the context designer has to investigate how the specified CEs should be acquired, processed and disseminated to different consumers. The *Context Management Design* activity comprises three sub-activities (Fig. 6) as follows:

Specify Context Acquisition (M1): specifies the acquisition parameters for each identified CE. Its inputs are the Context Conceptual Model and the Context Requirements document (from S2 and S3, respectively). Its output is an updated version of the Context Conceptual Model and an Acquisition Configuration document. It uses the Context Metamodel and the Context Profile as guidance.

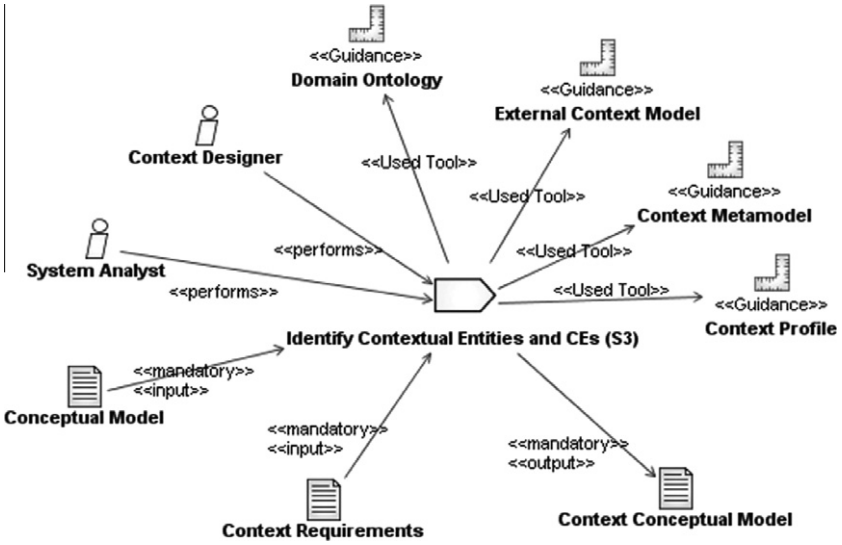

Fig. 7. Detailed view of identify contextual entities and CEs.

Design Acquisition Module (M2): defines elements responsible for the context acquisition (e.g. context sources APIs and adapters), indicating how the context acquisition should be implemented. Its input is the Acquisition Configuration document (M1) and it produces an Acquisition Module Specification. It uses the Context Architecture as guidance.
Design Processing Module (M3) defines and design the elements related to CE processing, i.e. derived CEs specification, CE knowledge base, inference rules and inference engine. Its inputs are the Context Conceptual Model (S3), the Context Requirements document (S2) and, optionally, the Context Behavior Model (from U1, discussed in Section 3.3). Its outputs are the Contextual Rules, the specification of the Processing Module elements, and an updated version of the Context Behavior Model. It uses the Context Architecture as guidance.
Design Dissemination Module (M4) defines the elements responsible for disseminating CEs among different consumers. Its inputs are the Context Conceptual Model (S3) and the Context Requirements document (S2). It produces as output the Dissemination Module specification. Its guidance is the Context Architecture.

These activities are performed by the system designer with the collaboration of the context designer. Activities M2, M3 and M4 are independent from each other and can be executed in any order.

5.3. Context usage design

We consider two main usages for context in a CSS: to support behavior adaptation and to enrich an agent's awareness with contextual information. The *Context Usage Design* activity aims to design how context is effectively used by the CSS. It comprises three activities (Fig. 6), as follows:

Design Context Behavior Model (U1) 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. Its input is the Context Conceptual Model and the Context Requirements document (produced in S3 and S2, respectively). Its output is the Context Behavior Model for the focus, identified as a contextual graph (as presented in Section 4.3). As guidance, it uses the CxG Profile defined in CEManTIKA.
Design Context Adaptation (U2) aims to specify how the CSS should adapt to the context. It uses as input the Context Conceptual Model (from S3), the Context Requirements document (from S2), and the Context Behavior Model (from U1). It generates the Adaptation Module Specification as output. To guide this activity, the designer may use specifications provided by the Context Architecture and guidelines with directives related to Adaptation and Usability Aspects.
Design Context Presentation (U3) has the purpose of designing the presentation of the managed CEs to the CSS agents in order to enrich their knowledge about the task being executed. The inputs for this activity are the Context Conceptual Model (from S3), the Context Requirements document (from S2), and the Context Behavior Model (from U1). It generates the Presentation Specification document as output. To guide this activity, the designer may use specifications provided by the Context Architecture and guidelines with directives related to Interface and Usability Aspects.

An interaction of these three activities should be performed for each focus identified in S1. Activities U2 and U3 are optional in the process, and the designer should decide which activity is necessary given the CSS requirements. These activities are performed by the system designer with the collaboration of the context designer.

6. Experimental studies

To investigate the applicability of the CEManTIKA framework, we conducted three experimental studies. The first one (described in Section 6.1) illustrates its usage by instantiating it on the design of a specific CSS. The second one (detailed in Section 6.2) aimed to study its usability by different designers on distinct in progress CSS projects. The third study (Section 6.3) was conducted to analyze the proposal acceptance by CSS designers with previous experience in conducting CSS projects.

6.1. Instantiating CEManTIKA in ICARE

The CEManTIKA proposed elements were instantiated in the design of a real project of an expert recommender system (ERS), named ICARE (*Intelligent Context Awareness for Recommending Experts*) (Petry, Tedesco, Vieira, & Salgado, 2008). 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, Schubert, & Wulf, 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. ICARE is an ERS that considers contextual information about users (who are requesting the recommendation) and experts when processing recommendations.

Fig. 8 exhibits an overview of ICARE. It maintains a *base of experts*. Users access ICARE through a *recommendation interface* and provide a set of keywords to receive a classified list of experts that correspond to the informed keywords. This is the basic functionality of ICARE (*without considering context*). To be effective, an ERS should consider the context of people involved in the recommendation (i.e. the user requesting the recommendation and the recommended expert). Context can enable ERS to provide better recommendations since it can inform, for instance, the role the user performs in the organization, the knowledge areas s/he has worked with or subjects s/he is 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, or if they have common friends that could introduce them or if user and expert have already worked together. To improve the recommendations providing experts that better match users' needs, ICARE uses a *context interface*. CEs related to the user performing the recommendation and to the recommended experts are considered in addition to the keywords. To modularize the context manipulation in ICARE we followed the sequence of activities suggested in the CSS Design Process, as described in the next subsections.

6.1.1. Context specification

The first main activity in the CSS Design Process is the Context Specification, which comprises four sub-activities: *Identify the*

Fig. 8. Overview of ICARE.

Focus (S1), *Specify Behavior Variations* (S2), *Identify Contextual Entities and CEs* (S3) and *Verify CEs Relevance* (S4). Each of them (as they were performed for ICARE) is described below.

S1: to identify the possible foci in ICARE we must analyze the possible Agents that interact with ICARE and the Tasks these agents could perform. We identified that ICARE assists two type of users: a *generic user*, who performs the recommendation and an *external system* that uses ICARE as a plugin (e.g. Text Editor or Email Reader) to automatically activate the experts search. The main task supported by ICARE is the *Experts Search*. In this sense, possible foci in ICARE are the tuples: <*User*, *Experts Search*> and <*External System*, *Experts Search*>.

S2: The conventional behavior of ICARE 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 enriched with information about each Expert CEs.

S3: the considered CEs related to the entities *User* and *Expert* (illustrated in Fig. 9) were: *availability* (how busy the user or the expert is), *knows/socialDistance* (indicates, respectively, a social relation between two people and the number of people that separates them); *currentLocation* (physical location of experts and users); *contactInfo* (informs how a person can be reached); *worksIn/organizationalLevel* (identifies, respectively, the work relation between a person and an organization and the person's position in the organization); *currentActivity* (the activity the person is currently performing); *interest* (subjects a person has interest in); *expertise/expertiseDegree* (indicates, respectively, the subjects a person has expertise and the level of expertise in the subject); *approachability* (denotes how easy it is to contact the expert); and *reputation* (points out the expert's overall quality as judged by users who contacted her/him).

S4: to evaluate the identified CEs, we conducted an experiment with 50 participants from development and research organizations. People were asked if they would consider the CEs identified in S3 when filtering and ranking experts. They were also asked to rank the CEs by relevance, according to their view. The results from this investigation (presented in Petry et al. (2008)) indicated that people agreed with the relevance of considering the *availability*, *reputation* and *expertise degree* when ranking an expert. The CE *organizational level* was considered relevant by 54% of the participants. An interesting result of this investigation refers to the CE *social distance* that was considered as irrelevant by 64% of the participants. This result contradicts what we initially thought since we believed that recommendations could be more effective if the user already knew the recommended expert or if they have common friends or

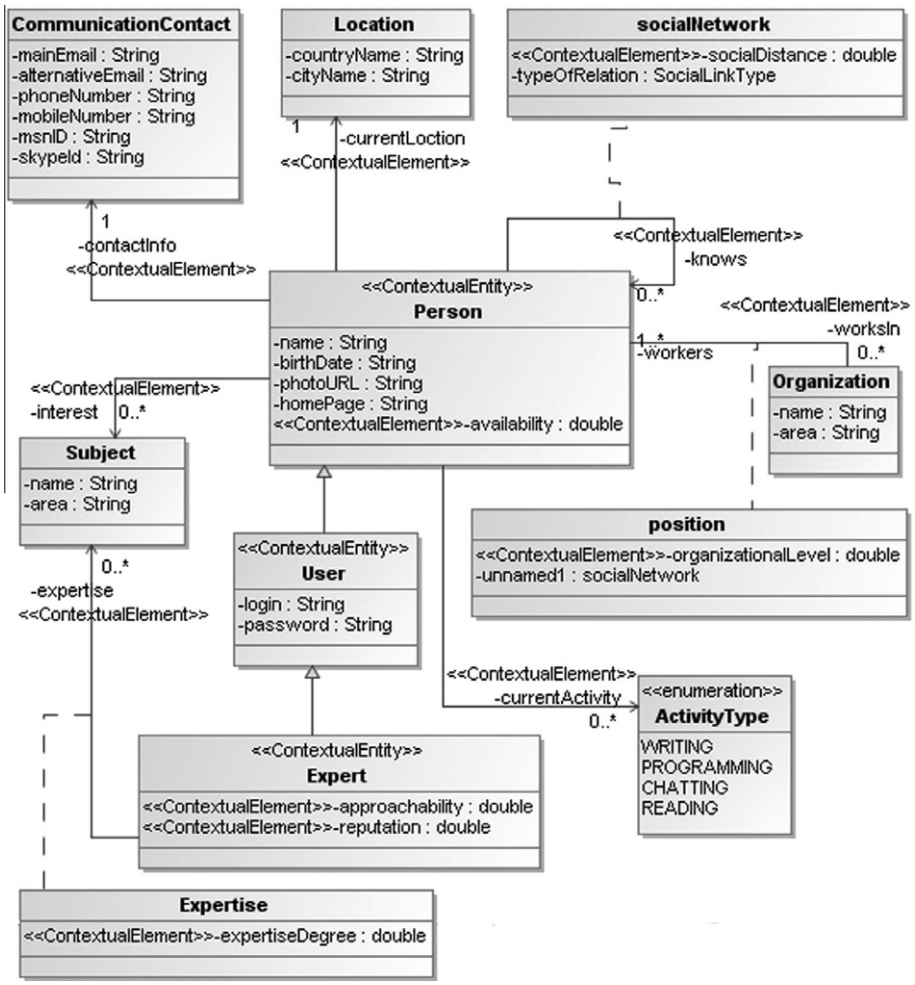

Fig. 9. ICARE conceptual class diagram enriched with context profile stereotypes and new CE definitions.

colleagues. These results showed us the importance of interacting with the system's end users when identifying the CEs and defining the context-sensitive behavior.

6.1.2. Context management design

The next activity in the CSS Design Process is to design the architectural elements related to the three main context management concerns: acquisition, processing and dissemination. These activities are described below:

M1: to model context acquisition, we identified three external and two internal context sources: *Lattes Database*[3] (a curricula database in Brazil that informs interests, expertise, previous experience, current organization and others); *GeoLite City*[4] (a database that supports the identification of country, state/region, city, latitude and longitude information for IP addresses worldwide); *Windows Live Messenger*[5] (an instant messenger application that enables a person to connect instantly to other people and informs a person's current status, current activity and people s/he knows); *User Profile* (a form filled by users when registering to ICARE that collects information about users' communication contacts, professional information and subjects of interest); and *History Cases* (a base with previous recommendation cases where each case contains the provided keywords, the returned experts list and a feedback provided by the user indicating the usefulness of the recommendation).
M2: to access the context sources external to ICARE we made the following decisions: (i)*Lattes Database*: we identified a tool, presented in Ribeiro (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; (ii) *GeoLite City*: this database provides a lookup API[6] 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; (iii) *WhatIsMyIP*:[7] 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*; (iv) *Windows Live Messenger* (MSN): the Messenger API Type Library[8] 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. To model the communication between ICARE and the context sources we used the design pattern Facade (Gamma, Helm, Johnson, & Vlissides, 1995), which allows us to isolate the internal functionalities of ICARE, thus changes in the context source do not impact its usage in ICARE.
M3: we used the JEOPS inference engine (Figueira Filho & Ramalho, 2000) to process the managed CEs and the defined contextual rules. To identify the contextual rules, we used a knowledge analysis software called Weka (Waikato, 2008). The questionnaire data (collected in S4) 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. 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.
M4: Since ICARE has only one context consumer, this activity was not performed.

The M4 sub-activity indicates the end of the Context Management Design activity and the Context Usage Design activity should be performed.

6.1.3. Context usage design

The CSS Design Process indicates three activities related to the design of context usage: Design Context Behavior Model (U1), Design Context Adaptation (U2) and Design Context Presentation (U3).

U1: in ICARE, context is used to change the relevance weight associated to the CEs used on experts ranking. According to the rules identified in M3, the conditions are associated to the CEs *User.availability* and *User.organizationalLevel*. The contextual graph was used to model the different paths associated to those CEs (Fig. 10). Each path in the graph denotes a contextual rule, as for example:

```
Rule 1:
  Conditions
    User.availability >= 0.7
    User.organizationalLevel > 0.5
  Actions
    CallBehavior ("Solve Keywords")
    CallBehavior ("Lookup Experts")
    CallBehavior ("Set Accessibility HIGH")
    CallBehavior ("Set Expertise HIGH")
    CallBehavior ("Calculate Fitness")
    CallBehavior ("Rank by Fitness")
    CallBehavior ("Show Experts")
```

U2: ICARE adapts the returned experts list by changing the experts' classification according to the defined *fitness formula* (1). This formula separates elements directly proportional from those inversely proportional to the expert's fitness for the user's search. To better fit the recommendation to the user's expectation, each CE is associated to a corresponding relevance weight. The relevance weights are adjusted using the contextual rules defined in activity M3. The fitness measure is computed at run time based on the CEs associated to each expert and to the user asking for the recommendation.

$$Fitness(e,u) = \frac{\alpha_1 \times ed_e + \alpha_2 \times (ap_e + av_e) + \alpha_3 \times rep_e}{\alpha_4 \times socialDist(e,u) + \alpha_5 |OL_u - OL_e|}, \quad (1)$$

where α_i is relevance weight for each CE; *ede*, expert's expertise degree; ap_e, expert's approachability; av_e, expert's availability; rep_e, expert's reputation; $socialDist(e,u)$, social distance (expert and user); and $|OL_u - OL_e|$ is difference between the user's (OL_u) and the expert's organizational level (OL_e).
U3: Context is also used to increase the user's awareness 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 identify which experts better fit what they need. The following CEs were selected to be presented to the users along with the expert's name and her/his contact links: subjects of expertise and corresponding expertise degrees, reputation,

[3] http://lattes.cnpq.br/english/index.htm.
[4] http://www.maxmind.com/app/geolitecity.
[5] http://get.live.com/messenger/.
[6] http://www.maxmind.com/app/java.
[7] http://whatismyip.com/automation/n09230945.asp.
[8] http://msdn.microsoft.com/en-us/library/ms630961(VS.85).aspx.

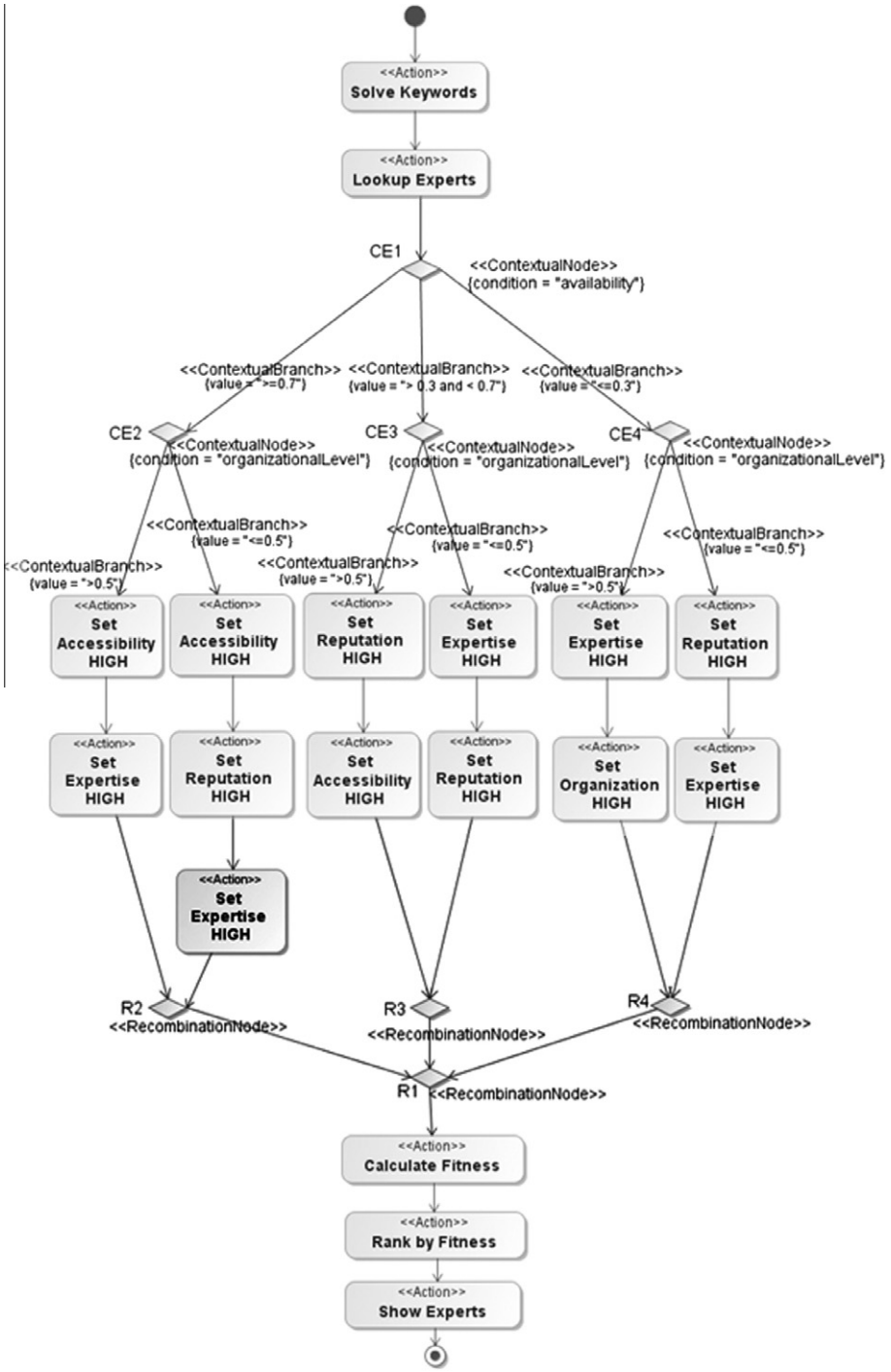

Fig. 10. ICARE context behavior model: contextual graph for the focus search experts.

availability, organizational level, social distance (from the expert to the user performing the search), current activity and current location.

Fig. 11 illustrates the implemented interface of ICARE with the indication of context usage for adaptation and presentation. In the upper part, the user can type the keywords to ask for a recommendation. A side box allows the user to know the values for the CEs 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 are shown in the interface's lower part.

6.1.4. ICARE prototype evaluation

We performed two experiments with a total of 46 participants using the ICARE's prototype. The first experiment counted with 21 participants while the second one had 25 participants; all of them were attendees of a course on Intelligent Agents belonging to the Computer Science Post-Graduate Program of our University. They were all gathered in a lab where they could test ICARE at the same time and received an anonymous questionnaire to answer. In the beginning, we presented ICARE's goal and features and how to

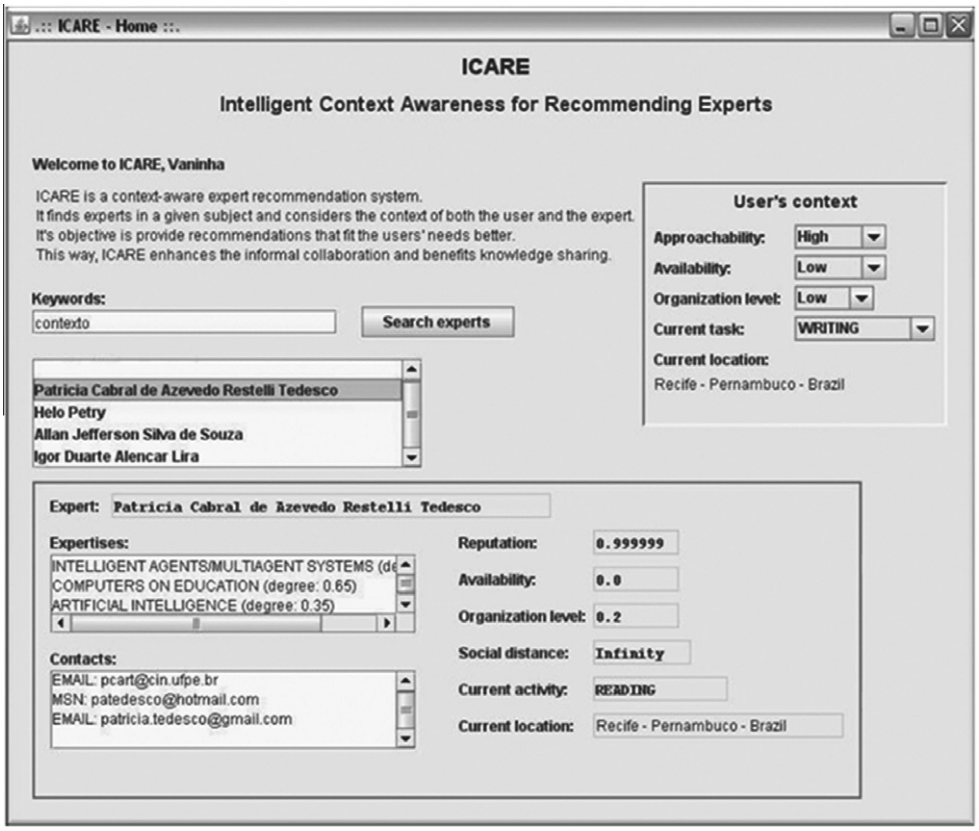

Fig. 11. ICARE interface with an example of a recommendation using context.

use it. Both experiments were similar and aimed at investigating users' acceptance with regard to four aspects: (i) *ICARE strategies* (does the user agree with the identified CEs, and the weights' variation according to the user's context?); (ii) *steps for the recommendation process* (i.e. identifying expertise, retrieving and ranking experts); (iii) *recommended experts*, meaning that the system returned suitable experts; and (iv) consistence of *experts' generated profiles* with their curricula.

Participants were asked to inform their current context: *approachability*, *availability*, *current task*, and *organizational level*, and to request five recommendations. Afterwards, they answered a questionnaire to evaluate the aforementioned objectives. Fig. 12 shows the results regarding to the first objective, i.e. to evaluate if the participants found the ranking elements relevant and if they agreed that the weights of the contextual elements should vary. The participants that answered "no" to the first question specified the irrelevant elements, in their opinion, as being: *social relationship*, *organizational level* and *expertise degree*. We also asked for suggestions of other contextual elements to be included in ICARE. Two participants suggested *geographical position*, *gender*, *city* and *academic degree*.

6.1.5. ICARE prototype evaluation

Addressing the second objective, we obtained the following results: 60.4% of the participants said that the system identified adequate concepts related to the informed keywords; 23.6% said that the concepts were correctly identified, but incorrectly ordered; and 16% indicated that the concepts were incorrectly identified. 57.1% of the participants approved the contextual elements' weights that the system generated and 42,9% disapproved.

For the last two objectives, the results showed that 50.5% of the participants classified the recommended experts as adequate. In other words, in half of the cases, they considered that the experts' curricula were consistent and that the system recommended the

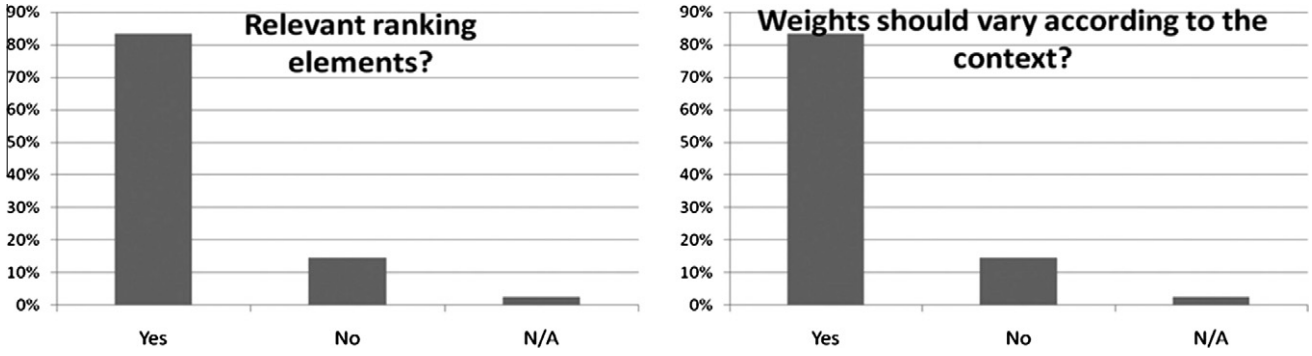

Fig. 12. Evaluation of ICARE: (a) ranking elements; (b) variation on the CEs and relevance weights.

correct experts to the given request. Amongst the participants that disapproved the experts found, 27.1% of them also classified the expertise identified as inadequate (in the same recommendation). Therefore, the system could not find the correct experts if the expertise was not identified correctly first. These results suggest that the system that was used to generate the experts' profiles did not work too well.

6.2. CEManTIKA usage by designers in their CSS projects

To investigate the CSS Design Process usage in different projects, we conducted an experimental study involving nine CSS designers. This experiment had the objective to analyze the comprehension of the CEManTIKA elements by different CSS designers and to identify advantages and difficulties on using the proposed approach.

Eight of the participants were students in a graduate course at our University. They used the process to design CSS projects for the course evaluation. Three projects were designed: (1) *Mobile Meeting*, a CSS that used contextual information about people location and activity to schedule meetings using mobile phones; (2) *Contextual TV*, an application to support program recommendation in digital television, according to patterns found on previous programs viewed by the user; and (3) *Context-Sensitive Middleware for Room Monitoring*, a middleware with functionalities to support applications related to room monitoring that considered contextual information associated to sensors and adaptors. The ninth participant was a M.Sc. student from another University. She used the process to support context modeling for a *notification service* on a distributed software development environment.

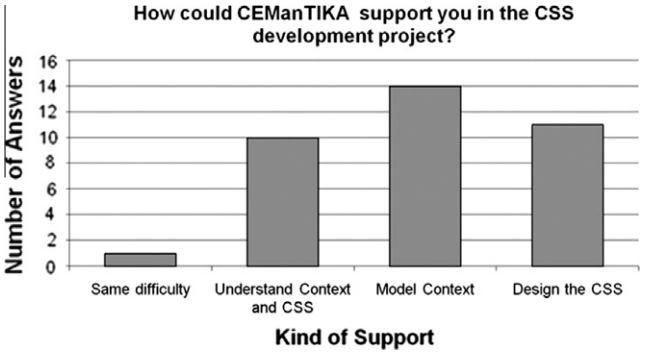

Fig. 13. Evaluation of the support provided by CEManTIKA.

The study was performed according to the following steps: (1) the overall proposal was explained to the participants; (2) they had two weeks to study the CEManTIKA framework and to develop a first version of their projects, following the CSS Design Process; (3) we had frequent interactions to clarify doubts about the process activities; (4) the three projects developed during the course were discussed in a debriefing session; and (5) the project developed by the master student was discussed in several virtual meetings, using a chat tool.

The results found are qualitative and describe users' impressions about the overall proposal and their suggestions for improvement and our observations on how they used the framework on real projects. We observed that the process usage and the integration of context structure with behavior models helped designers to think about how context influences the system's behavior variations, making context more explicit. A participant, for instance, has declared that "the usage of CEManTIKA helped understanding how to map the context dynamics into a conceptual model; the CSS Design Process supported understanding that before thinking about context one needs to analyze the system's behavior variations; by doing so, the applicability of context to the system became clearer".

We could also observe that the usage of the UML Activity diagram to model the contextual graphs following the CxG Profile, caused misunderstandings, since the semantics of the Activity diagram elements conflicted with the contextual graph elements semantics. To solve this, we intend to perform the following improvements: to increase the CxG Profile with icons and new graphical elements, to diminish the ambiguity with the Activity diagram elements; and to create OCL constraints to guarantee consistency checking.

Participants also provided suggestions for improvements that will be considered in further work. For example, they indicated that it will be interesting to create a context-sensitive version of the CSS Design Process, where the process could provide more examples and templates for novice designers while agile methods could be used for experts.

6.3. CEManTIKA evaluation survey with CSS designers

Another study performed with the CEManTIKA proposal was conducted with different CSS developers with previous experience on the design of context-sensitive systems. The objectives of this study were: (1) to assess the comprehension of the CEManTIKA elements; and (2) to elicit the perception of the interviewed participants about the CEManTIKA applicability into their projects. The

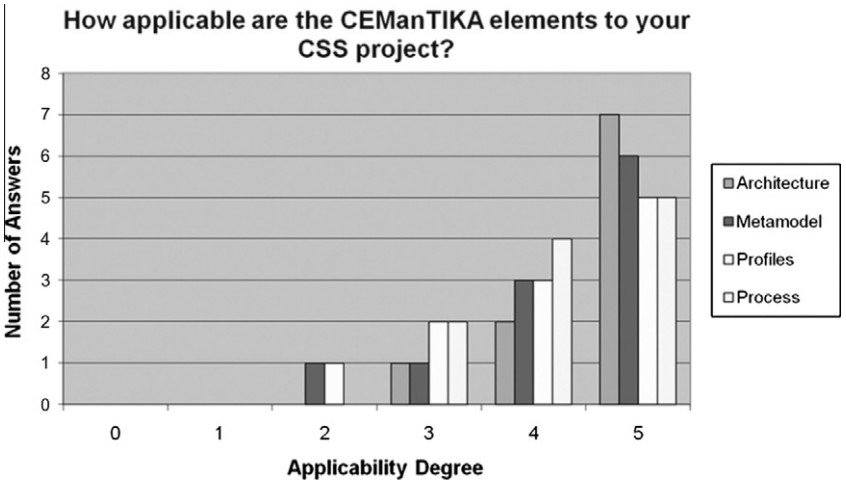

Fig. 14. Evaluation of CEManTIKA elements applicability.

participants were all designers with previous experience on the development of at least one context-sensitive application. Five participants that declared having no previous experience in the development of CSS projects were discarded. A total of 17 people were considered on this study, five (5) of them were also participants in the CEManTIKA usage study (Section 6.2). The participants received a simplified textual version of the CEManTIKA proposal with the explanation of the overall approach and a detailed view of each element. We asked them to read the specification, having their previous projects in mind, and to answer a questionnaire.

When asked about how the usage of the elements proposed in CEManTIKA could support the development of their CSS projects (Fig. 13), 12 of them answered that it supports context modeling, 10 found that it assists understanding context and CSS, 9 indicated that it assists CSS design; and 1 answered that the difficulty is the same. The participant that indicated the same difficulty complained about the absence of implementation support.

We also asked the participants to indicate the degree of applicability level of each element proposed in CEManTIKA in the design of their projects in a scale from 0 (not applicable) to 5 (totally applicable). Fig. 14 illustrates the results found: Ten (10) participants provided answers for this question; the *architecture* was considered totally applicable by seven (7) of them, while the *metamodel* was considered totally applicable by six (6) participants and five (5) participants considered the *profiles* and *process* totally applicable for their projects.

7. Conclusions and further work

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 (Hirschfeld et al., 2008; Yang, Huang, Chen, Tseng, & Shen, 2006), most solutions are proprietary or restricted to specific application domains (particularly, to Ubiquitous Computing). In a preliminary experiment (Vieira, 2008), we observed that designers have difficulties to understand and define what 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.

The research presented in this paper 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: *theoretical*, representing an advance in the literature about context and CSS; and *practical*, with the conception and specification of the CEManTIKA framework and its instantiation in scenarios of usage.

The CEManTIKA Framework included the requirement 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). 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 the particularities of a knowledge domain and there is another part that can be considered in a domain-independent way (Vieira et al., 2007). 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 manipulation of contextual elements and handling of context dynamics.

This paper presented a metamodel and a design process to support context modeling and the design of CSS. The metamodel represents a step towards the formalization of concepts particularly related to context manipulation. The challenges were to identify what concepts should be considered in the metamodel, what they mean, how they relate to each other, and how to formalize them.

The main contributions of the CSS Design Process are: (i) it provides a road map to support the CSS designer when starting a new CSS project; (ii) it proposes a clear separation of the context-related activities, creating a new role in the software development team, the *context designer*; (iii) 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; and (iv) it covers the main activities related to CSS design, providing guidelines, indicating input/output artifacts and a systematic way to execute each activity. The CSS Design 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. Its detailed view of context specification and CSS design activities represents a novelty in the context literature. Our proposal is original in the sense that it goes into detail about activities related to Context Specification and CSS Design. In particular, it argues that business models should be reused and extended to generate the context model.

Some integrated proposals, however, include a support for implementation aspects associated to the CSS development; implementation issues are out of the scope of this work. Currently, we are working on tools to support the instantiation of context models from the Context Metamodel. We are also working on the specification of templates for the documents indicated in the CSS Design Process and on specifying the activities related to CSS implementation, testing and evaluation. Since context is a novel and not yet mature concept, and its application to computer systems is not a trivial task, we believe that the proposed design process will be incrementally improved. To this end, it is necessary to conduct varied and more complex projects and experiments, as well as, to investigate technologies that could support the CSS development. In the future, we also plan to extend the evaluation executed on ICARE by performing a controlled study to test user responses and satisfaction on using a context-sensitive version of ICARE versus a non-context-sensitive one, for the same domain and problem.

Acknowledgements

Authors thank CNPq and CAPES for their financial support. First author also thanks UFBA for its support. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[9]), funded by CNPq Grant 573964/2008-4.

References

Ayed, D., Delanote, D., & Berbers, Y. (2007). MDD approach for the development of context-aware applications. In *Sixth international and interdisciplinary conference on modeling and using context* (pp. 15–28). Roskilde, Denmark.

Bardram, J. E. (2005). The Java Context Awareness Framework (JCAF) – A service infrastructure and programming framework for context-aware applications. In *Third international conference on pervasive computing* (pp. 98–115). Munich, Germany.

Bazire, M., & Brézillon, P. (2005). Understanding context before using it. In *Fifth international and interdisciplinary conference on modeling and using context* (pp. 29–40). Paris, France.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., et al. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing, 6*, 161–180.

[9] http://www.ines.org.br.

Brézillon, P. (2007). Context modeling: Task model and model of practices. In *Sixth international and interdisciplinary conference on modeling and using context* (pp. 122–135). Roskilde, Denmark.

Brézillon, P., & Pomerol, J.-C. (1999). *Contextual knowledge sharing and cooperation in intelligent assistant systems* (Vol. 62, pp. 223–246). PUF, Paris: Le Travail Humain.

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

Chaari, T., Ejigu, D., Laforest, F., & Scuturici, V. (2007). A comprehensive approach to model and use context for adapting applications in pervasive environments. *The Journal of Systems and Software, 80*, 1973–1992.

Chalmers, M. (2004). A historical view of context. *Computer Supported Cooperative Work (CSCW), 13*, 223–247.

Chedrawy, Z., & Abidi, S. S. R. (2006). Case-based reasoning for information personalization: Using a context-sensitive compositional case adaptation approach. In *IEEE international conference on engineering of intelligent systems* (pp. 1–6). Islamabad, Pakistan.

Cheverst, K., Mitchell, K., & Davies, N. (1999). Design of an object model for a context-sensitive tourist GUIDE. *Computers and Graphics, 23*, 883–891.

Costa, P. D. (2007). *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.

Decouchant, D., Imaz, G., Enriquez, A. M., Mendoza, S., & Muhammad, A. (2009). Contextual awareness based communication and coauthoring proximity in the internet. *Expert Systems with Applications, 36*, 8391–8406.

Desmet, B., Vallejos, J., Constanza, P., Meuter, W., & D'Hondt, T. (2007). Context-oriented domain analysis. In *Sixth international and interdisciplinary conference on modeling and using context* (pp. 178–191). Roskilde, Denmark.

Dey, A. K. (2000). *Providing architectural support for building context-aware applications*. Ph.D. Thesis, Georgia Institute of Technology.

Dey, A. K., Salber, D., & Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction Journal, 16*, 97–166.

Dourish, P. (2004). What we talk about when we talk about context. *Personal and Ubiquitous Computing, 8*, 19–30.

Ferrara, A., Ludovico, L. A., Montanelli, S., Castano, S., & Haus, G. (2006). A semantic web ontology for context-based classification and retrieval of music resources. *ACM Transactions on Multimedia Computing, Communications and Applications, 2*, 177–198.

Figueira Filho, C., & Ramalho, G. (2000). JEOPS – Java embedded object production system. *Lecture Notes in Computer Science, 1952*, 53–62.

Fuchs, F., Hochstatter, I., Krause, M., & Berger, M. (2005). A metamodel approach to context information. In *PerCom Workshops* (pp. 8–14). Kauai Island, HI.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Reading, MA/New York: Addison-Wesley/Longman.

Gehlen, G., Aijaz, F., Sajjad, M., & Walke, B. (2007). A mobile context dissemination middleware. In *Fourth international conference on information technology* (pp. 155–160). Las Vegas, Nevada, USA.

Gonzalez, A. J., & Brézillon, P. (2008). Integrating two context-based formalisms for improved representation of human tactical behavior. *The Knowledge Engineering Review, 23*, 1–21.

Gu, T., Pung, H. K., & Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Elsevier Journal of Network and Computer Applications (JNCA), 28*, 1–18.

Henricksen, K., & Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing Journal, 2*, 37–64.

Hirschfeld, R., Costanza, P., & Nierstrasz, O. (2008). Context-oriented programming. *Journal of Object Technology, 7*, 125–151.

Hong, J., Suh, E.-H., Kiim, J., & Kim, S. (2009). Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications, 36*, 7448–7457.

Hong, J., Suh, E.-H., & Kim, S. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications, 36*, 8509–8522.

Kashyap, V., & Sheth, A. (1996). Semantic and schematic similarities between database objects: A context-based approach. *The VLDB Journal, 5*, 276–304.

Kokinov, B. (1999). Dynamics and automaticity of context: A cognitive modeling approach. In *International and interdisciplinary conference on modeling and using context* (pp. 200–213). Trento, Italy.

Kramer, R., Modsching, M., Schulze, J., & Hagen, K. (2005). Context-aware adaptation in a mobile tour guide. In *Fifth international and interdisciplinary conference on modeling and using context*. Paris, France.

Kwon, O., & Kim, M. (2004). MyMessage: Case-based reasoning and multicriteria decision making techniques for intelligent context-aware message filtering. *Expert Systems with Applications, 27*, 467–480.

McCarthy, J. (1993). Notes on formalizing contexts. In *13th international joint conference on artificial intelligence* (pp. 555–560). San Mateo, California.

Merriam-Webster (2008). Merriam-Webster online search: Dictionary and thesaurus [On-line]. <http://www.merriam-webster.com/dictionary> Access 01/2010.

OMG (2003). MDA guide version 1.0.1 [On-line]. <http://www.omg.org/mda/> Access 01/2010.

OMG (2007). Unified modeling language: Superstructure. Version 2.1.2 [On-line]. <http://www.omg.org/spec/UML/2.1.2/> Access 01/2010.

OMG (2008). Software & systems process engineering meta-model (SPEM) specification v.2.0 [On-line]. <http://www.omg.org/technology/documents/formal/spem.htm> Access 01/2010.

Pan, W., Wang, Z., & Gu, X. (2007). Context-based adaptive personalized web search for improving information retrieval effectiveness. In *IEEE international conference on wireless communications, networking and mobile computing* (pp. 5427–5430). Shanghai, China.

Park, D., Hwang, S., Kim, A., & Chang, B. (2007). A context-aware smart tourist guide application for an old palace. In *International conference on convergence information technology* (pp. 89–94). Gyeongju, Republic of Korea.

Petry, H., Tedesco, P., Vieira, V., & Salgado, A. C. (2008). ICARE: A context-sensitive expert recommendation system. In *Workshop on recommender systems (ECAI'08)* (pp. 53–58). Patras, Greece.

Pressman, R. S. (2005). *Software engineering: A practitioner's approach* (6th ed.). New York: McGraw-Hill.

Reichling, T., Schubert, K., & Wulf, V. (2005). Matching human actors based on their texts: Design and evaluation of an instance of the ExpertFinding framework. In *ACM SIGGROUP conference on supporting group work* (pp. 61–70).

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

Riva, O. (2005). A context infrastructure for the support of mobile context-aware services [On-line]. <http://www.cs.helsinki.fi/u/kraatika/Courses/f4fMC/WS1/Riva.pdf> Access 01/2010.

Sacramento, V., Endler, M., Rubinsztejn, H. K., et al. (2004). MoCA: A middleware for developing collaborative applications for mobile users [On-line]. <http://www-di.inf.puc-rio.br/~endler/paperlinks/DSOnline.htm> Access 01/2010.

Sato, K. (2004). Context-sensitive approach for interactive systems design: Modular scenario-based methods for context representation. *Journal of Physiological Anthropological Applications in Human Science, 23*, 277–281.

Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. In *Workshop on mobile computing systems and applications* (p. 90). Santa Cruz, CA.

Sheng, Q. Z., & Benatallah, B. (2005). ContextUML: A UML-based modeling language for model-driven development of context-aware web services. In *International conference on mobile business* (pp. 206–212).

Siebra, S., Salgado, A. C., & Tedesco, P. (2007). A contextualised learning interaction memory. *Journal of the Brazilian Computer Society, 13*, 51–66.

Simons, C., & Wirtz, G. (2007). Modeling context in mobile distributed systems with the UML. *Journal of Visual Languages and Computing, 18*, 420–439.

Souza, D., Belian, R. B., Salgado, A. C., & Tedesco, P. (2008). CODI – A contextual ontology for data integration. In *Fourth workshop on ontologies-based techniques for databases (in VLDB'08)*. Auckland, New Zealand.

Strang, T., & Linnhoff-Popien, C. (2004). A context modeling survey. In *Workshop on advanced context modelling, reasoning and management*. Nottingham/England.

Van den Bergh, J., & Coninx, K. (2005). Using UML 2.0 and profiles for modeling context-sensitive user interfaces. In *Workshop on model driven development of advanced user interfaces Montego Bay*, Jamaica.

Vieira, V. (2008). *CEManTIKA: A domain-independent framework for designing context-sensitive systems*. Ph.D. Thesis Centro de Informática – UFPE, Brasil. <http://www.cin.ufpe.br/~vvs/cemantika/docs/2008_Thesis_VieiraV_CEManTIKA.pdf> Access 01/2010.

Vieira, V., Brézillon, P., Salgado, A. C., & Tedesco, P. (2008). A context-oriented model for domain-independent context management. *Revue d'Intelligence Artificiel, 22*, 609–627.

Vieira, V., Tedesco, P., Salgado, A. C., & Brézillon, P. (2007). Investigating the specificities of contextual elements management: The CEManTIKA approach. In *Sixth international and interdisciplinary conference on modeling and using context* (pp. 493–506). Roskilde, Denmark.

Vieira, V., Tedesco, P., & Salgado, A. C. (2009). A process for the design of context-sensitive systems. In *13th international conference on computer supported cooperative work in design* (pp. 143–148). Santiago, Chile.

Waikato (2008). WEKA – Waikato environment for knowledge analysis [On-line]. <http://www.cs.waikato.ac.nz/ml/weka/> Access 10/2010.

Wang, X. H., Gu, T., Zhang, D. Q., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. In *Workshop on context modeling and reasoning*. Orlando, Florida.

Yang, S. J. H., Huang, A. F. M., Chen, R., Tseng, S.-S., & Shen, Y.-S. (2006). Context model and context acquisition for ubiquitous content access in U-learning environments. In *IEEE international conference on sensor networks, ubiquitous and trustworthy computing* (pp. 78–83).

Yang, S. L. H., Zhang, J., & Chen, I. Y. L. (2008). A JESS-enabled context elicitation system for providing context-aware Web service. *Expert Systems with Applications, 34*, 2254–2266.

Ye, J., Coyle, L., Dobson, S., & Nixon, P. (2008). Representing and manipulating situation hierarchies using situation lattices. *Revue d'Intelligence Artificielle, 22*, 647–667.

Zimmermann, A., Lorenz, A., & Specht, M. (2005). Applications of a context-management system. In *Fifth international and interdisciplinary conference on modeling and using context* (pp. 556–569). Paris, France.

Zimmermann, A., Lorenz, A., & Oppermann, R. (2007). An operational definition of context. In *Sixth international and interdisciplinary conference on modeling and using context* (pp. 558–571). Roskilde, Denmark.