A Context Conceptual Model for a Distributed Software Development Environment

Ana Paula Chaves
System Analysis and Development
Faculdade Integrado Campo Mourao - Brazil
chavesana@gmail.com

Elisa H. M. Huzita
Dept. of Informatics
UEM - Brazil
elisa@din.uem.br

Vaninha Vieira
Dept. of Comp Sciences
UFBA - Brazil
vaninha@ufba.br

Igor Steinmacher
Coord. of Informatics
UTFPR - Brazil
igorfs@utfpr.edu.br

***Abstract*—Organizations seeking lower costs, quality software and specialized resources started to use an approach called Global Software Development (GSD). However, this approach also brought some drawbas imposed by the physical distribution, including issues related to communication, cooperation and coordination. There is a need to improve user awareness related to the process and the context in which collaborative artifacts are generated. In this context, conceptual modeling is relevant since conceptual specifications can be used to support understanding, problem-solving, and communication among stakeholders about a given subject domain. This paper presents a conceptual model focused on contextual information for DiSEN, a distributed software engineering environment. Main contributions found on the paper are: (i) the identification of a set of information that compose environment entities context ; and (ii) the definition of a model that facilitates domain semantics comprehension, reducing communication gaps.**

***Keywords*-Context modeling; Global software development; conceptual modeling**

I. Introduction

Global Software Development (GSD) was introduced to attend software industry quality requirements in a competitive way. Allowing collaborative interaction among geographically distributed development teams became an alternative to organizations looking for competitive advantages, such as time to market, better resource usage, increased productivity and reduced costs. However, GSD also brings many new challenges such as contextual, cultural, organizational, geographical, temporal, and political differences [1]. In order to minimize these differences, one fundamental aspect to focus is the communication. To ensure that geographically distributed individuals are collaborating, it is essential to have an infrastructure that guarantees information and knowledge exchange among all involved parties.

According to Pozza [2], interaction frequency among geographically distributed individuals occurs according to the task they are working on, when there are methods to offer awareness over activities performed by other team members. However, to have a full knowledge on the contributions generated by other team members it is necessary to know not only the collaborative artifact itself, but the way and the context it was produced as well.

DiSEN (Distributed Software Engineering Environment) [3] is an environment aiming to provide an infrastructure to support distributed software development. One of the goals of DiSEN is to provide meaningful and appropriate information regarding entities context during artifacts production. In this sense, DiSEN-CSE (DiSEN-Context Sensitive Environment) [4] was proposed. DiSEN-CSE is integrated to DiSEN and is focused on providing elements to capture, represent, process and disseminate context information to interested individuals. But, to allow context sharing, it is necessary to know what information is part of the context of a given action, how these information are related and how they affect environment behavior.

In order to facilitate contextual information handling, some representation models were created, focusing on offer information to different entities (people, software or devices) in a way that any of them would have the same semantic comprehension of what was informed [5]–[7]. Conceptual modeling is relevant on application design process, since conceptual specifications are used to support understanding, problem-solving and communication among stakeholders about a given subject domain. Based on this, an alternative is Context Metamodel [8], a domain independent context metamodel to support contextual model creation based on UML extensions.

This paper presents a context conceptual model focused on context information for a distributed software engineering environment. The context conceptual model presented here uses as conceptual base the context metamodel proposed in [8]. This model allows identifying the set of contextual information relevant to each environment entity, how these information are related among themselves and how context and environment behavior are related.

The rest of this paper is organized as follows: section II presents the Context Metamodel used to create context conceptual models, section III introduces the context conceptual model created for DiSEN, section IV presents a preliminary proof of concept, section V presents some related works and, finally, some conclusions and final considerations are presented on section VI.

II. Context Metamodel

After analyzing different existing context models, Vieira [8] checked that, in general, there is no formalism for

creating context models that associate contextual information and its usage. In this sense, existing models do not relate contextual information and its use. Based on that observations, Context Metamodel was proposed. Context Metamodel is a domain independent metamodel to support contextual model creation. This metamodel is an UML 2.0[1] extension.

Within Context Metamodel *Contextual Element* is defined as any piece of data or information that enables to characterize an entity in a domain; and *Context* of an interaction between an agent and an application, in order to execute some task, is the set of instantiated contextual elements necessary to support the task at hand. Other than that, contextual element is stable and can be defined at design time, while a context is dynamic, and must be constructed at runtime, when an interaction occurs.

Context metamodel is divided into two main packages that organize the concepts in two categories: structure, which describes the concepts related to the conceptual and structural elements of a context sensitive system (context structural model); and behavior, which contains the concepts related to the behavioral aspects of a context sensitive system (context behavior model).

Context structural model identifies contextual elements; the entities that contain them; relationships among them; and the way they are acquired. Context behavior model focuses on modeling how the system must react under a specific context, identifying the set of condition responsible for a system behavior variation. To support the design of the context sensitive system behavioral part, the context metamodel uses the concepts defined in the formalism of Contextual Graphs, defined in [9]. Contextual graphs are based on semantic networks, and support task models representation. They consider the relation between procedures established by an organization for executing a given task, and how contextual information influences this task execution. Each path on the contextual graph contains the rationale used to execute a task. It contains the sequence of triggered actions, conditions activated for each action, and context elements related to each condition. So, this kind of representation allows one to compose rules graphically, based on the paths described on the graph.

III. Modeling Context in DiSEN

A. DiSEN Environment

DISEN (Distributed Software Engineering Environment) [3] is an environment aiming to provide an infrastructure to support distributed software development. DiSEN design is composed of three main Managers : (i) Object Manager – comprises all environment entities, involving Artifacts, Processes, Resources, Projects, Tools, Users and Local Managers; (ii) Agent Manager – responsible for creating, registering, finding, migrating and destroying software agents; and (iii) Workspace Manager – provides shared workspaces, corresponding to environment instances where users can cooperate during tasks execution. DiSEN Object Manager has a Resource Manager responsible for managing physical resources (machines, paper, media) and also Places, Users and Tools. These three entities are responsible to provide contextual information to the environment. Users (a person or a group) are the most active entities within the environment capable to change states and generate information for the whole environment.

B. DiSEN-CSE

To allow contextual information sharing among several workspaces involved on a cooperative work, DiSEN-CSE (DiSEN-Context Sensitive Environment) was proposed. To reach this goal, every time a contextual information is generated by an Object Manager entity (Place, User or Tool) at any workspace within DiSEN, this information is captured by workspace manager. Within workspace manager, DiSEN-CSE transforms them into another information that can be understood by all team members geographically distributed, using a representation model based on a domain ontology. Then, DiSEN-CSE shares it with other workspaces, including the one that generated it.

DiSEN-CSE model has four essential elements: Capture Support, Context Representation, Processing Support and Awareness Mechanisms. The model also has a repository responsible for storing context information. Capture Support is responsible for recognizing context changes occurred during task execution. These changes can be captured by human agents or software agents. Context Representation is responsible for mapping information coming from Capture Support to a formal representation model – based on ontologies – and relates them to other information available within the repository. Processing Support corresponds to reasoning mechanisms, able to infer contextual information implicit on that captured beforehand, and fix possible inconsistencies, based on existing relationship among sets of information created by Context Representation. Finally, Awareness Mechanisms are responsible for identifying (automatically) what are the workspaces interested on the information, the methods that can be used to show them (defined by Pozza [2]) and for spreading them to workspace instances, using communication infrastructure.

So, DiSEN-CSE goal is to capture, manipulate and share contextual information available within DiSEN environment. However, to reach this goal, it is necessary to know what are contextual information available on the environment and what are the relationships among them. To achieve this, we propose a Context Conceptual Model, which is described in the next section.

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

C. The DiSEN Context Model

According to [8] context is composed by elements related to an entity that is useful when supporting a problem resolution. To identify the elements with this feature within DiSEN domain, a conceptual model was created using Context Metamodel to specify main environment functionalities, defining contextual elements and the relationships among them, and the possible environment actions.

Vieira [8] says that the first step when building a context model using the context metamodel is to identify the main *foci* to be considered in the system. *Focus* is what enables to determine which Contextual Elements should be instantiated and used to compose the context. Brezillon [10] considers it to be a step in a task execution, in a problem solving, or in a decision making. In the context metamodel, *focus* is determined by the association between a task ($\langle\langle Task \rangle\rangle$) and the agent who is executing it ($\langle\langle Agent \rangle\rangle$).

All *foci* considered are depicted on Fig. 1 and are determined by DiSEN functionalities. In this paper we will use the *focus* (User, `Acquiring Knowledge`) to illustrate context modeling, chosen because it is an easy to understand *focus* and has all elements needed to explain the model.

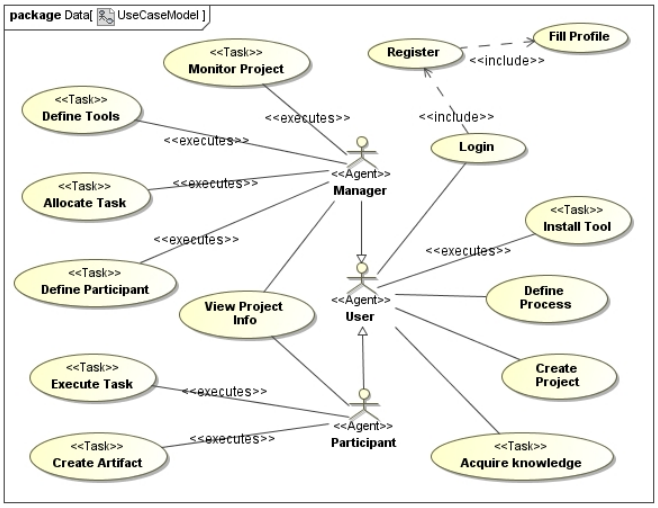

Figure 1. Use Case Model

1) Building the Context Structural Model: Inside DiSEN environment, an user can acquire knowledge in two different situations: when working on a project or when attending to a training. So, the following behavior variations were verified for the focus Acquiring Knowledge:

1. Users have knowledge. They can acquire knowledge when attending to trainings and when working on a project.

2. When an user attends to a training, the skillset acquired on that training and the level are defined. If an user attends to a basic level training, acquired knowledge is basic. If the training is an intermediate level, the skill offered is intermediate. And if the training is advanced, the skill level offered is advanced.

3. When an user participates on a project, he/she may not have all required skills for that project. But, at the end of the project he/she may have acquired some of them. Users must inform to the system all new skills acquired.

Based on these variations, it is possible to verify that the elements that interfere on *focus* context are: User.doATraining, UserTraining.level, Participant.isOnProject, User.hasKnowledge, project.Status, Project.requireKnowledge, UserKnowledge.level. These elements represent contextual elements for this *focus* and are depicted on Contextual Structural Model of Fig. 2. Each context element is tagged with $\langle\langle ContextualElement \rangle\rangle$ stereotype. An entity with at least one contextual element is a contextual entity, tagged with $\langle\langle ContextualEntity \rangle\rangle$ stereotype. In this sense, the behavior that indicates user knowledge is represented by the contextual element User.hasKnowledge. The behavior that shows that users can attend to a training is specified by the contextual element User.doATraining and so on.

Context Structural Model also represents the way the information is acquired. Each contextual element has its own acquisition properties, indicating where the information can be found, the type of acquisition and the frequency the information is updated. Optionally, it can have a formation expression, showing a new rule to transform an obtained information to the expected format. These properties are represented by an acquisition relationship among a contextual entity (contextual element container) and the context source (the one which indicates the information source).

Fig. 2 depicts acquisition relationships. PersistenceService and InterfaceGUI entities are tagged with $\langle\langle ContextualSource \rangle\rangle$ stereotype, representing context sources. For each contextual element there is a relationship with its own source, indicating acquisition configuration. These relationships are tagged with $\langle\langle acquisitionAssociation \rangle\rangle$ stereotype.

Context structural model presents the system structure for a given *focus*, showing up contextual entities and elements, relationships among them and their acquisition way. However, when it is desired to design a Context Sensitive System, it is necessary to know system behavior, determined by the context. In this sense, behavioral model was created. It consists on contextual graphs able to determine possible behavior variation for each context. Context behavior model and some rules will be detailed on the following sub-section.

2) Building the Context Behavior Model: The contextual graph that defines the behavior of Acquiring Knowledge *focus* is depicted on Fig. 3. Behavioral variation is represented as a decision node, tagged with $\langle\langle ContextualNode \rangle\rangle$. When an action occurs (tagged with $\langle\langle Action \rangle\rangle$ stereotype), it reaches a $\langle\langle ContextualNode \rangle\rangle$, and the environment must decide, according to the context, what is

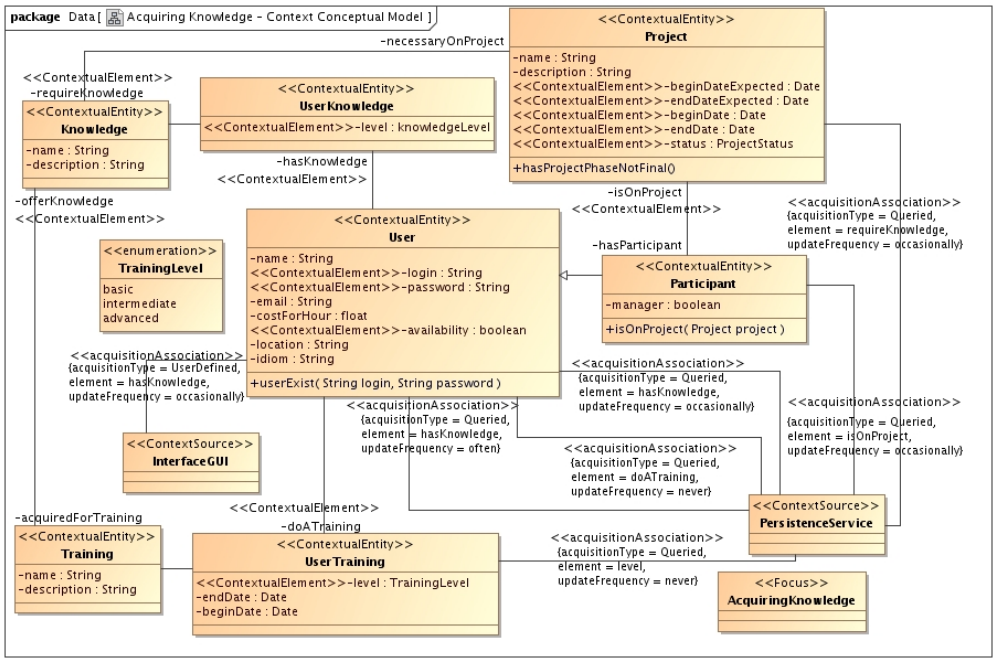

Figure 2. Context Structural Model – Focus: Acquiring Knowledge

the path to be followed. Possible paths are tagged with $\langle\langle ContextualBranch \rangle\rangle$ stereotype. At the end, after performing all actions related to chosen path, there is a $\langle\langle RecombinationNode \rangle\rangle$, responsible to ensure graph consistence. Recombination node indicates that all actions related to that context were concluded, regardless what is the chosen branch.

The paths depicted by the graph can be converted to compose the rules in the context conceptual model. Some examples for Acquiring Knowledge *focus* are depicted on the right side of Fig. 3. The rules presented indicate necessary conditions to allow a given user to acquire a new knowledge.

Structural and behavioral context models made it possible to determine which information influences the context and how the changes in the interactions affect environment behavior for each focus. Section IV describes how the context conceptual model was implemented to support information sharing within DiSEN environment.

IV. Proof of Concept

The context conceptual model presented here was implemented on DiSEN environment, during the development of an event notification manager called DiSEN-Notifier. DiSEN-Notifier is a responsible by receiving the information captured (Capture Support), represented (Context Representation) and processed (Processing Support) and deciding who should be notified about the events occurred. In addition, this manager prepares a message and sends it to any available Awareness Mechanism, which notifies the users in an appropriate way.

The decision on who should receive a notification at a given time is based on context behavioral models. Thus, DISEN-Notifier becomes context-sensitive as it implements the conceptual model presented in this paper and can, based on the model, find out what people are interested on a specific focus and therefore should be notified on events occurred . Moreover, as behavioral patterns define the context under which the notifications should be sent, when those predefined circumstances occur, the system triggers the notifications automatically, without any action performed by the user requesting to send or to check for new incoming information. More information about the DISEN-Notifier and how to use the conceptual modeling of context for reporting information can be found in [11].

In addition, an ongoing research is using the context conceptual model defined here to support processing rules definition for Context Representation element on DiSEN-CSE. These rules will be used to make inferences from information captured, based on an ontological model. The results of this research will be presented in future work.

V. Related Work

Many proposals had been made in order to allow context modeling. However, most of these proposals focus on ubiquitous or pervasive environments [7], or collaborative applications in general, so far there are few studies related to context modeling for Global Software Development. This section presents some researches related to this paper.

Regarding collaborative applications, Rosa et. al [12] proposed a conceptual framework to identify and classify

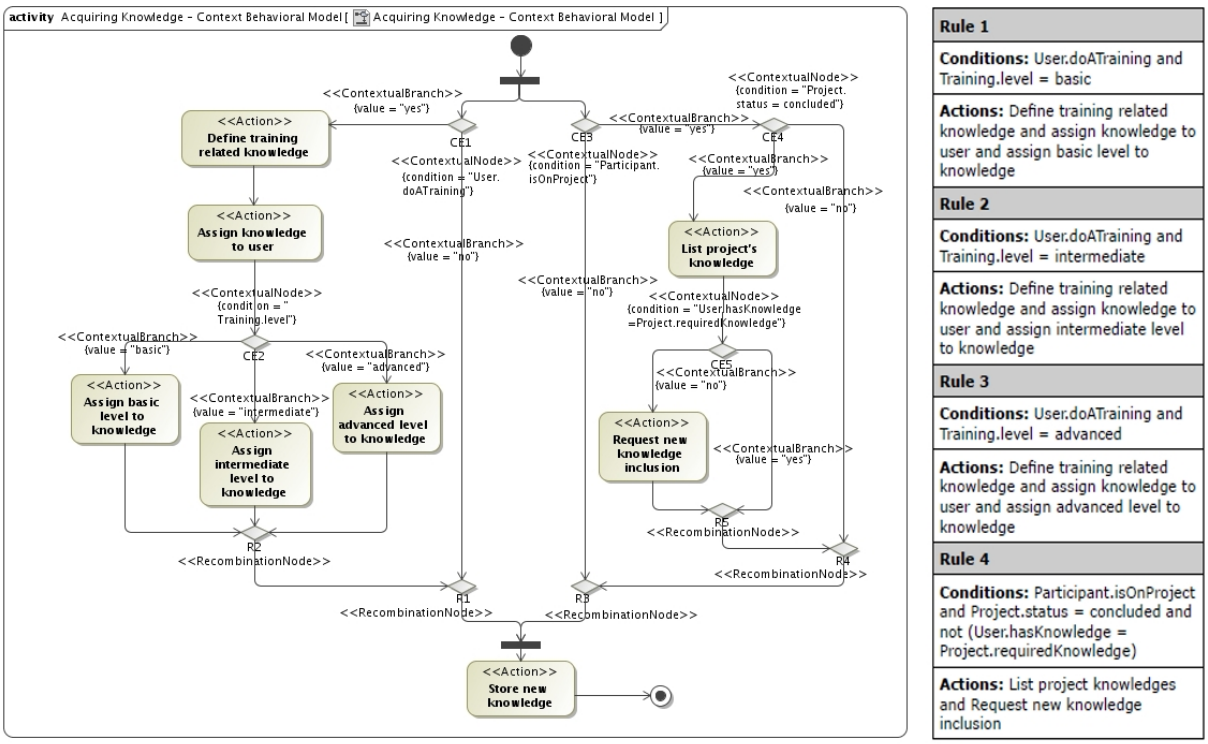

Figure 3. Context Behavior Model and some rules extracted – Focus: Acquiring Knowledge

most common contextual information on groupware tools. This framework classify the information into five types: group members, tasks, relationship among people and tasks, setting and completed tasks. Based on these information, this framework keeps important information for groupware users be aware of the influence of their interactions. Vieira et. al [13] propose an ontology to formally represent context in groupware systems. The idea is to identify which information presented in groupware applications could be classified as context and which kinds of context should be represented. Nunes et al. [14] presented a model for managing context-based knowledge, which addresses the creation, storage and reuse of contextual knowledge, encompassing the representation, capture, storing, comparison, and presentation of knowledge in the setting where the work process activity is performed. This knowledge management model aims to establish organizational memory, including the results of performing the work process activity and the context through which results were produced. However, none of these studies bring integrated solutions to support context structural and behavioral aspects, nor are related to GSD.

Specifically for context modeling in GSD, Eckhard [15] developed a language called NSL (Notification Specification Language) to support NOTICON, a tool that allows context based notification. The NSL language used to define context was created specifically for NOTICON and requires a specialist to model context. For each focus a file containing context specification is generated, making it difficult to reuse already modeled *foci*.

Veiel [16] defined a context model to adapt collaboration in shared workspaces. This context model addresses scenarios in which several actors collaborate to achieve a shared goal and thereby captures basic concepts of co-located and distributed collaboration. This model identifies classes and relationship needed to model configuration of shared workspaces and tools, and to capture the current context at runtime. Based on the relationships created at a given moment, adaptation rules capable to use context are defined to adapt user workspaces involved on the interaction. The rules that define the adaptive behavior are textually described, and do not follow any formalism or standardized language. Furthermore, behavior is described textually, without any model to support understanding and reuse.

Conceptual modeling, as presented on this paper, allows representing elements within a given application domain and their possible relationships. The use of conceptual modeling to identify contextual information on GSD domain brings as advantage the ability to represent structural (Context structural model) and behavioral (Context behavior model) viewpoints. In this sense, this work differs from the ones presented here by presenting a way to facilitate communication among geographically distributed individuals, considering dynamic context, differing context and contextual elements. The model also offers a standard way of context representa-

tion that allows model reuse and easy understanding, since UML is well established on industry and academy.

VI. Conclusion

A well known problem caused by geographical distribution on GSD environment is the communication gap. This is mainly caused by mistrust among team members, lack of disposal to help people involved on the same task, cultural and organizational differences. Because of this, a context-awareness based model was proposed to share contextual information among several workspaces on a distributed software development environment, DiSEN. However, to offer contextual information it is necessary to know entities and what set of information about these entities are important to create the context of a given focus. In this sense, this paper presented a context conceptual model for DiSEN.

To create the context conceptual model context metamodel [8] was used. The use of Context metamodel improved model semantic comprehension and made easier to design context concepts in context sensitive environment DiSEN. The metamodel also supports dynamic context understanding, brings up the difference among context and contextual elements and presents how context can be used to model behavior variation on different domain areas. Acquiring Knowledge focus was used to demonstrate the context conceptual model produced. Thus, the main contribution of this paper is a context conceptual model for GSD, integrating structural and behavioral aspects by: (i) identifying which set of information will compose environment entities context; (ii) identifying how these information can be used to design system behavior based on the context; and (iii) facilitating domain semantics comprehension and reuse of context models. In addition, this model was used as a base to implement an event notification manager called DiSEN-Notifier.

There are some future work in progress related to this research: (i) evaluate context information transformation, and its behavior within DiSEN when considering context information present on conceptual model; (ii) link conceptual information identified by conceptual model and the ontology based representation model, to verify consistency for the considered domain; (iii) explore sharing mechanisms for captured contextual information; and (iv) use the context conceptual model presented here to support processing rules definition for Context Representation on DiSEN-CSE.

Acknowledgments: The author Ana Paula Chaves thanks CAPES for financial support.

The author Vaninha Vieira thanks the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq grant 573964/2008-4, for their financial support.

References

[1] M. Ivcek and T. Galinac, "Aspects of quality assurance in global software development organization," in *31st International Convention MIPRO*, 2008, pp. 150–155.

[2] R. S. Pozza, "Proposta de um modelo para cooperacao baseado no gerenciador de workspace no ambiente disen," Master's thesis, PCC, UEM, 2005.

[3] E. H. M. Huzita, T. F. C. Tait, T. E. Colanzi, and M. A. Quinaia, "Um ambiente de desenvolvimento distribuido de software – disen," in *I WDDS*, Joao Pessoa - PB, 2007, pp. 31–38.

[4] A. P. Chaves, I. S. Wiese, C. A. da Silva, and E. H. M. Huzita, "Um modelo baseado em context-awareness para disseminacao de informaces em um ambiente de desenvolvimento distribuido de software," in *CLEI 2008*, Argentina, 2008.

[5] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Workshop on Advanced Context Modelling, Reasoning and Management, in 6th International Conference on Ubiquitous Computing*, Inglaterra, 2004, pp. 34–41.

[6] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of CNDS*, USA, 2004, pp. 270–275.

[7] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," in *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2. Inderscience, 2007, pp. 263–277.

[8] V. Vieira, "CEManTIKA: A domain-independent framework for designing context-sensitive systems," Ph.D. dissertation, CIn – UFPE, Recife, 2008.

[9] P. Brezillon, "Task-realization models in contextual graphs," in *CONTEXT*, ser. Lecture Notes in Computer Science, vol. 3554. Springer, 2005, pp. 55–68.

[10] P. Brezillon and J. C. Pomerol, "Contextual knowledge sharing and cooperation in intelligent assistant systems," *Le Travail Humain*, vol. 62, no. 3, pp. 223–246, 1999.

[11] A. P. Chaves, E. Huzita, and V. Vieira, "Context-based notification supporting distributed software teams management and coordination," in *SBSC 2009*. IEEE SC, 2009, pp. 80–89.

[12] M. Rosa, M. Borges, and F. Santoro, "A conceptual framework for analyzing the use of context in groupware," *Lecture Notes in Computer Science*, pp. 300–313, 2003.

[13] V. Vieira, P. Tedesco, and A. Salgado, "Towards an ontology for context representation in groupware," vol. 3706. Springer, 2005, pp. 367–375.

[14] V. T. Nunes, F. M. Santoro, and M. R. Borges, "A context-based model for knowledge management embodied in work processes," *Information Sciences*, pp. 2538 – 2554, 2009.

[15] B. Eckhard, "Context-aware notification in global software development," Master's thesis, Institut fr Softwaretechnik und interaktive Systeme – Technischen Universitt Wien, 2007.

[16] D. Veiel, J. Haake, and S. Lukosch, "Extending a shared workspace environment with context-based adaptations," in *CRIWG*, ser. Lecture Notes in Computer Science. Springer, 2009, pp. 174–181.