OntoDiSEnv1: an Ontology to Support Global Software Development

Ana Paula Chaves, Igor Steinmacher
Federal Technological University of Paraná – Coordination of Technology Systems for Internet
Campo Mourão, Brazil, 87301-006
{anachaves, igorfs}@utfpr.edu.br

and

Gislaine Camila Lapasini Leal
State University of Maringá – Departament of Production Engineering
Maringá, Brazil, 87020-900
gcleal2@uem.br

and

Elisa H. M. Huzita, Alberto B. Biasão
State University of Maringá –Department of Informatics
Maringá, Brazil, 87020-900
emhuzita@din.uem.br, biasao@gmail.com

Abstract
Global Software Development (GSD) brought competitive advantages to organizations, but it has also imposed some drawbacks due to the physical distribution. A critical aspect of this approach is related to communication. In order to provide the same semantic understanding about information exchanged on the environment to all team members it is necessary to minimize the ambiguity. This paper presents OntoDiSEnv1, application ontology for a distributed software development environment. The goal of this ontology is support communication among geographically dispersed team members. The ontology is integrated to a contextual information dissemination model, which notifies the team members about the actions that occur on the shared workspace and can influence their work. The main contribution of OntoDiSEnv1 is to support contextual information representation and processing, providing inference capability and semantic consistency of the information disseminated.

Keywords: Ontology, Global Software Development, Communication

1 Introduction
The Global Software Development (GSD) is a development approach in which teams that work collaboratively in a common project are geographically distributed. This kind of approach can bring many benefits in terms of cost and quality using qualified professionals spread from all around the world, better time-to-market and continuous development (follow-the-sun)[1]. Besides its advantages, GSD brings new challenges related to coordination, cooperation and communication, such as contextual, cultural, organizational, geographical, temporal, and political differences. To minimize these differences and ensure that geographically distributed individuals are collaborating, it is essential to have an infrastructure that guarantees information and knowledge exchange among all involved parties [2].

Communication is a crucial element, allowing developers to know the state of tasks and resources, the activities of other team members, and so on. According to [3], the communication has been assuming important role to the development of software, where the bad communication or the lack of it can put in danger success of the projects. However, when software development involves geographically distributed teams, communication is more difficult than with collocated teams [4]. In this case, the participants are much less likely to have unplanned contact with other sites due to absence of face-to-face, hallway, and lunch conversations [5]. The lack of this common sort of poor
socialization and other factors may reduce trust among team members [6], bringing losses over the communication. Thus, members of a virtual team tend to be less productive due to feelings of isolation and indifference [7].

DiSEN is a distributed software development environment aiming to provide tools and services to support coordination, communication, persistence and cooperation in geographically distributed settings. One of the goals of DiSEN is to provide meaningful and appropriate information regarding entities context during artifacts production. Given this fact, DiSEN-CSE (DiSEN-Context Sensitive Environment) [8][2] was proposed. DiSEN-CSE is a context-based model focused on providing automatic information and sharing regarding interactions occurring in DiSEN.

During the development of this model, we noticed that, as messages were sent automatically by the environment, there was a need to reduce possible ambiguities and find ways to ensure that individuals receiving these messages have the same semantic comprehension. Junior and colleagues [3] affirm that solutions for communication focused mainly on the need to define and formalize it. Attempting to accomplish this need, we suggested an ontological model, called OntoDiSEN. Currently, in OntoDiSENv1 version, the ontology depicts concepts of contextual elements that are represented and shared by the dissemination model. The goal of this paper is to present OntoDiSENv1, highlighting the main features of its latest version.

2 Research Methodology

The research presented in this paper is characterized as qualitative, of the exploratory type, divided into three phases: problem definition, literature review and solution development. The problem definition consists of analyzing DiSEN environment and the GSD domain, having the objective of identifying problems arising from communication. During this analysis, it was identified the need of a representation for information, which would make a common semantic comprehension among participants in the collaborative work. Section 3 describes the problem.

Concerning the literature review phase, we found some models for information representation and their main features. Among these models are: models based on key-valued pairs, logic based models, topic maps based models, context graphs based models, ontology based models [9][10]. Analyzing features like expressiveness, formalism, inference capacity and available tools, the technique that stood out was the representation based on ontologies. The concepts about ontologies and the advantages offered by this type of representation model are presented in section 4. Furthermore, research has been conducted to search studies that are, somehow, related to the objectives of this research. The results of this step can be found in Section 5.

Finally, the solution development phase consisted in the design of an ontology for DiSEN environment. The goal of this ontology is to allow that all individuals receiving automatic messages have the same semantic comprehension of what is being informed, regardless of their geographic location. Furthermore, the ontology facilitates contextual information processing, inferring the actions that can be performed, according to the context represented, to disseminate the right information the appropriate way. The ontology developed in this phase, the steps taken for its definition and the main features designed in the current version are presented in Sections 6 and 7, respectively.

3 Background

This section presents the background and relevant concepts in which this paper is based. We bring a short discussion about awareness and context definitions and a brief presentation of DiSEN environment and DiSEN-CSE model.

3.1 Awareness and Context

Trying to reduce the communication gap among geographically distributed teams, researches are conducted to increase their awareness [11]. Awareness represents, therefore, an understanding of the activities of others, which provides a context for your own activity [12]. Its objective is to allow a group of people working collaboratively to realize how and which of their contributions are relevant to the group activities [11].

When dealing with Global Software Development, the physical distance requires that to get to the contributions of other group members it is necessary to know not only the object of cooperation, but also the way in which it was produced [12]. This information reflects the context, comprised by the circumstances that are surrounding activities development.

The application of context to support the awareness has been widely discussed in recent years; however, there is still no consensus on its definition, as on each domain area there is a different understanding of the term according to specific interests, as can be observed in [13]. In this paper, the definition used was proposed by [14], which states that “the context of an interaction between an agent and an application, in order to perform a task consists of a set of instantiated elements that characterize the entities present in application domain and which are needed to support
task performance”. It is important to highlight the relationship between awareness and context. Awareness mechanisms are focused on sharing information about the collaborative object.

In this respect, context-sensitive systems have advantages over traditional systems, as it offers the ability to manage (acquire, represent, store, process and share) the context of actions that occur in the workplace. When the information is automatically shared – as on notification mechanisms – it is important to be aware of the context in which they were generated in order to provide only the information that is relevant to the context of the user. This feature reduces the information overload, increases the relevance and comprehension of the messages, and facilitates coordination among team members.

### 3.2 DiSEN and DiSEN-CSE

DiSEN is an environment that aims to provide tools to support persistence, information management and communication and cooperation among geographically distributed team members [15]. Many works have been developed in this environment, as mentioned in [15], including: project management tools, requirements modeling, version management and modification, and communication support tools.

A context based model for information sharing was proposed by [8]. The goal of this model, named DiSEN-CSE (DiSEN-Context Sensitive Environment), is to increase the awareness of project members about actions that occur in the environment and share these information automatically, according to the context that involves these actions and the individuals influenced by them.

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 in charge of recognizing context changes occurred during task execution. These changes can be captured by human agents or software agents. Context Representation deals with mapping information coming from Capture Support to a formal representation model – based on ontologies – and relates them to other information available in the repository. Processing Support corresponds to reasoning mechanisms, it is able to infer contextual information implicit on that captured beforehand, and fix possible inconsistencies, based on existing relationship among sets of info created by Context Representation. Finally, Awareness Mechanisms are responsible for identifying (automatically) the workspaces interested on the information, the methods that can be used to show them and also for spreading them to workspace instances, using communication infrastructure.

Differences raised by physical distance can affect individuals’ comprehension about shared information. The impact is that, information about the way collaboration objects are produced by distant teams, may present ambiguity or lack of clarity, what would cause fails or uncertainty on software development. The solution proposed by [8] was to include in DiSEN-CSE an element responsible for representing those information, based on an ontological model, promoting appropriate and uniform context information spread.

### 4 Ontology

The term ontology came up in philosophy, to describe things in the real world. In the last years, it achieved space upon computer science areas – artificial intelligence and computational linguistics. In those areas, the most accepted definition for the term was made up by [16], who states that an ontology is a formal specification of an explicit conceptualization. Conceptualization, in this case, corresponds to an abstract and simplified vision of the world wanted to be represented. According to [17], an ontology defines a common vocabulary for individuals or systems that need to share information about a domain. Nevertheless, according to [18], an ontology is not just a vocabulary or taxonomy. More complex ontologies include axioms that raise complexity upon relations, concepts and restrictions, offering the desired interpretation to that domain. In this paper, the definition used was the one proposed by [16], once OntoDiSEN goal consists in representing, formally and explicitly, the existent concepts in DiSEN environment, offering semantic meaning to these concepts and, consequently, reducing ambiguities and incomprehension.

An ontology is a set of entities, also called concepts or classes, that represent the domain concepts [17] and can be hierarchically organized. Entities have properties, which correspond to features and attributes that describe them. Moreover, these properties may have restrictions, to increase the precision of the specification. Each entity has a set of individuals, which represents the concepts instances. Thus, each individual has the same properties and restrictions as the entity it belongs to.

When an ontology assumes a fundamental role in an application, it is classified as an application ontology that, according to the classification proposed by [19], specializes the ontology concepts for that application, depicting concepts related to activities performed on its domain. According to [20], the motivations on using ontologies are: sharing common knowledge about information structure, among humans or software agents; analyze domain knowledge; and allow the use of inference mechanisms to reason over different contexts.
5 Related Works

There are many techniques used to represent and manipulate contextual information on different domain areas. Checking the literature it is possible to observe a convergence on the use of ontologies to constitute this kind of information. However, most of the proposals focus on other domains, mainly on ubiquitous and pervasive environments with physical sensors. Some studies [22][23][24][25][26] present ontologies as means to represent information for software engineering and GSD domain were found, but they are not used to represent and process contextual information.

On ubiquitous and pervasive domain, we can find Context Managing Framework [27], a pervasive computing framework that uses an ontology-based model to convey the context in order to share information among mobile devices/applications. LOTUS [28] is a plugin-based middleware to support pervasive application development with support to acquisition, representation and inference of contextual information, which are represented by an ontology-based model. SOCAM [29] is an architecture that supports the construction of context-sensitive services for pervasive computing settings that uses an ontology-based model to denote the context and infer actions to be triggered. CoBrA [30] proposed an agent-based architecture to support context-sensitive systems in intelligent environments that uses ontologies acting in behalf of the structure of knowledge acquired by physical sensors, software agents and via web.

On CSCW (Computer Support Cooperative Work) domain we can find a related study presented by [31] that proposes an approach, based on ontological models, which is devised to help the developer of an observation system for a groupware application to structure and record user actions. Their proposal is based firstly on an ontological model that describes the collaborative work processes to be carried out. Secondly, a set of descriptors is used for recording in real time the actions performed by the users through user interface components. The observation system subsequently captures the actions accomplished during the work process and records them in log documents. Although it brings a way to capture information on groupware applications, the work presented by Duque and his colleagues are not focused on GSD specific issues.

On software development domain in general, ontologies are explored in several researches. Falbo et al. [32] proposed ODE, an ontology-based and process-centered software development environment. Silva and Barreto [33] presented an approach to support model oriented software development, which allows developers to both describe and verify domain properties at run time. The ontology is used to specify domain properties that are translated to aspect-oriented code to be automatically merged in the implementation. In [34], the authors used ontologies and Web services to extend framework of components to the domain of content adaptation, which facilitates the development of software based on reuse in context of ubiquitous computing. Lima [35] inserted a domain ontology in the software development process to allow traceability of artifacts.

Regarding the use of ontologies within GSD domain, [22] presents a framework based on previous generic models for requirements elicitation processes, which focuses on predicting problems and proposing different strategies to avoid or decrease their impact on GSD project performance. They use ontologies in order to facilitate the communication during requirement elicitation process. Wongthongtham et al. [23] propose a framework that uses ontologies as part of a communication framework for multi-site distributed software development environments. They organize software engineering concepts, knowledge, software development methodologies, tools and techniques into an ontology and use it as the basis for classifying the concepts in communication, thereby enabling questions, problem solving and sharing solution development and knowledge to be shared between multi-site teams. Wongthongtham et al. [24] have analyzed software engineering ontology as knowledge and data warehousing for multi-site software development settings, using the communication framework presented in [23]. In [25] an ontology model of software engineering to represent its knowledge is presented. They have analyzed the characteristics of software engineering ontology and defined graphical notations of modeling software engineering ontology as an alternative formalism. Silva e Fantinato [36] consider the use of ontology as a resource to support the definition of electronic contracts for inter-organizational processes in GSD.

Zhao et al. [26] provide a survey and a classification on ontologies developed for software engineering, also reviews the current efforts on applying the Semantic Web techniques on different software engineering aspects and phases. They provide as result a comprehensive view on the current approaches proposed for software engineering, by presenting which area has been fully covered and which has not. Although presenting an extensive review of the state of the art and a classification on the use of ontologies and semantic web on different aspects of software engineering, they do not present or analyze any ontology to represent GSD information, neither to represent contextual software engineering information.

6 OntoDiSEN

OntoDiSEN is an application ontology, developed specifically for DiSEN environment, on GSD domain. There are several methodologies that aim to conduct and support the construction of ontologies [37]. However, none is
considered a development standard. The methodology used for the construction of OntoDiSEN corresponds to an intersection, based on [38], of the steps followed in studies in the literature. This intersection uses the approaches proposed in [39] and [17] in a complementary way. The first one defines the methodology phases and the second approach supports the conceptualization process. More details about the methodology can be found in [38]. The development of this ontology was based on the following items:

- **Domain definition**: Global Software Development Environment;
- **Application definition**: DiSEN environment;
- **Main goal definition**: unambiguously represent the information related to the action context of individuals, locals and tools of a GSD environment, more specifically, DiSEN;
- **User definition**: ontology users are the users of DISEN environment tools, services and software agents available in the environment;
- **Resources definition**: modeling tool (Protégé 4.0 [40]) and modeling language (OWL-DL [41]).

Once it had been planned, the next step to build the ontology was the knowledge acquisition. So, it was necessary to acquire (from different knowledge sources) relevant information regarding action context composition; GSD domain; and features of DISEN. The techniques used for this acquisition were: brainstorming with DiSEN project members, who are familiar with the business rule of the environment; interviews with experts (project coordinator and two professors responsible for the development of technologies for DiSEN environment); formal text analysis containing DiSEN information (thesis, graduation monographs, technical reports and papers published in conference proceedings) as listed in [15].

Based on knowledge acquired from these sources, we defined the competence questions, representing the ones that the ontology needs to answer, facilitating the process of concepts definition. To define these questions, the study focused on the elements that help defining the context, which are: location, identity, activity, time and presence; and the awareness elements (who, what, when, where, why and how). In addition, questions are focused on three environmental resources categories: users, locals and tools. Examples of these issues are:

- **User**: Where is the user logged in? (local); Which roles does the user play? (identity); What tasks does the user perform? (activity); What is the status of the project? (activity); When has the user started / completed an artifact? (time); Which users of the same project are logged in? (presence)
- **Tool**: Which of the workspaces is the tool installed in? (local) What kind of artifacts does the tool generate? (identity); Which project includes the use of this tool? (activity); Who is using the tool now? (presence)
- **Local**: Who is Online? (identity); How long is the local available? (time).

Besides the competence questions, we designed a conceptual map (Fig. 1), aiming to facilitate the understanding regarding the relationships among the concepts identified during the stage of knowledge acquisition.

Based on the semantics described by the knowledge acquisition, we transcribed concepts and relationships as entities and properties, using OWL-DL language and Protégé tool. In OWL, properties are classified as object and data properties. In the object properties, a relationship between two entities is created, a range (the entity that owns the property) and a domain (possible values for the property). In the properties of data, domain is not an entity, but a data type, such as string, char, integer, date, float, among others. The following figures show, respectively, the entities set (Fig. 2), data properties (Fig. 3) and object properties (Fig. 4) of OntoDiSEN. Below we describe some of the main entities and their properties (the others are described in [8]):

- **Resource**: human resources, technological resources or support resources (office supplies, for example) that can be allocated to participate or to be used in a project. It is specialized in Tool, PhysicalResource, Local and User. It presents as data property the attribute availability. It is domain of the property hasResourceStatus, which indicates that all resources have a status. The resource status is defined by the entity ResourceStatusValuePartition;
- **User**: specialization of Resource, they are individuals who have access to the environment, thus they can be allocated to Projects, in this case, specialized in Participants. Its data properties are name, password, login, costPerHour and email. A User works in a place (isInAPlace), he/ she can be responsible for the operation of a local (managesLocal), download and install tools on his/her workspace and accesses the environment by using a workspace (accessWorkspace).
Figure 1: Main concepts and relationships of OntoDiSEN

- **Tool**: specialization of Resource, it consists of software tools that can be installed and used to create artifacts. It has a concept, which indicates its purpose, a last update date (updateDate), vendor name, platform, size and version. A Tool can generate artifacts of a particular type (generatesArtifactOfType), it can depend on another tool for its operation (dependsOnTool), it is available in a tools repository (availableAtLocal), can be installed in workspaces and used by users (installedInWorkspace).

- **Local**: specialization of Resource, they are hardware devices and machines that provide services to the environment, for example, to work as a repository of data, artifacts or tools, to connect to workspaces, to interconnect servers. Each local has a configuration (LocalConfiguration); a user responsible for ensuring its availability (managedByUser) and must be available at a physical place (availableAtPlace). A local can be a repository of artifacts or tools (localHasArtifact, localHasTool), and can have physical resources used by a project (hasPhysicalResource) (for example, a print server). Users access the environment by connecting their workspace in a local (hasWorkspace).

- **Workspace**: area that contains local Artifacts and Tools that Users handle to do their job. A Workspace is active (connectedToLocal) when it is logged (accessedByUser) and belongs to the user who is logged into it. A workspace can have copies of artifacts (workspaceHasArtifact), working as a local repository, so users can modify the artifacts. Additionally, it can have installed tools (workspaceHasTools).

Figure 2: OntoDiSEN classes
Process: a set of predetermined actions that must be followed for the development of a software product. It has name, description and version. Processes are used to drive the projects (drivesProject) and are composed of phases (hasProcessPhase).

Project: an instance of a Process that includes specific details of process execution, such as objectives to be achieved, deadlines and goals to be reached. It has name, description, estimated start and end dates, actual start and end dates. A project has participants allocated to it (hasParticipant), is composed of project Phases (hasProjectPhase) and has a project Status that indicates whether it is in planning, ongoing, completed, canceled or suspended (hasProjectStatus). The project must be driven by a process (drivenBy), respect a time zone (respectsTimeZone) and – for its implementation – will require participants who are skilled and expert at it (requiresSkill, requiresKnowledge). The difference between skills and knowledge has been defined, for DiSEN environment, by [42].

After creating entities, object properties and data properties, each entity was formally defined using a descriptive logic to introduce constraints. As an example, we decided that in order for an entity to become a user it is necessary and enough that this entity accesses a workspace and have a login and password. Moreover, we decided that a user is a resource that manages a local, is in a local, uses a tool and has only these characteristics. This definition is presented in descriptive logic, as:

\[
User \equiv 1 accessWorkspace.Workspace \cap 1 login.String \cap 1 password.String
\]

\[
User \subseteq Resource \cap 3 managesLocal.Local \cap
\]

\[
1 isInPlace.Place \cap 3 usesTool.Tool \cap 1 accessWorkspace.Workspace
\]

\[
\cap 3 managesLocal.Local \cap 3 isInPlace.Place \cap 3 usesTool.Tool
\]
OntoDiSEN plays a fundamental role in this process. A main criterion to choose an ontological model to represent the context was the ability to infer new information from it. Thus, after the original definition, new concepts were inserted in order to include two important features that allow the use of the ontology as the representation model of DiSEN-CSE: action mapping and amended context registration. These new features led OntoDiSEN to a new version called OntoDiSENv1.

The changes made to the ontology were divided into three super classes. Everything that was already defined in the first version of OntoDiSEN was classified as Context. Additionally, Actions and AmendedContext were defined to accommodate, respectively, the actions that can be performed within DiSEN, and the context information that was acquired by the sensors more recently. These new classes are detailed in the following subsections.

7.1 Context

The main goal of OntoDiSEN, since its creation, is to provide a way to represent DiSEN environment contextual information. Given this, every mapped entity helps composing the action context that can occur into this environment. To highlight this feature, the entities in the original version of the ontology became a subclass of Context. Thus, the superclass Thing has only two subclasses: Context and Action. This made it easier to sort out the elements that compose the context, and actions that can be performed according to that context.

When classes were migrated, other adjustments were made aiming to improve the entities definition in Context. Among these adjustments, the most important was the removal of ValuePartition classes. Value partition is a design pattern that aims to refine the definition of a class. Value partitions were used in the OntoDiSEN first version to classify the status of other ontology context classes, such as: File, Artifact, Project Activity, Project Phase, Project, Resource, Task, Release, Knowledge, Ability and Training. Each class was associated with a value partition. For example, the status of a project can be defined as: canceled, completed, ongoing, planning or suspended. These statuses were defined as subclasses of ProjectStatusValuePartition class. However, these subclasses have a very small granularity, because each of them represents a single individual. For this reason, we replaced the value partitions by classes containing individuals representing each of the context classes statuses listed above. The figures below show the ontology on Protégé tool, presenting how we classified the classes (Fig. 5) and properties (Fig. 6) in OntoDISENv1 after such changes.

7.2 Action

As mentioned before, the context information processing proposed in DISEN-CSE model is related to the dissemination of this information in an appropriate way to DISEN users. Actions, in the context of this study, are DiSEN services that can be invoked to send the information to its destination. Actions can be, for example, pop-ups, e-mails or tool update services. These actions are described in the ontology as subclasses of Action (Fig. 5). At first, we divided Actions into Message (SynchronousMessage and AsynchronousMessage) and Updater classes. For the examples cited, email and pop-up are, respectively, instances of SynchronousMessageAsynchronousMessage, and directly represent these services available in DISEN.
7.3 Amended Context

To assist the information processing, as proposed in the model DISEN-CSE, it is necessary to know which entities were modified since the last information processing. The class AmendedContext was included to represent these modified elements. Individuals of this class represent all properties belonging to subclasses of Context that were added after the final processing. After the use of this information, all instances of AmendedContext class are released from OntoDiSENv1. This ensures that, if there are individuals in this class, they represent the latest changes that have occurred in DISEN context.

7.4 OntoDiSENv1 impact on DISEN-CSE model

The ontology presented here was proposed in order to support the representation of information, as presented on DISEN-CSE model. The ontological model was chosen, among other reasons, because of its inference capability, assisting in information processing. The following figure (Fig. 7) shows the relationship between OntoDiSEN and DISEN-CSE model. The figure (Fig. 7) shows the elements that compose DISEN-CSE. Note that OntoDiSENv1 relates directly to the Context Representation and Processing Support elements. Into DiSEN environment, these elements consist of software agents which, respectively, manipulate the ontology – ensuring the integrity of their information – and execute the parsing rules to infer new information from the context represented by OntoDiSENv1.

The dynamics of the model occurs as described next. When a change occurs in the DiSEN environment context, information sensors (represented by Capture Support element) retrieve data about the changes and send to Context Representation element. At this point, a software agent receives the data, includes the changes in the ontology and registers new properties in AmendedContext class. For example, suppose that a project receives a new participant (contextual change). A new property hasParticipant between Project class (domain) and Participant class (range) is created. In addition, two more properties are created: (i) hasDomain property, linking a new individual that belongs to AmendedContext class to Project class, indicating that the Project is the domain of hasParticipant property in the new context; (ii) and a hasRange property, connecting the same new individual from AmendedContext to Participant class, indicating that the Participant class...
is the range of hasParticipant property represented in the new context. Thus, individuals in AmendedContext class, represent all the properties that were added after the last processing.

Once represented, information is ready to be processed. A software agent, that is in charge of the element Processing Support, is notified of the changes. Then, the ontology can be queried, to retrieve the individuals in AmendedContext class. Based on these individuals, the software agent queries a file containing the logical rules. Each of these rules has an antecedent and a consequent, which are collections of one or more ontology properties. The antecedent represents the current contextual scenario. The consequent represents the scenario that must be inferred from the available information. Both antecedents and consequents of a rule can be any ontology property, but the resulting rules defined are properties that relate to individuals of Action class. This happens to meet DISEN-CSE goal of transforming the contextual information on notification actions. Thus, when an individual of AmendedContext class satisfies any antecedent, a predefined action must be inferred. The inference of new actions is the creation of individuals of the Action class, completing the process of representation and processing.

After that, Awareness Mechanism element performs the actions inferred and represented in the ontology, invoking the DISEN services remotely. According to DISEN-CSE model, these actions contain information that is, finally, disseminated. Details on how the dissemination is performed in DISEN environment can be found in [43].

8 Final Considerations

In this paper we have presented OntoDiSENv1, an application ontology for DiSEN environment. The initial motivation for its development was the communication difficulty in a GSD environment, which is influenced by differences in language, culture and time zone. The definition of an ontology aims to incorporate the information dissemination model DISEN-CSE, a representation that reduces the ambiguity in the messages sent through DiSEN environment to project participants, allowing all individuals to have the same semantic understanding of what is being reported. After the formalization of domain knowledge - through the ontology definition - some changes were made permitting its use to share contextual information, according to DISEN-CSE model. These changes led to the current version of the ontology.

The main contribution of this paper is providing an ontology that represents the context of a distributed software development environment, allowing contextual information processing in order to infer actions to support users. By using this ontology, in association with the implementation of DISEN-CSE model, users can receive information about actions of other dispersed team members more effectively, reducing the communication gaps. This is achieved once this ontology provides the following specific contributions:

- contextual information that is disseminated by DISEN-CSE model has semantics uniformity, due to the information formalization in the ontology, before any notification action;
- the ontology represents the current context of the environment, and is also able to represent the changes since an earlier moment. This facilitates information processing, ensuring that only the latest context changes are considered when inferring new information.

The information dissemination model DISEN-CSE is being implemented as an agent-based mechanism. This mechanism is the owner of the software agents that manipulate and use OntoDiSENv1 information to capture, represent, process and disseminate contextual information. The mechanism is currently being tested and validated. When results are obtained, it will be possible to evaluate the use of ontology in a GSD scenario. Thus, it will be also possible to explicitly demonstrate the benefits brought by its use. These results will be published at a future time.

A limitation of this research is regarding the definition of inference rules. Initially, all rules are written with an action as a consequent. This condition, however, does not constrain the use of the ontology. Rules can be created using any concepts as consequent, since the concept is mapped in the ontology. Eventually, rules that do not infer actions will be modeled to evaluate the behavior of agents that manipulate the ontology. Another limitation of this study is that OntoDiSEN was developed specifically for DiSEN environment, which limits its reuse on other GSD environments. Thus, future studies involve the generalization of the ontology to involve knowledge related to GSD environments as a whole.

We are also preparing activities to conduce experimental studies for this research, specifically a feasibility study in the academic community. In this type of study data is collected according to some experimental design, but full control over all possible variables is not achieved. Running studies in the classroom allows new concepts to be tested before using them with expensive developers from industry [44]. This approach will be important to refine our research. The objective is not to find a definitive answer but to build up a body of knowledge to check the feasibility to continue the study, generating new hypothesis on the approach and its utility.

This study will be carried out according to the process described in [45], considering the following activities: Study Definition, Planning, Execution, Analysis and Packing. In the Study Definition activity, we will define the main goal, measuring goal, study goals (using the Goal-Question-Metric approach) and the questions to be answered. The
Planning prepares the study taking into account: hypothesis formulation, context selection, variables selection, participant selection and project under study. We will use two surveys: one with questions regarding participant profile; and another regarding the work under evaluation. During Execution activity, both surveys will be applied on a classroom, to students that present knowledge on the concepts related to Global Software Development domain. In the Analysis and Packing the data collected will be analyzed using Chi-square statistical test. The results obtained on this experiment will be further published.

Acknowledgements
The author Ana Paula Chaves thanks CAPES for financial support.

References


OWL-DL. OWL Web Ontology Language. [Online]. http://www.w3.org/TR/owl-features/


