Journal of Computer Science

Original Research Paper

Improving Coordination and Communication in Distributed Software Development through Context-Based Software Artifacts Awareness: A Controlled Experiment

[1]Rafael Leonardo Vivian, [2]Elisa Hatsue Moriya Huzita,
[2]Renato Balancieri, [3]Simone do Rocio Senger de Souza,
[4]Gislaine Camila Lapasini Leal and [4]Edwin Vladimir Galdamez

[1]Federal Institute Catarinense, Fraiburgo, Brazil
[2]Department of Computer Science, State University of Maringá, Maringá, Brazil
[3]Department of Computer Science, University of São Paulo, São Carlos, Brazil
[4]Department of Production Engineering, State University of Maringá, Brazil

Article history
Received: 24-04-2018
Revised: 28-04-2018
Accepted: 06-08-2018

Corresponding Author:
Gislaine Camila Lapasini Leal
Department of Production Engineering, State University of Maringá, Brazil
Email: gclleal@uem.br

Abstract: Distributed Software Development (DSD) has brought many competitive advantages, such as increased productivity, improved product quality and cost reduction. However, the geographic and temporal distances and sociocultural differences between distributed teams, expanded some challenges and, above all, added new requirements with regard to communication and coordination. This scenario has influenced on the software artifacts that are produced and/or modified, because inconsistencies and ambiguities can be generated on them. In this study, we evaluate the applicability of an approach to support the context awareness on software artifacts such as source code and class diagram in DSD. A controlled laboratory experiment was conducted with 18 participants. During the experimental study, participants used two approaches. The results were collected and analyzed with statistical methods. It was found that the proposed approach directly influences the time taken to carry out the tasks of class diagrams and source code. Although, statistically, the proposed approach has not increased the number of artifacts identified correctly during activities, there was a reduction of effort compared to the time spent in carrying out activities. Thus, the proposed approach offers adequate support for context awareness on software artifacts, thereby contributing for distributed software development mainly on coordination and communication between distributed teams.

Keywords: Context Awareness, Software Artifacts, Distributed Teams, Experiment

Introduction

Aiming to achieve competitive advantage and cooperation, several organizations have distributed their software development projects, adopting activities multisite, multicultural and sometimes globally distributed. With this, they seek to increase team productivity, improve product quality and reduce costs. In this scenario, the software development is carried out collaboratively by distributed teams, featuring the Distributed Software Development (DSD). However, this development strategy has brought challenges related to communication, coordination and cooperation for the software projects caused by the geographical and temporal distance between the teams (Herbsleb *et al*., 2000; Herbsleb and Moitra, 2001; Damian, 2002; Hargreaves and Damian, 2004; Layman *et al*., 2006; Sangwan *et al*., 2006; Jiménez *et al*., 2010; Ivcek and Galinac; Yacoub *et al*., 2016). According to Herbsleb and Moitra (2001), in distributed environments, the communication channels and the ability of developers to work together are reduced. Thus, the reduction in communication frequency directly impact in the

productivity and quality of the software development (Jiménez *et al*., 2010).

Throughout software development, the members of distributed teams work on several software artifacts, at different times, with different individuals, in different roles and set up different perspectives of their workspace (Omoronyia *et al*., 2010).

The DSD has been studied by many researchers and practitioners (Aversano *et al*., 2004; Hargreaves and Damian, 2004; Sengupta *et al*., 2006; Whitehead, 2007; Jiménez *et al*., 2009; Lanubile *et al*., 2010; Noll *et al*., 2010; Prikladnicki *et al*., 2011; Silva *et al*., 2010; Leal *et al*., 2012) and thus several methodologies and tools to support it have been proposed and produced. An essential tool to support software development with distributed teams is the Version Control System (VCS), which allows developers to contribute throughout the project development by sharing resources and also code merging (Alwis and Sillito, 2009). Another important tool for software projects is a Computer Aided Software Engineering tool (CASE), which supports the construction, manipulation and presentation of models such as Unified Modeling Language diagrams (UML) (Lahtinen and Peltonen, 2005).

The different kinds of software artifacts such as source code and class diagram have structures and distinct forms. In addition, both VCS and UML CASE tools no have mechanisms to associate the software artifacts of different kinds according to their internal structure (Vivian *et al*., 2013). For example, when a source code is changed, the individual has no knowledge about the artifacts, both the source code as the class diagram, which can be impacted or be related to its activity. This makes it difficult for members of distributed teams are aware of what is happening in the source code and class diagram file, i.e., the perception about software artifacts. Awareness was defined by Dourish and Belloti (1992) as "an understanding of the activities of others, which provides a context for the own activity of the individual". The perception is essential for the flow and naturalness of work helping to reduce sensations of impersonal working and distance, common in virtual environments (Fuks and Assis, 2001).

Another issue, relates to the circumstances involved in the production or modification of software artifacts (Vivian *et al*., 2013). Sometimes it is important know the user that manipulated an artifact, the tool used or the date when the event happened. These conditions define the context information for a given situation. In Vieira (2008), is distinguished the terms context and contextual element and thus determined that "a contextual element is any data or information to characterize an entity in a domain", while "the context is a set of contextual elements instantiated that are needed to support the execution of a task".

The geographic and temporal distances among teams make difficult the spread of contextual information about the production and/or modification of software artifacts that result from a collaborative work. The reduction of such understanding, generates impact both on production as on modification of software artifacts, that may present ambiguities and thus cause failures or uncertainties during the lifecycle of a software project. The coordination failures and communication problems between distributed team members can generate software integration matter (Cataldo *et al*., 2007). In addition, coordination problems may lead to delays in the project and also worsen the quality and increase the cost of the product (Cataldo *et al*., 2006; Blincoe, 2012). One way to detect the coordination failures and communication problems can be by the presence of duplicate or inconsistent work about the software artifacts that are produced and/or modified. Therefore, individuals must perceive the contextual information (e.g., who made certain modifications in a software artifact, where, how and when the actions happened) on the software artifacts that are produced and/or modified in a software project with distributed teams.

The above described scenario motivated the development of an infrastructure able to support the dissemination of information concerning software artifacts. Thus, to support the context awareness of software artifacts, Vivian *et al*. (2013) proposed an approach that provides resources to capture contextual information from shared repositories and, based on contextual information captured and processed, it generates relationships dependencies among software artifacts, to then, be displayed. The proposed approach aims to help distributed teams to develop activities carried out by its members with respect to the software artifacts that are produced and/or changed. With this, it is hoped that communication, cooperation and coordination problems are reduced and hence improve the clarity of the information generated throughout the software development. Thus, the expectation is to increase productivity and the quality of the software product.

This paper presents an experimental study to evaluate the feasibility of an approach developed by Vivian *et al*. (2013) to support the context awareness about software artifacts in distributed software development. During the experimental study, participants used two approaches. The results were collected and analyzed. These results suggest that DiSEN-CollaborAR approach can improve coordination and communication in distributed software development.

Context Awareness on Software Artifacts

Distributed software development has characteristics of a collaborative work and therefore requires an infrastructure able to ensure the efficient information exchange among those involved (Chaves *et al*., 2010). To this end, perception techniques combined with

contextual information can improve communication among individuals involved in a collaborative work (Herbsleb *et al*., 2000). Thus, perception mechanisms are essential to provide individuals, with contextual information about the actions that occur on entities, such as software artifacts (Vivian *et al*., 2011).

The lack awareness of context on the software artifacts can lead to ambiguities throughout the distributed software development, as well as failures or uncertainties (Gutwin *et al*., 2005; Chaves *et al*., 2010; Steinmacher *et al*., 2012). According to Gutwin *et al*. (2004), the perception is essential for distributed teams coordinate their efforts, add code without causing problems, make changes that affect other parts of the code and avoid rework.

According to Jiménez *et al*. (2009), studies and literature related, combining DSD and awareness have increased. A systematic review presented by Vivian *et al*. (2011) aims to identify and analyze techniques for capture and dissemination of contextual information that have been proposed and used on creating artifacts in distributed software development.

In this review were identified and analyzed the following tools:

- Palantír: Supports the perception of artifacts for developers using Configuration Management Systems (CMS) (Sarma *et al*., 2003)
- Ariadne: Presents the socio-technical relationship among the artifacts. It increases the perception of developers about the social dependencies of their work (de Souza *et al*., 2004)
- Augur: Provides information on artifacts structure and their related activities in distributed software development (Froehlich and Dourish, 2004)
- ADAMS - ADvanced Artefact Management System: Supports traceability and change management of artifacts during software development (de Lucia *et al*., 2005)
- ProjectWatcher: supports the perception of the activities in distributed software development projects (Gutwin *et al*., 2005)
- EvolTrack: supports the perception of software evolution throughout development cycle (Cepêda *et al*., 2010)

The main aspects identified in these tools were:

- *Information Source:* Software artifacts can be generated from several tools and stored in different repositories. So, VCS, development environments, change control system (bug tracking system) and continuous integration are important sources. Thus, the software artifacts carry important contextual information
- *Artifact type:* Software artifacts have a variety of formats, including source code, diagram and documentation
- *Information Type:* Context information (e.g., historical of changes, relationships between artifacts and artifact structure), awareness elements and properties of software artifacts (e.g., traceability, filter and search for information) are important to raise awareness of distributed team members
- *Information Analysis:* The contextual information captured can be represented and processed to detect patterns or relationships, or can be inferred for new information
- *Information Presentation:* Visual resources (e.g., graph, colors and timeline) can support the presentation of contextual information and increase awareness of individuals
- *Presentation Location:* The perception mechanism can be integrated into the development environment (e.g., Eclipse IDE) or may be an independent application

Overview of the DiSEN-CollaborAR Approach

The DiSEN-CollaborAR approach was designed to support context awareness on the software artifacts by distributed teams. It presents an infrastructure for the individual to visualize the contributions from other developers regarding to information and dependencies between software artifacts. With this approach contextual information about the software artifacts, such as the circumstances of the moment that a software artifact has been produced and/or modified (e.g., contextual information such as version, date, tool, author, staff, location) can be managed and also allow that these information flow between distributed teams. The contextual information captured from software artifacts are represented, stored, processed and presented for individuals. This section briefly presents the approach, whose details can be found in (Vivian *et al*., 2013).

As presented on Fig. 1, the DiSEN-CollaborAR approach can be analyzed from four structures (Vivian *et al*., 2013): (i) Workspace: is the workspace of the individual who is part of a distributed team, consisting of Version Control System and UML CASE tool; (ii) Shared Repository: it includes the shared repository of software artifacts such as Code Repository and Model Repository. In addition, the context repository which is responsible for storing contextual information; (iii) Mechanism: Includes Support to Capture, Representation of Context, Support to the Processing, Tracking and Perception Mechanism; and (iv) Visual Component: is the visual component to the perception in the workspace of the individual through a software artifacts network with their contextual information and relationships of dependencies among artifacts.

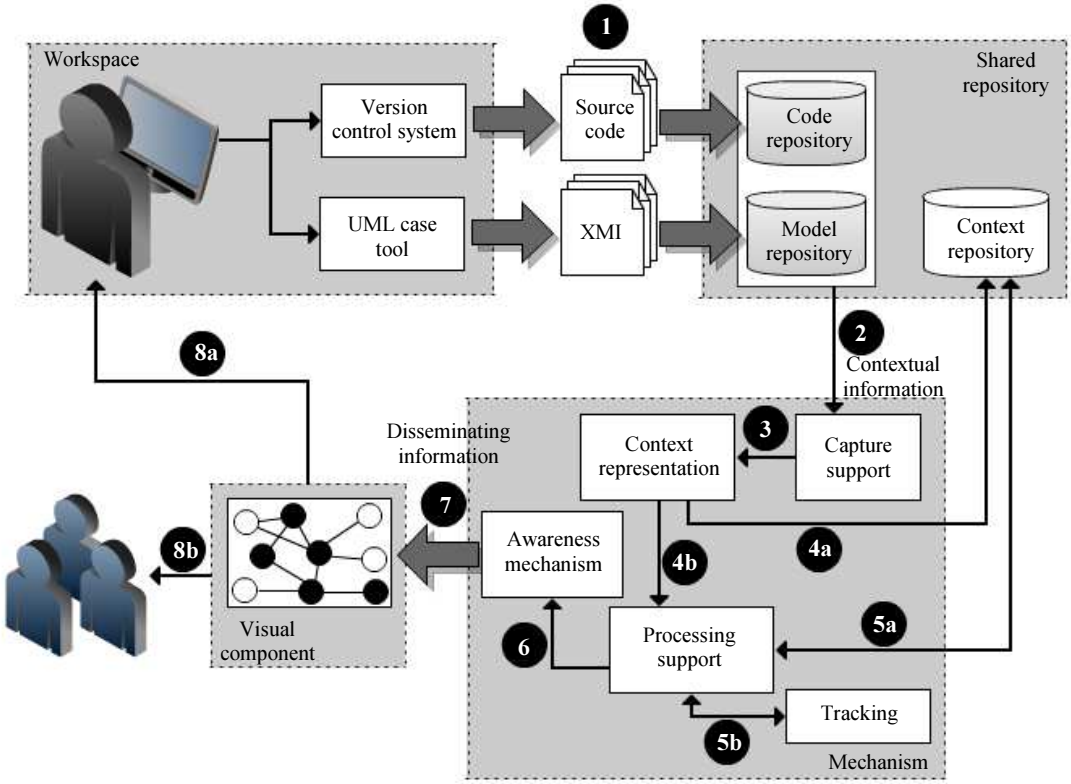

Fig. 1: Conceptual model of DiSEN-CollaborAR (Vivian *et al*., 2013)

Thus, DiSEN-CollaborAR approach has the following execution stages, as shown on Fig. 1. (1) The source code is stored in the Code Repository and the class diagram is exported in XMI format and stored in the Model Repository. (2) Changes occurring on the software artifacts that are in the repositories are captured by Capture Support. (3) The contextual information captured are mapped by Context Representation for a formal representation model based on ontology. (4a) The contextual information represented should be stored in the Context Repository, creating a historical of context information. In addition, (4b) contextual information can be sent for Processing Support to infer implicit contexts. (5a) The Processing Support can send the processed information for Context Repository or retrieve the information stored in it. (5b) The contextual information are sent for Tracking to generate dependencies relationship between software artifacts. (6) The contextual information about the software artifacts are made available for Perception Mechanism. (7) The contextual information about the software artifacts are presented by a graph. Finally, (8a and 8b) members of distributed teams realize the contextual information and the relationships dependencies between software artifacts.

The DiSEN-CollaborAR approach combines some positive aspects of other approaches and tools Sarma *et al*. (2003), de Souza *et al*. (2004), Froehlich and Dourish (2004), de Lucia *et al*. (2005), Vivian *et al*. (2011) and Cepêda *et al*. (2010) to support the context awareness about software artifacts, such as capturing contextual information from Version Control System and UML CASE Tool and the presentation by graphs. However, DiSEN-CollaborAR approach explores also other challenges such as the combination of traceability links between different types of software artifacts - source code and class diagram. Furthermore, this approach explores the semantic representation of contextual information about the software artifacts by means of an ontology (Chaves *et al*., (2011). This allows the automatic generation of dependency relationships between software artifacts, according to the structural information and semantic found in own artifacts. So the DiSEN-CollaborAR offers an infrastructure for that the members of distributed software projects, can be aware of the contributions made for others developers on the source code and class diagram from contextual information related to them.

In order to verify the viability of DiSEN-CollaborAR approach, a prototype called ACAS (Artifact Collaborative Awareness System) was implemented (Vivian *et al*., 2013). It helps individuals in carrying out tasks, providing resources to support the context awareness on the software artifacts. Figure 2 shows the user interface of ACAS prototype.

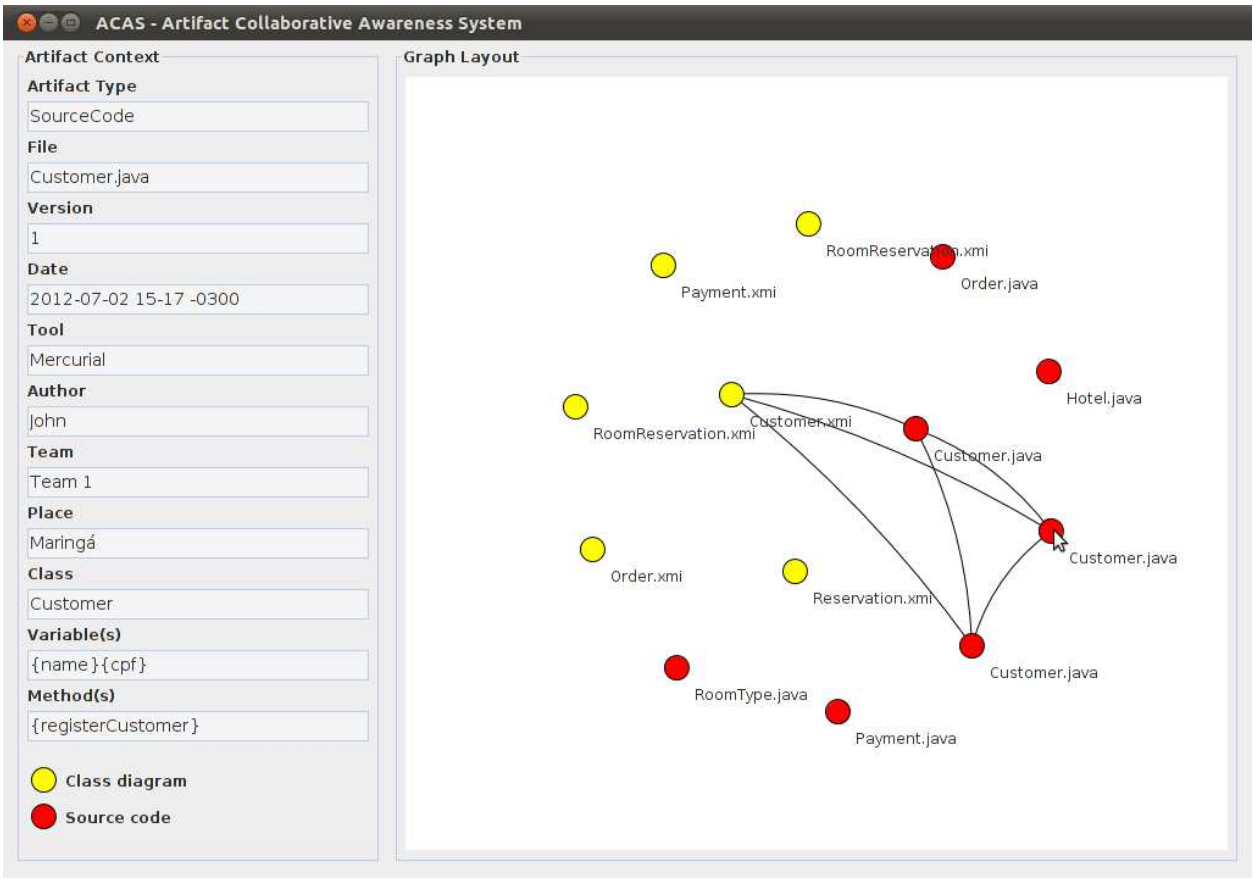

Fig. 2: User interface on ACAS (Lahtinen and Peltonen, 2005)

The ACAS tool has integration with the Mercurial version control system. When an event occurs, such as commit or push, a hook script is run and information about the event are captured. Regarding UML CASE Tool, ACAS has integration with ArgoUML. When an event occurs, like saving a XMI file, the information is captured via a shell script that monitors such event. The parser in the code that represents the class diagram in XMI, is performed by API SAX. The framework DiSEN Agency provides support for the processing of contextual information in the ACAS (Monte-Alto *et al*., 2012). The DiSEN Agency is a framework to support the development of knowledge-based multi-agent systems. For handling graphs in ACAS, the JUNG framework was used - Java Universal Network/Graph that provides an API for the creation, manipulation and visualization of data represented as graphs or networks.

Experimental Study

A controlled experiment that considers the proposal and, an evaluation of the approach regarding to the viability of its application in distributed software development environments was performed in a laboratory. It is noteworthy that the ACAS prototype, was used in this experimental study to support implementation of activities and therefore be possible to evaluate the DiSEN-CollaborAR approach. The experimental study was conducted in four phases: (1) Definition, (2) planning, (3) operation and (4) analysis and interpretation (Wohlin *et al*., 2000).

Definition

Thus, to assess whether approach supports the context awareness of software artifacts in distributed teams, it is necessary to analyze the information of the structure of software artifacts together with the information of activities performed by human at distributed software development process. Thus, several aspects of approach can be evaluated: (1) Efficacy, (2) ability to visual presentation, (3) performance and (4) usability. In this experimental study was evaluated only the aspects "1" and "2": Effectiveness because it should provide insight into the actions taken on the software artifacts; and ability to visual presentation because the information related to software artifacts should be presented visually to facilitate the context awareness on the software artifacts by individuals.

In accordance with the principles of GQM (Basili *et al*., 1994), this experimental study was defined as follows:

> **Analyze** the DiSEN-CollaborAR approach compared to an ad hoc approach.
> **In order** to assess/characterize.
> **With respect to** understanding and knowledge by individuals on the software artifacts that are produced and/or modified during the distributed development.
> **From the viewpoint** of the researcher.
> **In the context of** academics.

The following research questions are posed to aim to achieve the presented research global:

Q1: The adoption of DiSEN-CollaborAR approach increases the perception of software engineers on the software artifacts that are produced or modified when compared to ad hoc approach?
Q2: The adoption of DiSEN-CollaborAR approach reduces the effort during activities of software development with distributed teams compared to ad hoc approach?
Q3: The adoption of DiSEN-CollaborAR approach increases the amount of artifacts correctly identified during software development activities with distributed teams compared to ad hoc approach?
Q4: The adoption of DiSEN-CollaborAR approach reduces the complexity in the tasks during software development with distributed teams when compared to ad hoc approach?
Q5: The contextual information about the software artifacts presented by DiSEN-CollaborAR approach is sufficient for the individual's perception?
Q6: The traceability among software artifacts presented by DiSEN-CollaborAR approach is useful for the context awareness?

Planning

This phase describes the plan to conduct out the experiment and consists of the following elements: definition of hypotheses, instrumentation description, context selection, selection of subjects, variable selection, experimental design and validity.

Definition of Hypotheses

The null and alternative hypotheses formulated for this experimental study were:

Null hypothesis (H0): The adoption of DiSEN-CollaborAR approach does not increase the context awareness on the software artifacts by members of distributed teams when compared to an ad hoc approach adoption. This means that:

(H01): There is no effort difference to perform the activities using DiSEN-CollaborAR when compared to ad hoc approach.
(H02): There is no difference of amount of artifacts correctly identified adopting DiSEN-CollaborAR when compared to ad hoc approach

Alternative hypothesis (H1): The adoption of DiSEN-CollaborAR reduces effort during software development activities with distributed teams when compared to an ad hoc approach adoption.

Alternative hypothesis (H2): The adoption of DiSEN-CollaborAR increases the amount of artifacts correctly identified during software development activities with distributed teams if compared to an ad hoc approach adoption.

Description of Instrumentation

To perform this experimental study, the instrumentation included a scenario of distributed development of a Hotel Management system. So, for such scenario, the teams adopted the DiSEN-CollaborAR approach and also ad hoc approach. In addition, the experiment presented the following instruments: (i) A consent form to experimental study; (ii) participant's characterization questionnaire; (iii) document describing the system scenario to be developed; (iv) class diagram of the scenario used in the experiment; (v) task lists to be performed by the participants; (vi) assessment questionnaire for qualitative analysis.

Context Selection

This experimental study assumes the offline process because the activities were carried out in the laboratory and in a day predetermined for its accomplishment. Participants were undergraduate and graduate students of Computer Science of Computer Department at State University of Maringá (DIN-UEM) and of Mathematics and Computer Science Institute at University of São Paulo (ICMC-USP). The generality of the study is specific because the experimental results are valid for distributed software development context.

In the experiment, participants were responsible for the production and modification of software artifacts related to a hotel management system. Such kind system provides functionalities to streamline the activities in hotels such as reception desk, reservations, guests control, rooms, daily and payments. Somehow, this type of system does not require advanced knowledge for its understanding and development, requiring only basic knowledge about object-oriented design and implementation. The UML and Java languages were adopted respectively for modeling and implementation in this scenario.

Selection of Objects

Undergraduates and graduate students in Computer Science from the DIN-UEM and ICMC-USP were selected as participants for this experimental study. It was assumed that these individuals were available for the study and had knowledge of software development. Participants completed a questionnaire that aimed to characterize their training from an academic viewpoint, experience and expertise to analyze the data and reducing the bias. Furthermore, participants were prepared by a training carried out before the experimentation.

Variable Selection

The independent variables identified for this study were: (i) the DiSEN-CollaborAR approach; (ii) an ad hoc approach; (iii) experience and knowledge of the participants; characterization of the application domain (Hotel Management System). As dependent variables were: (i) time taken to perform the activities; (ii) number of artifacts identified correctly; (iii) number of indications concerning to reduce the degree of complexity of the tasks; (iii) number of indications in regard to the adequacy of contextual information; (iv) number of indications with respect to the usefulness of traceability among software artifacts.

Experimental Design

The design of this experimental study involved two factors: (1) Context awareness of software artifacts and (2) application domain.

This experimental study compared a distributed software development scenario "with" and "without" the adoption of DiSEN-CollaborAR approach. Thus, the context awareness factor on software artifacts presented two treatments:

- **Context awareness of software artifacts adopting DiSEN-CollaborAR approach:** The participants adopted the approach, consisting of Version Control System, UML CASE Tool and ACAS prototype
- **Context awareness of software artifacts adopting an ad hoc approach:** The participants adopted only Version Control System and UML CASE tool

The domain factor had the following scenario:

- **Development of a Hotel Management system:** The participants produced and modified software artifacts related to the classes of a system that provides resources for activity management in hotels

The selection of participants was not randomly within the universe of candidates. The groups were made up according to their location, in this case Maringa - at Paraná state - and São Carlos - at São Paulo state. Thus, this experiment consisted of a group of 8 people in Maringa and a group of 10 people in São Carlos, which were observed by a moderator.

The experimental procedure presented two sessions. First, in Session 1 the two groups performed the activities using an ad hoc approach. Then, in Session 2 the two groups performed the same activities, but adopting the DiSEN-CollaborAR approach.

At first, people were given the consent form and, if they agreed to participate in the experiment, responded Participant characterization questionnaire. The data gathered were used to interpret the results obtained by individuals. Then the research topic was introduced and a short training section conducted. The objective of this training was to present the DiSEN-CollaborAR and ad hoc approaches for the groups aiming to familiarize themselves with their features. In addition, teams were given a document outlining the scenario featuring the development of a hotel management system.

During the experiment, the teams received the Tasks List and some software artifacts. Individuals in both treatments - DiSEN-CollaborAR and ad hoc- received diagrams and source code of some system classes and then held changes and new software artifacts produced. Participants recorded the start and end time, of the job tasks, in the Tasks List. In addition, participants recorded in the Tasks List the artifacts produced and / or changed in carrying out activities.

At the end of the simulation, an evaluation questionnaire about the experiment with the DiSEN-CollaborAR approach and the perception during the software development was applied. In both approaches, ad hoc and DiSEN-CollaborAR, teams were given the same scenario and set of modeling and implementation activities.

It is important to note that prior to the actual execution of the experiment, a "pilot experiment" was undertaken to assess the instrumentation used in this experimental study. For this end, three individuals - graduate students DIN-UEM who were not aware of the research questions - were invited, including the same instruments of the experimental study. The data obtained by the pilot experiment were not used to supplement this study.

Validity

The threats of validity identified in this experimental study were (Wohlin *et al*., 2000):

Internal validity: For the selection of individuals, this experimental study used students from the Computer Science course, which usually tend to develop software in academic level. Students are an important mechanism to conducting pilot studies in software engineering (Salman *et al*., 2015). To reduce the influence of threats to internal

validity, such as the fact that one group has more knowledge and experience and therefore perform better, regardless of the approach taken, participants from both groups performed the tasks adopting both approaches in different sessions. However, the order of application of approaches can influence the results because the participants after performing the Session 1, may present a greater learning for holding the Session 2. In the experimental study consent form was included the confidentiality of relevant information on the experiment.

External validity: The study participants, generally, were considered representative for the population of software developers in academic level. Although this is a controlled laboratory experiment, in this study the environment and the characteristics were simulated to be as closely as possible with the industrial practice. However, the external validity may be compromised because the experiment was conducted on a specific scenario and using students, that could threaten to generalize the results to other case studies and the industry. However studies like this have the potential to increment build knowledge and contribute to the body of evidence (Salman *et al*., 2015). Other threats to external validity are: The adoption of the Java programming language and geographical location.

Construct validity: Construct validity was achieved, since the DiSEN-CollaborAR approach and the scenario used in the experiment, at the level where it was applied, did not require substantial experience of the participants. However, the validity of construction could be undermined if a participant had experience in developing such systems. With the use of two groups of participants, we attempted to generalize the results to a real scenario, although the results cannot be generalized as they are influenced by groups of participants and the chosen study. This validity was achieved, since the DiSEN-CollaborAR approach and the scenario used in the experiment, the level in which was applied, did not require extensive experience of the participants. However, note that the validity of construction could be undermined if a participant had experience in developing such systems.

Conclusion validity: This validity is reached because the data analysis was performed using descriptive statistics and hypothesis testing with nonparametric statistical method to determine the completion of the study. However, the amount of individuals - 18 persons - who participated in the study was low. Furthermore, only 10 individuals with DSD knowledge can also be considered a low number.

Operation

This phase presents the experiment application and consists of the following elements: Preparation, implementation and validation.

Preparation

The subjects were eighteen students of Computer Science course of DIN-UEM and ICMC-USP, as follows: 6 undergraduate, 8 graduate students, one master and three doctoral students. Students were informed that would investigated the result of applying an approach in distributed software development. However, they were not aware about which aspects would be studied nor knowledge of what were the stated assumptions. All students had the guarantee of anonymity. All instruments of experiment were ready and were therefore provided to participants before experiment.

Implementation

The experiment was conducted at the Distributed Software Development Laboratory (LDDS) DIN-UEM and Software Engineering Laboratory (LABES) the ICMC-USP. The participants signed a consent form that explained about the overall purpose of the study and authorized that the artifacts produced were used in this experimental study. In addition, participants filled out a characterization form to assess their knowledge and experience levels in DSD, Java, UML, VCS and Case Tool UML.

All students had expertise in systems modeling and implementation. However, only seven of them had experience in industrial projects. Table 1 summarizes the participants profile.

Initially, participants received training on the concepts and approaches would be adopted. Participants were clustered into two groups (A and B) according to their location (Maringá and São Carlos). During the Session 1, the groups accomplished the activities with the ad hoc approach. Then, during Session 2, the groups adopted the DiSEN-CollaborAR approach.

During operation, the participants performed a sequence of tasks, recorded the start and end time and the name of the artifacts produced and/or changed.

Validation

Data were collected from the characterization questionnaire, task lists and evaluation questionnaire from 18 students. These data were not considered as invalid or questionable. Thus, none data of participants was removed. Therefore, all participants were considered for statistical analysis and interpretation of results.

Analysis and Interpretation

After the operation of the experiment, data were collected and analyzed following the procedures defined in the study planning. The data were analyzed in a computational environment for statistical analysis, called R. This section presents descriptive statistics, hypothesis testing and qualitative analysis.

Table 1: Participants profile

			Group A								Group B									
Question			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	10
1	Knowledge	In DSD	1	1	0	1	0	1	0	2	0	1	1	0	0	0	2	0	0	0
2	(0 = none,	In Java	3	1	2	2	1	1	2	3	1	2	2	3	3	2	1	2	1	1
3	1 = basic, 2 = intermediate, 3 = advanced)	In UML	2	1	2	2	1	2	1	2	1	1	1	1	2	1	1	1	2	1
4	Experience (0 = none, 1 = basic,	With Version Control System	3	1	0	1	0	1	0	2	1	1	1	1	1	1	3	2	0	0
5	2 = intermediate, 3 = advanced)	With UML CASE Tool	2	1	1	2	1	2	1	2	1	1	1	1	2	1	0	1	1	1

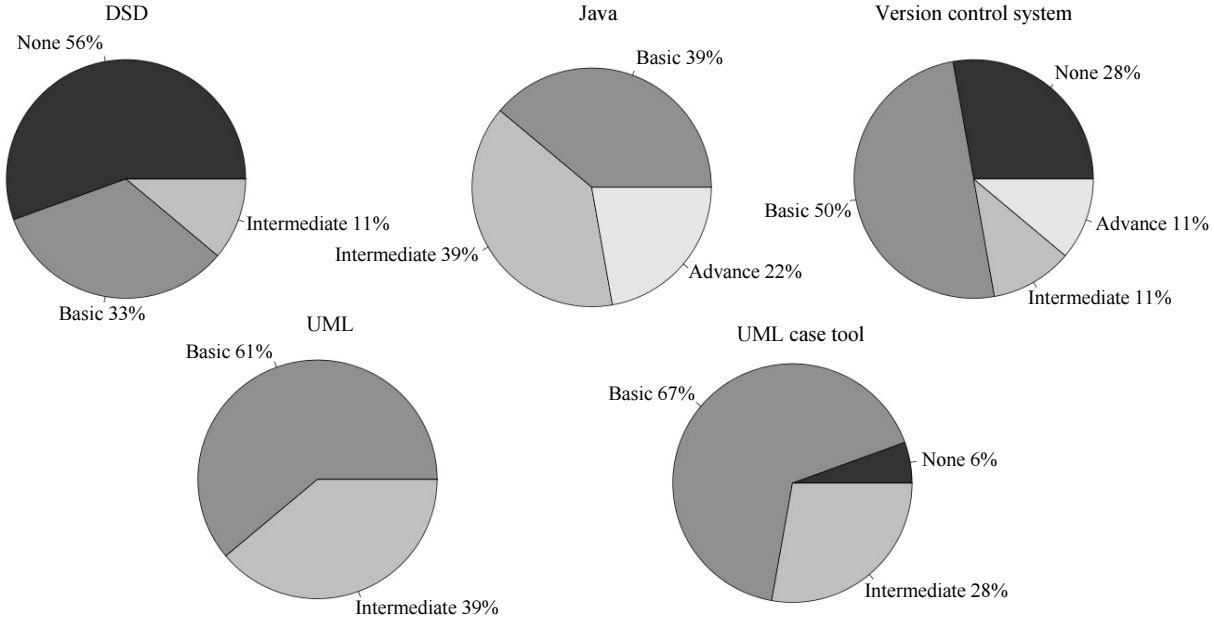

Fig. 3: Knowledge of participants

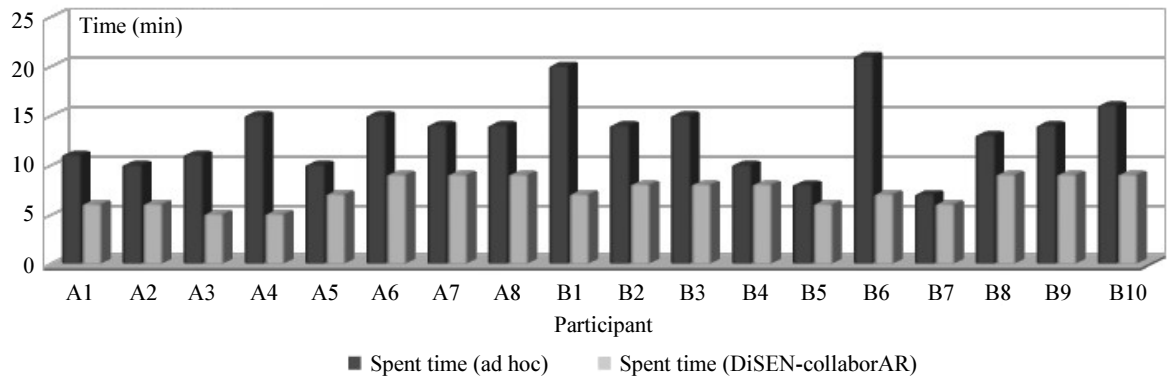

Fig. 4: Time spent to carry out the tasks per participant and approach

Descriptive Statistics

Descriptive statistic was used to improve the understanding and visualization of data collected. Figure 3 shows the percentage of occurrence of data, in this case, about the participants knowledge.

The time taken to perform the tasks and the amount of artifacts identified correctly by the participants are shown on Fig. 4. One can visualize that in order to perform the tasks, the adoption of ad hoc approach consumed a longer time when compared to the adoption of DiSEN-CollaborAR approach.

Descriptive statistics provided an overview of the data, in terms of what could be expected from the hypothesis testing.

Hypothesis Testing

First of all, was performed the test of normality using the Shapiro-Wilk test. The Shapiro-Wilk test is the most suitable to be used for identifying normality in variables with less than 50 values (Araújo and Travassos, 2009). In the case of this experimental study are 36 values. Table 2 presents the test results for these variables.

For the variable time spent, it was found that the data distribution was not normal, because a value of p-value 0.006509 is less than the value indicated for normal distribution of this method is 0.05. Thus, for this variable in the hypothesis test the nonparametric Kruskal-Wallis method was used. For the variable number of artifacts, it was found that the data distribution was not normal, because the p-value 0.000000000002090 is less than the value indicated for normal distribution of this method is 0.05. Thus, for this variable in the hypothesis test the nonparametric Kruskal-Wallis method was used too.

Applying the Kruskal-Wallis nonparametric statistical method, one can realizes that there is a statistical difference in the adoption of DiSEN-CollaborAR approach when compared to the ad hoc approach, as shown by the p-value which stood at 0.000004626, lower than the value 0.05 (Table 3).

The null hypothesis is that the approach time per are identical population. Applying the Kruskal-Wallis method to compare the independent data, the p-value is almost zero (p-value = 0.000004626) (i.e., p-value <0.05). Thus, at the 0.05 significance level, reject the null hypothesis and can conclude that the approach time per population are not identical.

Applying the Kruskal-Wallis nonparametric statistical method one can observes that there is no statistical difference in the adoption of DiSEN-CollaborAR approach if compared to the ad hoc approach, as can be seen by the p-value was at 0.1513, higher than the value 0.05 (Table 4).

The null hypothesis is that the number of artifacts per approach are population identical. Applying the Kruskal-Wallis method to compare the independent data, the p-value is 0.1513 (e.g., p-value >0.05). Thus, the 0.05 significance level, accepts the null hypothesis and one may conclude that the number of artifacts per approach are population identical.

Qualitative Analysis

The issues Q4, Q5 and Q6 do not exhibit the respective hypotheses and therefore have no hypothesis testing. This is due to the fact that such questions have answers based on the opinion of the participants of the experiment. Thus, these issues must be evaluated through a qualitative analysis of the DiSEN-CollaborAR approach. Thus, the purpose of this analysis is to identify whether the adoption of DiSEN-CollaborAR approach has: (1) Indications that the degree of complexity in carrying out tasks is reduced, (2) indications that contextual information shown on are sufficient and (3) indications that the traceability among software artifacts is useful to support the context awareness.

Table 2: Normality Test Results Shapiro-Wilk

Variable	W	*p-value*	
Time spent	0.9101	0.006509	Non-normal distribution (*p-value* < 0.05)
Number of artifacts	0.2455	0.000000000002090	Non-normal distribution (*p-value* < 0.05)

Table 3: Results of Kruskal-Wallis hypothesis test

Date	Degrees of freedom (df)	*p-value*
Time-based approach	1	0.000004626

Table 4: Results of Kruskal-Wallis hypothesis test

Data	Degrees of freedom (df)	*p-value*
Number of artifacts for approach	1	0.1513

Regarding the complexity to perform tasks adopting ad hoc approach (Q4), 11.1% of participants considered very complex, 22.2% considered complex, 55.6% considered simple and 11.1% considered very simple. When they were asked about the complexity in carrying out the tasks adopting DiSEN-CollaborAR approach, none considered very complex, 5.5% considered complex, 61.2% considered simple and 33.3% considered very simple. By generalizing, 94.5% of participants considered simple or very simple the degree of complexity to perform tasks adopting DiSEN-CollaborAR approach. Thus is verified that the adoption of DiSEN-CollaborAR approach reduces the degree of complexity in carrying out tasks by the participants.

When asked if the contextual information about the software artifacts adopting DiSEN-CollaborAR approach were sufficient (Q5), 88.9% indicated yes and 11.1% indicated partially. Thus, it can be evidenced that the contextual information about the software artifacts presented by DiSEN-CollaborAR approach are sufficient. Regarding the traceability utility of software artifacts by adopting DiSEN-CollaborAR approach (Q6), 100% indicated yes. Thus, it is noted that, with the adoption of DiSEN-CollaborAR, the traceability among software artifacts is useful to support the perception of context on them.

Discussion

Regarding the hypothesis presented the null hypothesis (H0) was rejected, the alternative Hypothesis (H1) was accepted and the alternative Hypothesis (H2) was refuted. According to hypothesis testing, the adoption of DiSEN-CollaborAR approach reduces the time taken to carry out the activities, but does not increase the number of artifacts identified correctly.

The results suggest that DiSEN-CollaborAR approach does not influence in the number of artifacts identified correctly in relation to the adoption of the ad hoc approach. From the data analysis, it turns out that adopting DiSEN-CollaborAR approach the average number of artifacts identified correctly is higher. However, statistically there is no difference when comparing the two approaches.

In analyzing the results in respect to hypothesis, can be identified that when DiSEN-CollaborAR approach is adopted it results in an increase of context awareness on software artifacts. As mentioned earlier, the qualitative analysis sought to identify if the adoption of DiSEN-CollaborAR approach provides: (1) Indications that the degree of complexity in carrying out the tasks is reduced, (2) indications that the contextual information shown on are sufficient and (3) indications that the traceability between software artifacts is useful to support the perception of context. Analyzing the answers to the questions, it is observed that there are positive indications for such qualitative analysis.

With the adoption of DiSEN-CollaborAR approach, the complexity in carrying out the tasks, both in production and in the modification of software artifacts (diagram classes and source code) are reduced. This is due to the fact that the approach, through the ACAS prototype tool, presents for the individual the contextual information about the software artifacts.

The contextual information provided by DiSEN-CollaborAR approach showed to be enough to support the context awareness on software artifacts during its production and/or modification by distributed teams. The approach offers support to understanding and knowledge, by individuals, of the circumstances involved in a particular situation about the software artifact.

Through the experiment, it was found that the traceability between software artifacts, provided by DiSEN-CollaborAR approach also showed useful to support the context awareness in the DSD. Traceability links provide support for software engineers understand the relationships and dependencies between software artifacts generated during the software development process (Zhang *et al*., 2008). Thus, when two people are working on two different classes, joined by some type of relationship, for example, there is the possibility that the activities of these individuals are also related, as their software artifacts are. In addition, the visuals resources have an important role, because they present information on the software artifacts, aiming to raise awareness of individuals. To do this, from the traceability between software artifacts, the approach uses a network of software artifacts associated to present the contextual information. This presentation is based on a graph, with vertices representing software artifacts - source code and class diagram - and edges representing the dependencies that exist between these software artifacts. For example, when a source code is sent to a code repository or a class diagram is stored in model repository for an individual, such actions are reflected on the graph presentation. Thus, the software artifacts network is shared among all team members, allowing the context awareness on the software artifacts.

The feedback received from the participants of the experiment indicates that this is an interesting approach. In addition, some suggestions for improving the approach and prototype tools were made. Among the suggestions we can mention: (i) Add a filter to select the software artifacts according to some criterion; (ii) automatically update the graph in real time; (iii) open the software artifact from a click on the vertex of the graph; (iv) use the own development environment, such as IDE and UML CASE tool to present the information; (v) add a mechanism which automatically change the source code based on the associated class diagram and vice versa.

Some limitations on technology related to Ontology and restrictions on access to internal network of universities (UEM and USP), hindered some resources that could be used. For example, the groups - A and B - were unable to simultaneously perform activities because of restrictions on the ontology affecting the operation of the ACAS prototype. At the time of implementation of ACAS, there was not also a consolidated mechanism for concurrency control the ontologies, such as a Database Management System. Technologies about this are in development, including works by other members of the Research Group on Distributed Software Engineering DIN-UEM, to offer support to such issue. Thus, this does affect the operation of processing the contextual information to the deployed prototype, the same used as a mechanism for extending the framework DiSEN Agency proposed by Monte-Alto *et al*. (2012) for performing the process of inference. Thus, such a mechanism has limitations on the concurrent access and, at present, does not offer full support to the above mentioned issue. This is an important factor to be considered in a scenario with distributed teams, since many individuals perform activities simultaneously. It is therefore necessary to control access and concurrent updates to the knowledge base, so as not to lead to inconsistencies. Thus, members of a group might not see the artifacts produced and/or modified by the other group and also carry out direct communication with other participants through ACAS prototype. So, the increased context awareness about software artifacts by participants, checked through the experiment, has evidence that this approach can improve also the collaboration, communication and coordination.

This experimental study was a first step towards a more complete assessment of the DiSEN-CollaborAR approach (Vivian *et al*., 2013). The experiment is limited in some respects, as described above, restricting the generalization of the results. Therefore, the approach and the prototype can be improved in several points, like describe earlier. It can be concluded that the DiSEN-CollaborAR approach has direct influence on the time taken to carry out the tasks, specifically, in class diagrams and source code. Thus, it is an approach that helps the distributed software development, with support for the context awareness of such software artifacts. This approach contributes to improve the coordination and communication between distributed teams. Furthermore, the approach offers features that can increase the productivity and quality of software developed by distributed teams.

Conclusion

In this study was presented DiSEN-CollaborAR, an approach that supports the dissemination of information about the software artifacts (Lahtinen and Peltonen, 2005). This approach helps to improve communication and coordination among distributed teams and thus reduce ambiguities in software artifacts.

An assessment of DiSEN-CollaborAR approach is presented, using ACAS prototype tool by Lahtinen and Peltonen (2005), which help individuals in carrying out tasks, by providing resources to support the context awareness on software artifacts. Thus, it was possible to realize a controlled experiment in the laboratory and then collect and analyze the data.

The experimental study, although limited, indicated that DiSEN-CollaborAR increases the context awareness on software artifacts by individuals while carrying out their activities. Although, statistically, this approach has not increased the number of artifacts identified correctly during activities, there was a reduction of effort during activities.

Therefore, one can highlight as contributions of DiSEN-CollaborAR approach: the construction of an infrastructure to support distributed software development so that context information may contribute to the perception of individuals and thus promote communication and coordination between distributed teams. So, it can avoid ambiguity in the distributed software development, as well as failures or uncertainties that affect the team work as a whole. Thus, by adopting this approach can be avoided problems of coordination, which may cause delays in the project and also worsen the quality and increase the cost of the product, as cited by Cataldo *et al*. (2007). In addition, the storage of all information about the software artifacts that are produced and/or modified by distributed teams produce a memory of group that can be consulted at any time by other individuals. This information can help, for example, the transfer of skills, competencies and knowledge to others. Also, this approach becomes useful in allowing individuals to make decisions and develop their artifacts based on artifacts already analyzed, commented and stored by other individuals in the DSD.

Another contribution is the possibility of holding new findings about the domain from the software artifacts produced and/or modified. Moreover, can better exploit the reuse of software artifacts, in which individuals can post comments about their experiences on their use. Also, from the DiSEN-CollaborAR approach, it is possible to promote further integration between individuals, as well as the way to act of each of them and thereby lead to a more active and participatory vision of this individual in the workplace.

As future work can be highlighted: (i) Explore other information sources since several tools can generate artifacts during the software development cycle; (ii) consider other types of software artifacts such as use case diagram, sequence and package diagram,

requirements and validation documents and description of the system architecture and also offer support for other programming languages; (iii) incorporate an information filter mechanism in the prototype; (iv) identify socio-technical networks from software artifacts. Team members of distributed software project are connected by inter-related software artifacts, constituting a social network of developers. These networks can reveal patterns of collaboration and communication that influence the perception of individuals. Socio-technical networks have been explored by de Souza *et al*. (2004) from only one type of software artifact - source code. However, it would be interesting the generation of socio-technical networks based on various types of software artifacts.

Finally, it is necessary to reflect on the use of DiSEN-CollaborAR approach for business purposes. It is believed that this approach is suitable for the industry, since it can assist in the coordination and communication, it can also be used in real projects to reduce the difficulties caused by the DSD geographical and temporal distances. However, it is necessary and important carry out a case study in the industry to determine the feasibility of the approach in a real scenario of a project with distributed teams.

Acknowledgement

The authors thank for Brazil's National Council of Scientific Development (CNPq) and Araucaria Foundation for funding the research project.

Authors Contribution

Rafael Leonardo Vivian: Designed the experimental study, executed the experimental study, elaborated the discussion of the work, article writing, conducted the analysis and interpretation of data.

Elisa Hatsue Moriya Huzita: Designed the experimental study, elaborated the discussion of the work, article writing, conducted the analysis and interpretation of data.

Renato Balancieri: Designed the experimental study, elaborated the discussion of the work, conducted the analysis and interpretation of data, article writing.

Simone do Rocio Senger de Souza: Designed the experimental study, elaborated the discussion of the work.

Gislaine Camila Lapasini Leal: Designed the experimental study, executed the experimental study elaborated the discussion of the work, article writing, conducted the analysis and interpretation of data.

Edwin Vladimir Galdamez: Designed the experimental study, elaborated the discussion of the work, article writing, conducted the analysis and interpretation of data.

Ethics

Authors confirm that this manuscript has not been published elsewhere and that no ethical issues are involved.

Conflict of Interest Declaration

Authors declare that there is no conflict of interest regarding the publication of this manuscript.

References

Alwis, B. and J. Sillito, 2009. Why are software projects moving from centralized to decentralized version control systems? Proceedings of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering, May 17-17, IEEE Xplore Press, Vancouver, BC, Canada, pp: 36-39. DOI: 10.1109/CHASE.2009.5071408

Araújo, M.A.P. and G.H. Travassos, 2009. The use of statistical methods for planning and analyzing experimental studies in software engineering area. Proceedings of 6th Experimental Software Engineering Latin American Workshop, (LAW' 09), pp: 10-10.

Aversano, L., A. Lucia, M. Gaeta, P. Ritrovato and S. Stefanucci *et al*., 2004. Managing coordination and cooperation in distributed software processes: The GENESIS environment. Software Process: Improvement Pract., 9: 239-263. DOI: 10.1002/spip.206

Basili, V.R., G. Caldeira and H.D. Rombach, 1994. Goal question metric paradigm. Encyclopedia Software Eng., 2: 527-532.

Blincoe, K., 2012. Timely detection of Coordination Requirements to support collaboration among software developers. Proceedings of the International Conference on Software Engineering, Jun. 2-9, IEEE Xplore Press, Zurich, Switzerland, pp: 1601-1603. DOI: 10.1109/ICSE.2012.6227230

Cataldo, M., P.A. Wagstrom, J.D. Herbsleb and K.M. Carley, 2006. Identification of coordination requirements: implications for the design of collaboration and awareness tools. Proceedings of the 20th Conference on Computer Supported Cooperative Work, Nov. 04-08, ACM, Banff, Alberta, Canada, pp: 353-362. DOI: 10.1145/1180875.1180929

Cataldo, M.B., M. Bass, J.D. Herbsleb and L. Bass, 2007. On coordination mechanisms in global software development. Proceedings of the International Conference on Global Software Engineering, Aug. 27-30, IEEE Xplore Press, pp: 71-80. DOI: 10.1109/ICGSE.2007.33

Cepêda, R.S.V., A.M. Magdaleno, L.G.P. Murta and C.M.L. Werner, 2010. Evoltrack: Improving design evolution awareness in software development. J. Brazil. Comput. Society, 16: 117-131. DOI: 10.1007/s13173-010-0011-5

Chaves, A.P., E.H.M. Huzita, V. Vieira and I. Steinmacher, 2010. A context conceptual model for a distributed software development environment. Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, (EKE' 10), pp: 437-442.

Chaves, A.P., I. Steinmacher, G.C.L. Leal, E.H.M. Huzita and A.B. Biasao, 2011. OntoDiSENv1: An ontology to support global software development. CLEI Electronic J., 14: 1-12.

Damian, D., 2002. Workshop on global software development. Proceedings of the 24th International Conference on Software Engineering, May 19-25, ACM, Orlando, Florida, pp: 667-668. DOI: 10.1145/581339.581435

de Lucia, A., F. Fasano, R. Francese and R. Oliveto, 2005. Traceability management in ADAMS. Proceedings of the 1st International Workshop on Distributed Software Development, (DSD' 05), pp: 135-149.

de Souza, C., P. Dourish, D. Redmiles, S. Quirk and E. Trainer, 2004. From technical dependencies to social dependencies. Proceedings of the Workshop on Social Networks for Design and Analysis: Using Network Information in CSCW, (UNI' 04).

Dourish, P. and V. Bellotti, 1992. Awareness and coordination in shared workspaces. Proceedings of the ACM Conference on Computer-Supported Cooperative Work, Nov. 01-04, ACM, Toronto, Ontario, Canada, pp: 107-114. DOI: 10.1145/143457.143468

Froehlich, J. and P. Dourish, 2004. Unifying artifacts and activities in a visual tool for distributed software development teams. Proceedings of the 26th International Conference on Software Engineering, May 28-28, IEEE Xplore Press, Edinburgh, UK, pp: 387-396. DOI: 10.1109/ICSE.2004.1317461

Fuks, H. and R.L. Assis, 2001. Facilitating perception on virtual learning ware based environments. J. Syst. Inform. Technol., 5: 93-113. DOI: 10.1108/13287260180000761

Gutwin, C., K. Schneider, D. Paquette and R. Penner, 2005. Supporting group awareness in distributed software development. Proceedings of the IFIP International Conference on Engineering Human Computer Interaction and Interactive Systems, (IIS' 05), Springer, Berlin, pp: 901-904. DOI: 10.1007/11431879_25

Gutwin, C., R. Penner and K. Schneider, 2004. Group awareness in distributed software development. Proceedings of the ACM Conference on Computer Supported Cooperative Work, Nov. 06-10, ACM, Chicago, Illinois, USA, pp: 72-81. DOI: 10.1145/1031607.1031621

Hargreaves, E. and D. Damian, 2004. Can global software teams learn from military teamwork models? Proceedings of 26th International Workshop on Global Software Development, May 24-24, Edinburgh, UK, pp: 21-23. DOI: 10.1049/ic:20040307

Herbsleb, J.D., A. Mockus, T.A. Finholt and R.E. Grinter, 2000. Distance, dependencies and delay in a global collaboration. Proceedings of the ACM Conference on Computer Supported Cooperative Work, Dec. 02-06, ACM, Philadelphia, Pennsylvania, USA, pp: 319-328. DOI: 10.1145/358916.359003.

Herbsleb, J.D. and D. Moitra, 2001. Guest editor's introduction: global software development. IEEE Software, 18: 16-20. DOI: 10.1109/52.914732

Ivcek, M. and T. Galinac, 2008. Aspects of quality assurance in global software development organization. Proceedings of the 27th International Conference on Telecommunications and Information of the 31th International Convention MIPRO, (CTI' 08), pp: 150-155.

Jiménez, M., A. Vizcaíno and M. Piattini, 2010. Improving distributed software development in small and medium enterprises. Open Software Eng. J., 4: 26-37. DOI: 10.2174/1874107X01004020026

Jiménez, M., M. Piattini and A. Vizcaíno, 2009. Challenges and improvements in distributed software development: A systematic review. Adv. Software Eng., 2009: 1-14. DOI: 10.1155/2009/710971

Lahtinen, S. and J. Peltonen, 2005. Adding speech recognition support to UML tools. J. Visual Lang. Comput., 16: 85-118. DOI: 10.1016/j.jvlc.2004.08.001

Lanubile, F., C. Ebert, R. Prikladnicki and A. Vizcaíno, 2010. Collaboration tools for global software engineering. IEEE Software, 27: 52-55. DOI: 10.1109/MS.2010.39

Layman, L., L. Williams, D. Damian and H. Bures, 2006. Essential communication practices for extreme programming in a global software development team. Inform. Software Technol., 48: 781-794. DOI: 10.1016/j.infsof.2006.01.004

Leal, G.C.L., A.P. Chaves, E.H.M. Huzita and M.E. Delamaro, 2012. An integrated approach of software development and test processes to distributed teams. J. Univ. Comput. Sci., 18: 2686-2686. DOI: 10.3217/jucs-018-19-2686

Monte-Alto, H.H.L.C., A.B. Biasão, L.O. Teixeira and E.H.M. Huzita, 2012. Multi-Agent Applications in a Context-Aware Global Software Development Environment. In: Distributed Computing and Artificial Intelligence: Advances in Intelligent and Soft Computing, Omatu, S., J. De Paz Santana, S. González, J. Molina and A. Bernardos *et al*. (Eds.), Springer, pp: 265-272.

Noll, J., S. Beecham and I. Richardson, 2010. Global software development and collaboration: Barriers and solutions. ACM Inroads, 1: 66-78. DOI: 10.1145/1835428.1835445

Omoronyia, I., J. Ferguson, M. Roper and M. Wood, 2010. A review of awareness in distributed collaborative software engineering. Software: Practice Exp., 40: 1107-1133. DOI: 10.1002/spe.1005

Prikladnicki, R., S. Marczak, T. Conte, C. Souza and J.L.N. Audy *et al*., 2011. The evolution and impact of the research in distributed software development in Brazil. Proceedings of the 25th Brazilian Symposium on Software Engineering, Sept. 28-30, IEEE Xplore Press, Sao Paulo, Brazil, pp: 126-131. DOI: 10.1007/s13173-013-0114-x

Salman, I., A.T. Misirli and N. Juristo, 2015. Are students representatives of professionals in software engineering experiments? Proceedings of the 37th IEEE International Conference on Software Engineering, May 16-24, IEEE Xplore Press, Florence, Italy, pp: 666-676. DOI: 10.1109/ICSE.2015.82

Sangwan, R., M. Bass, N. Mullick, D.J. Paulish and J. Kazmeier, 2006. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). 1st Edn., Auerbach Publications, Boca Raton, ISBN-10: 0849393841, pp: 288.

Sarma, A., Z. Noroozi and A. van der Hoek, 2003. Palantir: Raising awareness among configuration management workspaces. Proceedings of the 25th International Conference on Software Engineering, May 3-10, IEEE Xplore Press, Portland, OR, USA, pp: 444-454. DOI: 10.1109/ICSE.2003.1201222

Sengupta, B., S. Chandra and V. Sinha, 2006. A research agenda for distributed software development. Proceedings of the 28th International Conference on Software Engineering, May 20-28, ACM, Shanghai, China, pp: 731-740. DOI: 10.1145/1134285.1134402

Silva, F.Q.B., C. Costa, A.C.C. Franca and R. Prikladinicki, 2010. Challenges and solutions in distributed software development project management: A systematic literature review. Proceedings of the International Conference on Global Software Engineering, Aug. 23-26, IEEE Xplorev Press, Princeton, NJ, USA, pp: 87-96. DOI: 10.1109/ICGSE.2010.18

Steinmacher, I., A.P. Chaves and M.A. Gerosa, 2012. Awareness support in distributed software development: A systematic review and mapping of the literature. Comput. Supported Cooperative Work, 22: 113-158. DOI: 10.1007/s10606-012-9164-4

Vieira, V., 2008. CEManTIKA: A domain-independent framework for designing context-sensitive system. Tese de Doutoramento, Centro de Informática - CIn, Universidade Federal de Pernambuco-UFPE, Recife - Pernambuco.

Vivian, R.L., E.H.M. Huzita and G.C.L. Leal, 2013. Supporting distributed software development through context awareness on software artifacts: The DiSEN-CollaborAR approach. Proceedings of the 28th Annual ACM Symposium on Applied Computing, Mar. 18-22, ACM, Coimbra, Portugal, pp: 765-770. DOI: 10.1145/2480362.2480509

Vivian, R.L., E.H.M. Huzita, G.C.L. Leal and A.P. Chaves, 2011. Context-awareness on software artifacts in distributed software development: A systematic review. Proceedings of the 17th CRIWG Conference on Collaboration and Technology, Oct. 02-07, Springer, Paraty, Brazil, pp: 30-44. DOI: 10.1007/978-3-642-23801-7_3

Whitehead, J., 2007. Collaboration in software engineering: A roadmap. Proceedings of the Future of Software Engineering, May 23-25, IEEE Xplore Press, Minneapolis, MN, USA, pp: 214-225. DOI: 10.1109/FOSE.2007.4

Wohlin, C., P. Runeson, M. Host, M. Ohlsson and B. Regnell *et al*., 2000. Experimentation in Software Engineering: An Introduction. 1st Edn., Kluwer Academic Publishers, USA, ISBN: 0792386825, pp: 204.

Yacoub, M.K., M.A.A. Mostafa and A.B. Faria, 2016. A new approach for distributed software engineering teams based on kanban method for reducing dependency. J. Software, 11: 1231-1241. DOI: 10.17706/jsw.11.12.1231-1241

Zhang, Y., R. Witte, J. Rilling and V. Haarslev, 2008. Ontological approach for the semantic recovery of traceability links between software artefacts. IET Software, 2: 185-203. DOI: 10.1049/iet-sen:20070062