Journal of Science and Technology
ISSN: 2456-5660 Volume 6, Issue 05 (September – October 2021)
www.jst.org.in *DOI:https://doi.org/10.46243/jst.2021.v6.i05.pp168 - 179*

Tools used in Global Software Engineering: A systematic mapping review

D Rajeev Naik, S. Leelashyam, M Venkataratnam
Assistant Professor[1], Associate Professor[2,3]
Department of ECE,
drajeevnaik.ece@anurag.ac.in, vleelashyam.ece@anurag.ac.in, vvaralaxmi.ece@anurag.ac.in
Anurag Engineering College, Kodada, Telangana

To Cite this Article
D Rajeev Naik, S. Leelashyam, M Venkataratnam, **Tools used in Global Software Engineering: A systematic mapping review**
" *Journal of Science and Technology, Vol. 06, Issue 05 September- October 2021, pp168-179*
Article Info
Received: 23-08-2021 *Revised: 8-09-2021* *Accepted: 16-10-2021* *Published: 28-10-2021*

ABSTRACT
Method: We performed a systematic mapping review through a search for studies that answered our research question, ''Which software tools (commercial, free or research based) are available to support Global Software Engineering?'' Applying a range of related search terms to key electronic databases, selected journals, and conferences and workshops enabled us to extract relevant papers. We then used a data extraction template to classify, extract and record important information about the GSD tools from each paper. This information was synthesized and presented as a general map of types of GSD tools, the tool's main features and how each tool was validated in practice.
Results: The main result is a list of 132 tools, which, according to the literature, have been, or are intended to be, used in global software projects. The classification of these tools includes lists of features for com- munication, coordination and control as well as how the tool has been validated in practice. We found that out the total of 132, the majority of tools were developed at research centers, and only a small per- centage of tools (18.9%) are reported as having been tested outside the initial context in which they were developed.

Keywords:
Global Software Development Distributed Software Engineering Tool
Systematic Mapping Study*Context*: This systematic mapping review is set in a Global Software Engineering (GSE) context, charac- terized by a highly distributed environment in which project team members work separately in different countries. This geographic separation creates specific challenges associated with global communication, coordination and control.
Objective: The main goal of this study is to discover all the available communication and coordination tools that can support highly distributed teams, how these tools have been applied in GSE, and then to describe and classify the tools to allow both practitioners and researchers involved in GSE to make use of the available

Introduction

Global Software Engineering (GSE) has become a growing area of research, apart from being an expanding trend in the Informa- tion Technology (IT) industry [3]. GSE requires software tools (management tools, development tools, etc.) to support the special characteristics that this environment has, and which have princi- pally come about as a result of the distance factor (temporal, geo- graphic and socio-cultural distance) [4].
Modern software development, such as globally dispersed teams, creates specific challenges and risks (in spite of the benefits that can be obtained) for the software industry, which need to be considered [5]. In fact, developing software systems through col- laboration with other partners and in different geographical loca- tions is a great challenge for organizations [6,7].
Software tools for GSE should therefore help to alleviate prob- lems such as: (a) *Geographic Dispersion*, which sometimes causes a loss of synchronous communication or team interactions, since the sites are in different time zones; (b) *Control and Coordination Breakdown*, owing to the difficulties created by a distributed envi- ronment; (c) *Loss of Communication*; this is the case in this type of environment, if we consider that the richest communication med- ium is face-to-face communication; (d) *Loss of Team Spirit* and trust among team members [8] and (e) *Cultural Differences* which occur when people from different cultures work together in a global environment [9].
Tools designed to alleviate the challenges stated above should therefore include special features, such as supporting the interac- tion of distributed teams by applying communication and collaboration technology [10], supporting the development of real-world projects [11], minimizing the cost of the tools and infrastructure needed, along with their maintenance effort [12] or helping to create a feeling of trust between the members [13,14], and facilitating the knowledge of team ethics [15], among others. However, there is insufficient information regarding which tools are able to assist in the aforementioned challenges, or about which particular tools offer features that are suitable to allow them to be used in a GSE environment. The most that we can affirm is that certain surveys exist in which some of the existing tools, usually those regarding collaboration, are briefly presented. A good example of this is [16], in which the authors present a set of collaboration tools for GSE, classified by the areas in which they can be used.
In this respect, tools from the area of Distributed Software Engi- neering (DSE) often include interesting features that may be useful in a global environment. Moreover, according to the study pre- sented in [17], collaborative software tools for distributed develop- ment constitute one of the research areas in which important research questions need to be addressed. For example, selecting appropriate tools that correspond to the characteristics of glob- ally-distributed projects is not an easy task [18] since, among other reasons, there is not enough information about the tools that are available to support GSE teams.
The goal of this work is, therefore, to carry out a systematic map- ping review of GSE tools, in order to obtain information about which tools are available for GSE and what features they include. This review was performed by following the process described in [1] which explains how effective such studies have been when used for software engineering topics. Other systematic mapping review papers in this field, such as that shown in [2], were also studied. The main goal of our review is to compile the most complete list of GSE tools possible and to present these results as a visual sum- mary (map) of classified features. The classification of the tools will be based on an extraction of common features across studies (prin- cipally related to communication, owing to its importance).
Companies, practitioners and researchers may find this work use- ful, since it provides a wide description of tools. That being the case, companies and

practitioners can use this paper before buying a tool, and employ it as preliminary information about what tools exist and what their features are. Researchers can also consult the paper to discover the state of the technology in the field of GSE, and to observe how tools can be grouped according to the process they sup- port. The fact that this paper is the first systematic study of GSE tools makes it a significant contribution to the GSE community.
This work is structured as follows. In Section 2 we outline the methodology used to perform the systematic mapping review, including question formulation, the selection of sources and stud- ies, information extraction and the mapping process. In Section 3, the results obtained after performing the systematic mapping review are presented. In Section 4 we discuss the threats to the validity of the results. Finally, in Section 5, we outline the conclu- sions obtained.

Systematic mapping review of tools to support GSE

The systematic mapping review which follows has been devel- oped by using the guidelines presented in [1] and taking into ac- count the information obtained after studying other systematic mapping reviews, for example, [19–22]. In this section, we provide an overview of the steps involved in the process.

Definition of research question

The tools which are available for use in GSE were obtained by formulating the following question:
Which software tools (commercial, free or research based) are available to support Global Software Engineering?
In this work, ''research tools'' are considered to be those devel- oped by researchers in research labs or groups, which are not developed for profit or commercially available. These tools are not usually available for download and must be obtained on re- quest from the researchers who developed them.
''Free'' research tools are those that are open source, where there is no obligation to pay for the license. ''Commercial'' tools on the other hand require a payment for the license, though they may offer a free trial period.
The list of keywords used to discover and answer the research question consisted of:*tool, software, global, engineering, development* and *distributed*.
The research question selected was expected to provide the fol- lowing results once the systematic mapping review had been com- pleted. Our intention was that the information would tell us:

Which software tools are used/available in the context of GSE. In this case, the area of Distributed Software Engineering (DSE) was also considered within the context of GSE.
The main features that GSE tools often include.
The classification of the software tools that are available, depending on their features or/and the areas in which they are used.

The population observed was the set of software tools, com- posed of both those that have been presented as specific tools to support GSE, and those which were not specifically designed to support GSE, but which contain features that are useful in GSE. The set of tools was compiled through a study of the assortment of published work shown in our list of sources and presented in the following section.

Conducting the search

The list of keywords shown above was combined by using the logical connectors ''OR'' and ''AND'', to obtain the main search string (see Table 1). The search string thus had the structure P1 AND P2 AND P3, each part of which is defined as follows:

P1: tool OR technology. P2: global OR distributed.
P3: software development OR software engineering.

The terms used in the search string are commonly employed in GSE/DSE research, and this implied encountering a large amount of non-useful papers, but the idea of this review was to obtain the
maximum number of tools. We should state, however, that in line with the research question, inclusion and exclusion criteria that were designed to obtain solely useful work were used. With P1, we aimed to obtain all those pieces of work in which anything re- lated to tools was included. With P2 and P3, we wished to find work related to GSE and DSE. It was deemed that DSE would be considered to be GSE if the level of distribution was sufficiently high.
The list of the sources selected, and in which the search strings were executed to carry out the systematic review, is:

Science@Direct, on the subject of Computer Science.
Wiley InterScience, on the subject of Computer Science.
IEEE Digital Library from the Computer Society.
ACM Digital Library.

This list of sources was selected from the recommendations made by experts in the area of this systematic review. These sources include certain highly important journals, in which our re- search area is widely dealt with, such as: Information and Software Technology, IEEE Software, Computer, Information and Manage- ment, and Systems and Software. Moreover, from IEEE we included the proceedings of the most relevant conference on GSE (*Interna- tional Conference on Global Software Engineering* – ICGSE).
The search string presented above was adapted to each search engine of the respective sources (see Table 2), owing to the special features or restrictions that each search engine had. For instance, the *ACM Digital Library* allows other publishers' papers to be searched for, but in this case we only wished to use the ACM search engine to access those papers published by ACM itself, thereby avoiding duplicated work. This feature was obtained (as is shown in Table 2) by ensuring that the publisher in the search string was ACM.
This table has been included to permit possible future reviews of the results obtained. By selecting the options indicated in each search engine, the same results should therefore be obtained.
Another restriction included in all the search engines was that of selecting only those pieces of work published from the year 2000 onwards, since, regardless of the particular research area, tools become obsolete after a few years, owing to the rapid evolu- tion of technology. We therefore believe that any tools mentioned before the year 2000 can now be considered as obsolete. Moreover, taking GSE as an effect of globalization and as a 21st century trend [23], only studies performed after 2000 have been considered to be important in this work.

During the process of the study selection we assumed that the quality of the papers obtained would be ensured by the evaluation process followed in deciding what papers to publish. However, this selection of studies is not based on quality but rather on relevance. When choosing, we aimed to gather relevant papers which were in accordance with the focus of the research question and the review goal. As has already been mentioned, the most relevant papers were obtained by performing four stages or phases in each source. The first stage (Based on Search) consisted of recovering an ini- tial set of papers by using the sources' search engines and the strings presented above in Table 2. In this first phase, we obtained a high number of results in some cases (ACM or IEEE), but in many other cases these results were not useful, since they consisted of
comments, letters or repeated work.

The second stage (Exclusion upon Title) thus consisted of elim- inating both non-useful results and repeated work, by considering only the title (and, in some cases, the author(s)) of each result. In this case, non-useful results were those that were not journal arti- cles, workshop papers and conference papers. A result was consid- ered to be repeated if there was another paper with the same title and the same authors.

Once we had eliminated the non-useful results, we began a revision phase by studying abstracts (Exclusion upon Abstract). This phase consisted of examining the paper's abstract, in order to ascertain whether the subject of the paper was related to tool support for GSE/DSE or whether the work focused on presenting a specific tool. This phase was carried out in two steps, the first of which consisted of a rapid review of paper abstracts. However, we realized that, in some cases, merely reviewing the paper's ab- stract was not sufficient, and that it would also be necessary to re- view the work's conclusions.

Finally, in order to obtain the definitive list of primary studies, a complete review of the texts of the papers (Exclusion upon Full Text) remaining from the previous phase was carried out. In this case, all the papers were related to tool support, or presented a spe- cific tool. However, we encountered two problems. The first was that we had papers presenting tools for DSE but we did not know whether the tools mentioned would be useful for GSE. In these
cases, we reviewed the texts in full to discover whether the tools had been designed for co-located teams (low level of distribution) or for a higher level of distribution. In this respect, we decided that Web-based tools can be used in a globally distributed environment because they are accessible from any Web browser. The second problem in this phase was that some papers were related to tool support for GSE/DSE, but they discussed types of tools (communica- tion tools, design tools, etc.) or tool features (chat, e-mail, etc.) without referring to a specific tool. All these works were eliminated. Once these two problems had been solved by reviewing the texts of the remaining papers in full, we obtained the list of 66 pri-
mary studies presented in Appendix A.

In just a few cases, different papers mentioned or presented the same tool. We therefore checked all the papers regarding the same tool and rejected those that simply mentioned the tool and did not provide any useful information about it. Moreover, in those cases in which a tool was presented in several papers, we selected the most up-to-date paper as the primary study associated with each tool. In this study, each tool has therefore been related to just one paper (primary study). It is also important to note that one pa- per may include several tools.

What is more, in order to avoid missing any important informa- tion about a tool, we checked the information provided in the other papers that had been rejected. In all cases, the information pro- vided by both papers was similar, and the updated paper usually contained more information since, for instance, a new version of the tool had been implemented and or tested.

Data/information extraction and mapping of studies

Once the primary studies had been chosen, the relevant infor- mation for the systematic review was obtained. The inclusion cri- terion for the information originating from the primary studies consisted of the names of specific software tools, the area in which they are used and the features that make them usable in a GSE environment. The information from the primary publications was stored in a table similar to Table 4, in which the data extraction for- mat was structured in two parts: The first part was used to obtain information about the study and the second was used to obtain information about the tools found in the study.

that the topic of GSE in general, and GSE tools in particular, is one in which there is an increasing amount of interest, mainly from 2006 onwards. This increase is not uniform in all the areas studied. Areas such as Knowledge Man- agement Tools (KMTs), Virtual Meeting Tools (VMTs) and Software Engineering Management Tools (SEMTs) have been continuously tackled in publications over the last few years, while other areas, such as those of Software Quality Tools (SQTs) or Software Engi- neering Process Tools (SEPTs), are only dealt with in a few publica- tions. What is more, the area of Socio-Cultural Tools (S-CTs), while not dealt with much in early years, has recently been gaining importance. This change has come about because of the problems that exist in GSD, which have arisen as a result of the cultural and social differences between globally distributed teams.

In terms of tools (see left-hand side of Fig. 1), the biggest group
is that of Research Tools (58 tools), since the papers studied are, on the whole, research works, while the smallest group is that of Com- mercial Tools (30 tools). However, some free, commercial or re- search tools exist for almost all the processes. Companies can thus decide whether they prefer a commercial tool, a free tool, or whether they would rather obtain a research tool. It is important to note that the 2010 data included in Fig. 1 appertains to the first quarter of 2010.

Validation of tool classification scheme

In order to ensure that the classification of tools and papers was reliable, we decided to ask three different researchers to carry out three different classifications and then check the level of agree- ment among them in the classification. The level of agreement was checked by applying an inter-rater reliability analysis using the Kappa statistic, which determines the consistency among rat- ers. To be more precise, we used the Fleiss Kappa statistic, which can be applied for 2 or more raters (in our case 3 raters). By using the data presented in Table 5, we obtained a Kappa value of 0.952, which means an *almost perfect agreement* in the tools' classifica- tion. Moreover, we obtained a Kappa value of 0.966, also meaning an *almost perfect* agreement in the papers' classification.

We can therefore have confidence that the classification of the papers and tools is fairly reliable because, according to the kappa test, the agreement regarding the classification is almost perfect among the three researchers. Results and discussion

In this section we present the results obtained from the system- atic review of GSE tools. Having established that there was a need for such a review and that no other review had been published in this area, we proceeded to conduct the mapping study.

Fig. 2 shows that 44% of the 66 primary studies present a single tool for a specific area (a), 46% present a set of tools for a single area
(b) and 10% present a set of tools for different areas (c). These re- sults indicate that only a minority of papers deal with a set of tools covering the complete software lifecycle; only 10% present a set of tools for different areas that relate directly to software lifecycle processes as defined in the SWEBOK [25] and briefly explained in Section 3.1.

One of our goals after performing the systematic mapping re- view was to identify which software tools are used/available in the context of GSE, according to the literature studied.

Once we had studied each tool, we realized that the features in- cluded in them could be categorized. These feature categories are part of the expected results and are presented in the following subsection.

Classification of features

With regard to the second expected result, to identify the main features that GSE tools often include, we identified seven feature groups. The main features of the tools studied are summarized in Table 14, but we shall first describe each category as follows.

The first category is *Subject*. In this case, we have attempted to classify each tool into a related subject. For example, if a tool is de- scribed as a UML modeler, it will be classified in the design subject. This classification was carried out by considering the use of differ- ent classification frames. One of these was that presented in [26], in which the author proposes a well-structured classification framework for CASE tools. However, as we needed a process- oriented classification to check the processes covered by the tools

[26], the classification was more oriented towards the type of tool (IDE, Framework, etc.). That being so, we eventually came to the conclusion that it was better to use the Areas of the Software Engineering Body of Knowledge (SWEBOK) [25], which is more ori- ented towards software engineering processes. We have therefore specifically used the subjects defined in SWEBOK for the Software Engineering Tools and Methods knowledge area. Moreover, we have extended the subjects included in SWEBOK with other sub- jects not included in software engineering. For example, the sub- jects of Knowledge Management Tools and Virtual Meeting Tools have been added. The latter has been included because virtual meetings are an important means of communication in GSE. We have also included the subject of Socio-Cultural Tools, since in GSE, socio-cultural aspects influence project performance and knowledge management [27]. The values used to classify a tool into a knowledge area are shown in Table 6.

The type of tool that can be included in some of these knowl- edge areas is often sufficiently clear, but in some cases, such as the *Miscellaneous* subject, it is not clear which kind of tool can be included. In order to understand what kind of tool can be included in certain knowledge areas, we considered the following points [25]:

The SRT subject includes *Requirements Modeling Tools* (for eliciting, analyzing, specifying, and validating requirements) and *Requirement Traceability Tools*.
The SDT subject includes tools for creating and checking software designs.
The SCT subject includes Program Editors, Compilers and Code Generators, Interpreters and Debuggers.
The STT subject includes Test Generators, Test Execution Frameworks, Test Evaluation Tools, Test Management Tools and Performance Analysis Tools.
The SMT subject includes *Comprehension Tools* (for instance, visualization tools such as animators and program slicers) and *Reengineering Tools*.
The SCMT subject includes Defect, Enhancement, Issue, and Problem-Tracking Tools, Version Management Tools and Release and Build Tools.
The SEMT subject includes Project Planning and Tracking, Risk Management and Measurement Tools.
The SEPT subject includes Process Modeling Tools, Process Management Tools, Integrated CASE environments and Pro- cess-centered Software Engineering Environments.
The SQT subject includes *Review and Audit* and *Static Analysis* Tools. It also includes *Inspection Tools*, which in this case are considered to be special kinds of review and document management tools that help to increment the quality of product documentation.
The MTI subject principally includes *Meta-tools* or *Integra- tion Tools*, that is, tools that integrate several tools in order to construct a more complex one.
The KMT subject (not included in SWEBOK), includes tools that support the knowledge lifecycle processes, such as the creation or distribution of knowledge (for instance a WIKI tool).
The VMT subject (not included in SWEBOK), includes tools which principally permit communication among distributed teams. Examples of these are videoconference tools, virtual room tools, etc.
The S-CT subject (not included in SWEBOK), includes tools related to offering support to socio-cultural aspects through, for instance, social networks, and an analysis of them.

As regards the subjects presented, Table 7 shows the areas on which the primary studies are focused, that is, which particular areas are supported by the tools presented in each primary study. In some cases (the most complete pieces of work) the studies are related to several areas, because they present different types of tools. As Table 7 shows, tools to support software construction, software engineering management (project management, issue tracking, etc.), knowledge management and virtual meetings are those most frequently mentioned in the selected studies.

On the other hand, subjects such as Software Maintenance Tools (SMTs), Software Engineering Process Tools (SEPTs), Miscellaneous Tool Issues (MTIs) and Socio-Cultural Tools (S-CTs) are not well supported or researched.

The second category, *License*, is used to define which kind of li- cense is associated with each tool. We have thus defined three types of categories in relation to their licenses. These types are de- scribed in Table 8.

Upon observing Fig. 3, we can see that 43.6% of the tools studied are research tools (the highest group). The explanation for this is probably that the works reviewed are mainly from research. These tools are seldom used; they are, however, quite useful for allowing companies to learn which features are used in GSE research tools, in order to include them in commercial tools. Moreover, one goal of this review was to obtain the maximum number of tools and to at- tain a list of them that was as comprehensive as possible.

33.5% of the tools studied are free tools (Fig. 3). We consider that this group of tools may be especially useful in the research do- main because researchers can experiment without having to pay for a license. Finally, 22.9% of the tools studied are commercial tools.

Moreover, as is shown in Fig. 4, the group of free tools provides better support in areas related to coding, such as those of construc- tion (SCT), testing (STT) or configuration management (SCMT) (see Table 6 for subject abbreviations). On the other hand the research tools listed in Fig. 5 offer support in areas such as software design or software quality, for which there are no free tools.

With regard to commercial tools, Fig. 6 shows that there is a lack of tools in the areas of *Software Engineering Process Tools* (SEPTs), *Software Quality Tools* (SQTs), *Software Testing Tools* (STTs) and *Software Construction Tools* (SCTs). That being so, there is an opportunity for organizations to develop tools in these areas.

The third category is *Communication*. This category includes features that allow a team member to communicate with other distributed team members. We therefore differentiated between synchronous communication features, such as chat, videoconfer- ence or VoIP, and asynchronous communication features, such as e-mail. Table 9 shows the different communication types considered.

With regard to the type of communication used in each tool, and taking into account those which included any communication channel (54%), the majority of the tools studied enable asynchro-

In this category we also consider another type of awareness, de- fined as *Change Awareness*. This type of awareness considers those features related to letting users know ‘‘who is doing what’’, inde- pendently of whether team members are working synchronously or asynchronously. We have identified two main features to sup- port this (Table 10 describes the features related to awareness):

Visual features: These include features that help users to be aware of the actions performed by other users, mainly in a synchronous context, for instance, *color identification*.
E-Mail Notifications: This feature is that which is most widely-used in an asynchronous context to make users
aware of the actions performed by other users.

The fifth category is that of *Control and Coordination*. This cat- egory includes features that principally assist managers with con- trol and coordination issues. For instance, in geographically dispersed teams it is important, when controlling the progress of tasks or activities, to track different issues such as bugs or tasks, and to do so through the use of software tools. In this particular case, these tools are called Issue Tracking Systems. This kind of sys- tems also assists in aspects of coordination, since managers have an overview of the project's progress and are able to make coordi- nation adjustments. In addition, these kinds of systems are com- plemented with *Version Control Systems* or *Build Management Systems*, which offer more control and coordination information. Note that these tools can be considered as independent tools or as features when they are integrated into other more complex tools. In order to classify these control and coordination aspects, we have detected three main types of tools, which are presented in the following table (Table 11).
The sixth category is that of *Knowledge Management*. Here we
indicate whether the tool supports knowledge acquisition, knowl- edge sharing, knowledge distribution, etc. After reviewing the tools found, we have observed that these features are mainly supported by Wikis, Document Management Systems and Blogs. The main advantage of using these kinds of tools is that most of them are Web-based, which allows knowledge to be managed in distributed environments. Three kinds of features are therefore considered, and these are summarized in the following table (Table 12).
The seventh and last category is the *Socio-Cultural* category. In this category, the features included are intended to help users to reduce socio-cultural distance. Examples of this are social net- works. These social networks may include information that can be social, cultural or concerns language, and so on. Team members can consult this information to obtain a better awareness of, for example, other team members' cultural customs. We have, in gen-

eral, identified three kinds of features/tools included in the tools found, which are summarized in the following table (Table 13).
84.8% of the tools studied do not include features that support socio-cultural aspects (see Fig. 8). Moreover, of the total number of tools that include socio-cultural features (15.2%), only some of them include a simple profile manager. The most comprehensive tools in this aspect include social network support.

Tool classification and description

With regard to the third goal of this literature review, defined as classifying the available software tools into groups depending on their features and/or the areas in which they are used, Table 14 shows a description of the tools studied, indicating which features each tool includes.
In order to complete the table, we have assumed that a tool in- cludes a feature if it is part of the tool or if it can be included in the tool through the integration of another simpler tool. A typical case of this is when tools have an Issue Tracking System (ITS) through the integration of an Issue Tracking Tool such as Jira. However, this does not mean that all the possibilities of integration have been checked. Only the most obvious and easy-to-find integration possi- bilities have been considered.
As was previously mentioned, the information regarding the
132 tools included in the table has been extracted from the primary studies (which are identified in the table in the PS column) and the tools' websites. This implies that the information provided in the table is limited in this respect. However, more features could probably be discovered by actually installing the tools.
If the table above is to be understood correctly, then the infor- mation shown must be read as follows. Imagine that you need to use a set of tools in your company or research lab to, for instance, support distributed communication (VMT). Bearing this require-

ment in mind, the table can be used to select those tools related to VMT at a glance.
If you know which tools are available for this process, you can compare the most comprehensive ones by observing which fea- tures are included in each one. For example, for VMT, you could se- lect *Webex*, *TeamSpace* and *Yahoo Messenger* as possible options, since they support the majority of the features presented in the ta- ble (i.e., they support audio, video and chat communication – val- ues A, V, C in the table in the communication feature). Moreover, the tool shows when users start a new session, and/or it also dis- plays which users are working on the same session (value S in the presence awareness feature), in addition to the awareness information when there are changes in the tool or when the ses- sion is presented visually (value V in the change awareness feature).
At this point, the features offered by these tools are similar,
depending on the particular needs or the company or lab policy. However, the license can also be taken into account. If a research lab wishes to experiment, it may be advisable to select a research tool (TeamSpace in this case) or a free tool (Yahoo Messenger). One critical process in the context of a distributed development is the Project Management Process. By using this table, a company can select the most desirable tools to support this process (SEMT in Table 14) and reduce the problems related to distribution. Thus, by following the process explained in the previous example, it is possible to select the most comprehensive tools presented in the table, taking into account the type of license. In the case of Project Management, it is important to use tools which include features supporting control and coordination. With that in mind, a person can select from the table those tools related to Project Manage- ment (SEMT) which include the maximum number of coordination and control features. This person could therefore select *ActiveCol- lab*, *Assembla* or *Rational Team Concert*, because they appear to be the most comprehensive tools that include communication fea- tures, such as Chat (C) or Forums (F). Moreover, they offer Version Control Systems (VCSs), Build Management Systems (BMSs) and Is- sue Tracking Systems (ITSs) as control and coordination mecha- nisms and they also support knowledge management with the use of wikis (W) and/or

document management features (DM).

In GSD, socio-cultural aspects influence project performance. In order to help reduce socio-cultural problems, Table 14 includes information about which socio-cultural features are used in each tool, along with information concerning which tools exist that are directly related to socio-cultural aspects. As is shown in the table, the most common features used are the inclusion of user profiles (P) to provide details concerning personal information, together with the incorporation of a social network (SN) in the tool. Moreover, the majority of those tools directly related to socio-cul- tural aspects focus mainly on studying social dependencies in social networks (SDNs), such as *Tesseract, CASOS or Ariadne*.

Tools and features used in each knowledge area

With regard to the different knowledge areas in which the tools have been classified, Table 14 can also be used to extract which features are most frequently provided by the studied tools in each area. The following paragraphs describe which features are com- monly used by the studied tools in each knowledge area. Moreover, the lists of tools that can be used in each area are also provided.

Requirement Tools (SRTs). In this case, the most frequently provided feature is the Issue Tracking System, which makes users aware of important issues or changes. This feature is usually com- plemented with visual awareness techniques (value V in the Changes Awareness column), such as highlighting important as- pects in different colors and the possibility of inserting comments (Cmt in the table). Moreover, the most complete tools, such as Ra- tional Requisite Pro, also include a document manager (DM) with which to attach important requirement documents. The list of Requirement Tools is therefore the following: ARENA, DOORS, EGRET, eRequirements, GatherSpace, Rational Requisite Pro and Rational Requirement Composer.

Design Tools (SDTs). The feature most frequently provided in design tools is Awareness. Bearing in mind that most of these tools have been designed to be used synchronously (but distrib- uted in this case), the majority of the tools use both visual aware- ness (V) and session awareness (S in the table). By combining these types of awareness, the tools are able to show each user who is editing what, or who is working on the session. Moreover, these kinds of tools usually allow comments to be written by the users. The most comprehensive tools, such as Together, also include an Issue Tracking System and a Version Control System. The list of De- sign Tools is: Artisan Studio, CAB, CAMEL, CoDesign, Creately, Glif- fy, GroupUML, Libra-on-Chat, Rational Software Modeler, STEVE, Sysiphus and Together.

Construction Tools (SCTs). In the case of the construction tools, something similar to the Design Tools occurs, the most fre- quently provided feature being the awareness feature (session and changes awareness). However, the construction tools also usu- ally include an Issue Tracking System (ITS) and a Version Control System (VCS), with the latter being used more frequently. The most complete tools, i.e., CollabVS, also include different channels of communication, such as audio and video communication. An important aspect with regard to construction tools is that some of them (GitHub, Google Code, SCI and Share) include a feature that is not usually included in any other tool shown in the table. This feature is the Social Network (SN in the table), which allows users to include information about which programming languages or development environment they specialize in, the projects on which they have worked, and contact information. The list of construction tools is: byteMyCode, CheckStyle, CollabVS, Copper, GForge, Git- Hub, Google Code, ICI, Moomba, SCI, Share, Syde, TagSEA and TUKAN.

Testing Tools (STTs). This group of tools does not include special features. The main feature is to allow the remote execution of tests. Some of these also include an Issue Tracking System such as OpenSTA, a Version Control System such as TestLink and Build Management System such as SoftFab. The list of testing tools is, therefore: HttpUnit, JWebUnit, OpenSTA, Selenium, SoftFab, Test- Link, Watir and WebTest.

Maintenance Tools (SMTs). No maintenance tools were found by our study.

Configuration Management Tools (SCMTs). The main features of this group of tools are the Version Control System and the Issue Tracking System. They also include visual awareness mechanisms to inform of changes. One important innovation is that presented in WikiDev 2.0 which implements this kind of systems as a Wiki, is accessible from any Web browser and is very useful in a highly-distributed environment, owing to this very availability. The list of configuration management tools is: CASI, Darcs, Git, Mercurial, MUDABlue, Palantir, Perforce, Rational Clearcase, SCAR- AB, Subversion, TortoiseSVN and WikiDev.

Engineering Management Tools (SEMTs). Although this group of tools includes *Project Planning and Tracking, Risk Management* and *Measurement* Tools, the majority of them are related to *Project Planning and Tracking*, and the features mentioned here are there- fore mainly related to this type of tools. The main feature that this kind of tools should include is Awareness. The majority of them of- fer features such as email notifications in order to inform users about what is happening in the project. Moreover, Issue Tracking and Version Control Systems are commonly used for Project Track- ing. Other features that make the tools more complete and which are included in tools such as ActiveCollab, Assembla, Milos ASE or Rational Team Concert, are document managers, chat tools, for- ums or wikis. The list of Engineering Management Tools is thus: ActiveCollab, ADAMS, Assembla, Augur, Bugzilla, CodeBeamer, Cruise Control, DrProject, ‘‘Fonseca Tool’’, IssuePlayer, Jira, Mantis, MasePlanner, Maven, MILOS ASE, Rational Team Concert, TAMRI, Trac, WorkSpace Activity, WorldView and XPlanner.

Engineering Process Tools (SEPTs). This group of tools makes use of similar features to those in the design tools, since some of the Engineering Process Tools are modeling tools that use the same features. These features are usually awareness features (visual and session awareness) and also chat features for writing comments. The list of tools is: GENESIS, Hobbes, SPEARMINT and XCHIPS.

Quality Tools (SQTs). As the number of tools found in this to- pic is small, it is risky to draw conclusions. However, we can state that the main features are the document manager and the possibil- ity of writing comments. Some of the awareness features such as email notifications are also used. The list of SQT is: AISA, HP Quality Center, HyperCode, IBIS, WiT, WiP and XATI.

Miscellaneous Tool Issues (MTIs). This group includes *Meta- tools* or *Integration Tools* but only a couple of them are on the list. These two tools have the common feature that they are able to integrate different ITS, VCS and BMS features to be used as a single tool. However, the integration possibilities are very limited. The specific tools are MerlinToolChain and RepoGuard.

Knowledge Management Tools (KMTs). The importance of documents in which knowledge can be written and shared signifies that the main feature included in most of the tools in this group is the document manager. This feature is also complemented by a Version Control System to control the document versions and keep the users updated. Moreover, owing to the extended use of Web applications, other very commonly used features are

the Wikis, Forums and Blogs. The list of knowledge management tools is: 4everedit, ADDSS, BSCW, CAWS, CollabDev, DOCTOR, GalaxiWiki, Google Docs, Google Groups, iBistro, Knowfact, Knowledge Tree,
LiveNet, Lotus Quickr, MS Sharepoint, MoinMoin, MULTIMIND, PAKME, Saperion, ECM, TWiki and Xerox Docushare.

Virtual Meeting Tools (VMTs). The majority of these tools enable virtual meetings through the use of a chat tool. Moreover, they also usually include video and audio chat combined with awareness features, in order to know who is connected or to ascer- tain the state of each user (available, not available, disconnected, etc.). The most comprehensive tools also include a document man- ager to allow documents to be shared and edited in a virtual meet- ing. The list of VMT is: Connect Now, Consensus@nywhere, CVE, eConference, Google Wave, Lotus Sametime, MS Office Communi- cator, Miramar 2.0, MPK 2.0, MSN Messenger, P2P Conference, pcA- nywhere, TeamSpace, WebEx, WorkSpace 3D, Yahoo Messenger.

Socio-Cultural Tools (S-CTs). These tools are directly related to social networks and their analysis. Two main features are there- fore used in this kind of tools. The first is the use of social network analysis to obtain social dependency networks (SDN in the table). The use of these networks makes it possible to analyze how inter- actions occur in a work team. This could, for example, help in the making of decisions related to the group structure. The second fea- ture is the management of a social network (SN in the table). This feature is included in well-known tools such as Twitter and is com- monly used to keep people close to each other, by sharing personal information. The list of S-CT is: Ariadne, CASOS, Friendfeed, Tesser- act and Twitter.

Common groups of features provided by the studied tools
This review has also allowed us to identify groups of features commonly used by the tools included in this study. To extract these groups we reviewed the *Collaboration Features* column of the extraction form (Table 4). We specifically extracted the follow- ing groups of features:
Awareness: We identified that several types of awareness fea- tures are usually provided by the studied tools. For instance, in [28], WorldView and WAV are presented as tools that are able to identify the team structure. Another tool that attempts to improve awareness mechanisms is Augur, which includes a system to mon- itor developer's activities and explores the distribution of activities in time and space in order to explore the history and context of particular development activities in the code base [29].
Other tools attempt to address how to propagate any changes in the entire distributed team. For instance, FASTDASH, Palantir or

Syde, attempt to address this problem by providing real-time infor- mation about ongoing changes and warning developers about emerging conflicts [30]. More specifically, FASTDASH enables information to be accessed as regards which code files are being changed, who is changing them and how they are being used [31]. Change notification receivers should understand how these changes will make an impact on their work. Moreover, it is some- times relatively easy to ignore the work of others in decoupled dis- tributed teams because teams typically focus on their own models and ignore the dependent artifacts produced by others [32]. This problem is considered by SYSIPHUS, which was designed and implemented to enable teams to focus on overlaps between mod- els at different sites [32]. In this respect, ADAMS also supports event notifications by taking into account relevant events related
to the artifacts the engineer is working with [33].
Social Dependencies: Some of the studied tools integrate social features into IDEs to help developers save the time involved in switching between different tools and to enrich the collaborative IDE with social activities. For instance, in [35], SCI is presented as a solution to provide IDEs with ''social presence'' by including so- cial awareness and communication in a collaborative development environment. Other examples are FriendFeed, which is used in [34] to integrate and disseminate personal information into develop- ment environments.
Other tools associate social dependencies (''dependency be- tween developers as a result of the calls to each other's code'' [28]) with technical dependencies. One of the tools that the authors of ([28,34]) have selected in order to extract social depen- dencies from it, in this case, code files is Ariadne. This is a visual collaborative tool that highlights the socio-technical relationships between source-code artifacts and the developers implementing those artifacts in, for example, a repository [28].
Informal Communication: The need for informal communication and informal meetings has been taken into account by tools such as iBistro, which were designed to support the efficient capture, structure and navigation of meetings and their integration into the project [36].
Some of the studied tools have communication channels incor- porated into them. We thus mention, for example, CollabVS [37], CAMEL [38], CruiseControl [16], GENESIS [39], GoogleDocs [40] or GroupUML [41], which are not communication tools but which in- clude a chat to communicate with the other members while using the tool. This idea is mainly related to distributed design tools in which synchronous sessions need to be supported by communica- tion channels.
Knowledge Management: Architectural Knowledge (AK) Man- agement (AKM) needs to be adapted to today's distribution models [42]. In [43] the authors state that PAKME can be successfully used to help systematize the architecture knowledge management and evaluate the process of an industrial collaborator. LiveNet tool, de- spite not having been designed for AKM, is presented in [42] as being particularly applicable to capturing and encouraging the sharing of AK in distributed teams. To end this section, we should state that the information shown in the table can be consulted in the original papers from which the tools were extracted.
There are globally distributed companies such as IBM in which wikis serve as a platform for informal knowledge sharing among the collaborating teams [44]. There are also systems such as Col- labDev that have been specifically developed for large projects and large distribution of the team members. This allows specific application knowledge to be acquired and makes the knowledge available to the main stakeholders in order to solve maintenance problems [45].
It is also important to make the tools compatible with com- monly used formats or file types in order to facilitate their use and the sharing of information or knowledge. For instance,
SharePoint and Quickr allow knowledge workers to directly share information from Word, Excel and PowerPoint. They therefore sim- plify the sharing process since they do not have to save the file locally, and then transmit it via electronic mail or a Web-based upload [46].
Web-based version: One of the ideas used by some of the tools studied is that of implementing Web-based tools, thus allowing users to access them from anywhere with a simple Web-browser. Good examples of this are WebEx, which provides meeting services from a Web-browser [47], or WEB- DAV, which is a Web-based dis- tributed authoring and versioning system [46].
Wiki webs, such as WikiDev, are also a good example of Web- based systems that allow, in this case, information to be shared. WikiDev integrates information about various artifacts of interest, and uses clustering to obtain relevant artifacts and presents them in different views [48].
Data Integration: Software Development activities need to be supported by a set of tools such as version control systems, bug tracking systems, issue tracking systems, etc. The problem is that all these tools usually ''live in their own world'', are only loosely coupled and do not interact with each other [49]. Repoguard ad- dresses this problem by linking version control systems to other tools such as Mantis, Bugzilla and Trac.
This last analysis has allowed us to extract a set of the features commonly used by the GSE tools presented in this study. One of these features is that of awareness. This feature could help to keep the team members informed about activities that are being per- formed by other team members. Moreover, the awareness feature should consider social aspects including, for instance, personal information. Supporting informal communication is another of the key features that should be included in a tool, since there is a lack of informal information in distributed teams owing to the dif- ficulty of having face-to-face meetings. Offering interoperability among tools has been also detected as important in order to avoid information and

coordination breakdowns. Finally, allowing the formal and informal knowledge generated by the team members in a distributed environment to be managed has been also consid- ered to be a desirable feature.

Tools' evaluation

Efficient tool selection and evaluation processes are key issues in software engineering if development efficiency is to be in- creased [50]. It is therefore important to know whether the tools studied have been evaluated in a distributed environment. In this case, we have considered as valid evaluations those based on case studies, experiments, scenario-based evaluations, users' ratings and, of course, real project-based evaluations.

Moreover, we have separated tools that have been internally evaluated from those that have been externally evaluated. Here, internally evaluated refers to those tools that have been evaluated by the tool's builder/researcher. Examples of internal evaluation are presented in [28,51]. In [28] a tool called Ariadne is evaluated by the design team who perform an evaluation of Ariadne's visual- ization using inspection methods. In [51] the researchers perform two experiments to verify the effectiveness of the tool. In these cases we can state that we feel a certain degree of confidence that the tool will be used to fulfill its intended use, within its given context.

Externally evaluated tools are those tools whose performance has been tested outside the environment in which they were orig- inally built. In [52], the authors explain how the 4everedit tool was successfully evaluated in a large-scale industrial process engineer- ing project, while in [39], the GENESIS tool was evaluated by two industrial partners of the project. There is, therefore, a certain de- gree of confidence that these tools can be considered as useful tools for distributed environments.

A first result is that only 25.8% of the tools presented (see Table 15), that is, 33 out of 132 tools, have been evaluated in a distrib- uted environment. Moreover, not all evaluations show that the tool is really useful in a distributed environment because some of them are only preliminary evaluations since the tool is a prototype in the early phase of development, or an evaluation in a real environment is part of its developers' future work.

Although 132 tools are described in this review, it was not pos- sible to report whether all these tools have been evaluated or are used in practice because they are at an early stage of development, have been put forward as a theory, or the papers in which they ap- peared have not included a report on how they were validated or evaluated. 98 tools, that is, the 74.2% of the tools have not therefore been detected as evaluated tools. Moreover, the goal of a large number of papers was not to evaluate the tools, but to present them and their features, as occurs in [16], whose authors include a large number of tools which are classified by the process in which they can be used.

Table 16 lists those tools that have undergone some form of eval- uation as reported in the associated published paper. Clearly, tools not listed in this table may have been evaluated, but seeking this information outside the associated report is not within the scope of this study. Moreover, we include information about the goal of the tool, how the evaluation has been performed and the references in which an extended explanation of this evaluation can be found.

Another important result is that, as only 39% of the studied re- search tools presented any kind of evaluation (with only 24 individ- ual tools reporting a level of external validity), there is a need for studies in general to perform tool evaluation as a standard to pro- vide some assurance that the tools presented will be useful in GSE. Appendix B presents an extended version of Table 16 in which each tool is listed along with an explanation of how the evaluation was performed and a detailed description of each tool's intended use.

been sufficiently thorough to allow us to obtain the main features of each tool.

With regard to the construct validity, which is related to obtain- ing the right measure, the main challenge was to define the scope in relation to what is considered to be a GSE tool. In this respect, we considered GSE tools to be those presented as GSE tools in the studies, as well as those presented as DSD tools with a high le- vel of distribution (those tools that can be used in a highly distrib- uted environment).

In the case of the conclusion validity, which is concerned with the ability to replicate the same findings, we consider that the study has been validated through the systematic process and the periodic reviews carried out by the three researchers involved in this work. Moreover, in this work we have included sufficient details to allow the process to be reproduced. However, one possible problem in relation to this type of validity concerns the paper and tool classifi- cation. This was done by 3 researchers with the same background and belonging to the same research group; a slightly different clas- sification might therefore have been obtained if it had been done by other researchers from other groups. The number of results ob- tained from the searches might also have been different.

Conclusions

This work presents a systematic mapping review of GSE tools which was performed by following both the guidelines of Petersen et al. [1] and those of other important mapping reviews in the area of GSE.

After carrying out the literature review, a first conclusion was that there are no other systematic reviews of GSE tools. The most complete work in terms GSE tool research is [16], in which the authors present a set of tools classified by the area in which they can be used.

In the papers studied, the descriptions included are sometimes brief and do not provide sufficient information about the potential of the tools, nor do they allow users to discover which features they may offer. This makes the systematic process more complex, since it is necessary to introduce recursive searches if the main fea- tures of each tool are to be discovered.

With regard to the tools studied, we found that most of them focus on the subjects of *Virtual Meeting Tools* (12.2%), *Software Engi- neering Management Tools* (16%) and *Knowledge Management Tools* (16%). We believe that these results have been obtained because the main tools that GSE companies need are those related to com- munication, such as *Virtual Meeting Tools*. In the case of *Software Engineering Management Tools* we have found a large amount of tools, owing to the expansion of Web-based tools that provide pro- ject control from a simple Web-browser. Finally, the number of *Knowledge Management Tools* has increased because of the growing use of wikis to manage knowledge (also accessible from a Web- browser). In fact, these three subjects cover 44.2% of the tools found. However, other subjects such as *Socio Cultural Tools, Soft- ware Engineering Process* Tools or *Software Quality Tools* consist of only 3%, 3% and 5.3% of the total number of tools studied.

Bearing in mind that only 3% of the tools are related to socio-cul- tural aspects, perhaps it would be advisable to develop tools or tool features that will help group members to get to know each other and to facilitate communication by increasing the feeling of trust.

With regard to the percentages obtained, we can state that most of the tools found were applications developed in research groups or labs, or free tools, because 77.1% of the tools discovered are re- search or free tools (43.6% are research tools and 33.5% are free tools). This would appear to be logical, since the sources selected to search for the primary studies focus on research areas. As future work, we propose to carry out another review of GSE tools using other kinds of search engines, in order to obtain more commercial tools, such as *Jazz* tools *Rational Clearcase, Rational Requisite Pro,* etc.

We can also state that awareness features are usually supported by the studied tools at two levels (team activity awareness and so- cial awareness) to make team members feel closer to the rest of the team and have the best overview of what is happening in the project.

Supporting informal communication is usually considered by those of the studied tools that are focused on software design activities and it has also been detected that this support is com- monly integrated into the tool itself, principally if the tool allows synchronous collaboration. Moreover, in terms of integration, it seems that it might be useful to use ‘‘compatible’’ tools in order to share a common backbone, and thus save time and avoid incon- sistencies, incompatibilities and duplicated information that make distributed coordination and control more difficult. In fact, this may imply a lack of coordination among team members in relation to the information shared, because the information or data gener- ated by a tool cannot be used in other processes, owing to the format used.

We can thus conclude that the most common features provided by the set of 132 GSE tools included in this study are: Awareness as regards both the team members’ activities and social aspects; informal communication support; interoperability among tools; and formal and informal knowledge management.’’ One problem detected after studying the tools is that, although there are suffi- cient tools to support most areas or processes in the software life- cycle, there is a lack of connection between the tools. Almost only when using tools from the same company (i.e. IBM or Microsoft tools), and only in some areas, is it possible to integrate the differ- ent tools.

Another problem was the difficulty involved in studying
whether the tools were useful in a GSE domain. In this respect, the general rule was to consider all collaborative and Web-based tools as being useful for GSE. However, as future work, the list of tools presented in this study may be reviewed to test which partic- ular GSE task(s) each tool supports. To carry out this task, we are currently performing a survey which includes structured inter- views with practitioners to discover which tools are being used in companies and labs for GSE projects or experiments.

Finally, we plan to use this list of 132 tools to obtain informa- tion about the features that an integrated framework needs in or- der to develop a technological framework to support the complete software lifecycle in a GSE context.

Acknowledgements

This work has been funded by the PEGASO/MAGO Project (Min- isterio de Ciencia e Innovación MICINN, TIN2009-13718-C02-01) and co-funded by FEDER. It is also supported by ENGLOBAS (PII2I09-0147-8235) and GLOBALIA (PEII11-0291-5274), funded
by Consejería de Educación y Ciencia (Junta de Comunidades de Castilla-La Mancha) and co-funded by FEDER (Spain), as well as, ORIGIN (IDI-2010043 (1-5)) funded by CDTI and FEDER. This re- search is also supported, in part, by Science Foundation Ireland Grant 10/CE/I1855. Finally, we would also to thank the anonymous reviewers for their useful and constructive comments.

Appendix A

This section provides the primary studies selected from the sys- tematic review, sorted alphabetically:

List of primary studies in the systematic review

B. Al-Ani, et al., Continuous coordination within the context of cooperative and human aspects of software engineering, in: Proceedings of the 2008 International Workshop on Cooperative and human aspects of software engineering, ACM, Leipzig, Germany, 2008, pp. 1–4.
M. Ali-Babar, The application of knowledge-sharing workspace paradigm for software architecture processes, in: Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge, ACM, Leipzig, Germany, 2008, pp. 45–48.
M. Ali-Babar, et al., Introducing tool support for managing architectural knowledge: an experience report, in: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008), 2008, pp. 105–113.
J. Andrea, Envisioning the Next Generation of Functional Testing Tools, IEEE Software, 2007, pp. 58–66.

Appendix A (*continued*)
List of primary studies in the systematic review
Andreas Braun, Allen H. Dutoit, Andreas G. Harrer, Bernd Brüge, iBistro: A Learning Environment for Knowledge Construction in Distributed Software Engineering Courses apsec, pp. 197.
Y. Assogba, J. Donath, Share: a programming environment for loosely bound cooperation, in: Proceedings of the 28th International Conference on Human Factors in Computing Systems, ACM, Atlanta, Georgia, USA, 2010, pp. 961–970.
L. Aversano, et al., Managing coordination and cooperation in distributed software processes: the GENESIS environment. Software Process: Improvement and Practice 9(4) (2004) 239–263.
J.y. Bang, et al., CoDesign: a highly extensible collaborative software modeling framework, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering – Volume 2, ACM, Cape Town, South Africa, 2010, pp. 243–246.
H. Bani-Salameh, C. Jeffery, J. Al-Gharaibeh, SCI: towards a social collaborative integrated development environment, in: International Conference on Computational Science and Engineering, 2009, pp. 915– 920.
R. Bartholomew, Evaluating a networked virtual environment for globally distributed avionics software development, in: International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India, 2008, pp. 227–231.
K. Bauer, et al., WikiDev 2.0: discovering clusters of related team artifacts, in: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, ACM, Ontario, Canada, 2009, pp. 174–187.
J.T. Biehl, et al., FASTDash: a visual dashboard for fostering awareness in software teams, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, San Jose, California, USA, 2007, pp. 1313– 1322.
P. Bouillon, J. Krinke, Using Eclipse in distant teaching of software engineering, in: Proceedings of the 2004 OOPSLA Workshop on Eclipse Technology eXchange, ACM, Vancouver, British Columbia, Canada, 2004, pp. 22–26.
N. Boulila, Group support for distributed collaborative concurrent software modeling, in: 19th IEEE International Conference on Automated Software Engineering (ASE’04), Linz, Austria, 2004, pp. 422–425.
B. Bruegge, A.H. Dutoit, T. Wolf, Sysiphus: enabling informal collaboration in global software development, in: International Conference on Global Software Engineering (ICGSE’06), Florianopolis, Brazil, 2006, pp. 139–148.
B. Bruegge, et al., Supporting distributed software development with fine-grained artefact management, in: International Conference on Global Software Engineering (ICGSE’06), 2006, pp. 213–222
F. Calefato, F. Lanubile, Using frameworks to develop a distributed conferencing system: an experience report. Software: Practice and Experience 39(15) (2009) 1293–
1311. Appendix A (*continued*)
List of primary studies in the systematic review
F. Calefato, D. Gendarmi, F. Lanubile, Embedding social networking information into jazz to foster group awareness within distributed teams, in: Proceedings of the 2nd International Workshop on Social Software Engineering and Applications, ACM, Amsterdam, Netherlands, 2009, pp. 23–28.
K.M. Carley, et al., Toward an interoperable dynamic network analysis toolkit, Decision Support Systems 43(4) (2007) 1324–1347.

M. Cataldo, et al., CAMEL: a tool for collaborative distributed software design, in: IEEE International Conference on Global Software Engineering (ICGSE 2009), Limerick, Ireland, 2009, pp. 83–92.
A. De Lucia, et al., Enhancing collaborative synchronous UML modelling with fine-grained versioning of software artefacts, Journal of Visual Languages and Computing 18(5) (2007) 492–503.
K. Dullemond, B.v. Gameren, R.v. Solingen, How technological support can enable advantages of agile software development in a GSE setting, in: IEEE International Conference on Global Software Engineering (ICGSE 2009), Limerick, Ireland, 2009, pp. 143–152.
R.L. Edwards, J.K. Stewart, M. Ferati, Assessing the effectiveness of distributed pair programming for an online informatics curriculum, ACM Inroads 1(1) (2010) 48–54.
A. Fernández, et al., Guided support for collaborative modeling, enactment and simulation of software development processes, Software Process: Improvement and Practice 9(2) (2004) 95–106.
S. Fonseca, C.d. Souza, D. Redmiles, Exploring the relationship between dependencies and coordination to support global software development projects, in: International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 243–244.
J. Froehlich, P. Dourish, Unifying artifacts and activities in a visual tool for distributed software development teams, in: 26th International Conference on Software Engineering (ICSE'04), Edinburgh, Scotland, 2004, pp. 387–396.
V. Garousi, J. Leitch, IssuePlayer: an extensible framework for visual assessment of issue management in software development projects, Journal of Visual Languages and Computing 21(3) (2010) 121–135.
A. Gupta, S. Seshasai, 24-h knowledge factory: using Internet technology to leverage spatial and temporal separations, ACM Trans. Internet Technol. 7(3) (2007).
B. Hanks, Empirical evaluation of distributed pair programming, International Journal of Human– Computer Studies 66(7) (2008) 530–544.
L. Hattori, M. Lanza, Syde: a tool for collaborative software development, in: Proceedings of the 32nd ACM/ IEEE International Conference on Software Engineering – Volume 2, ACM, Cape Town, South Africa, 2010, pp. 235– 238.
S.R. Haynes, et al., Collaborative architecture design and evaluation, in: Proceedings of the 6th Conference on Designing Interactive Systems, ACM, University Park, PA, USA, 2006, pp. 219–228.

Appendix A (*continued*)
List of primary studies in the systematic review

M. Held, W. Blochinger, Structured collaborative workflow design, Future Generation Computer Systems 25(6) (2009) 638–653.
S. Kawaguchi, et al., MUDABlue: An automatic categorization system for Open Source repositories, Journal of Systems and Software 79(7) (2006) 939–953.
T. Krishnamurthy, S. Subramani, Ailments of Distributed Document Reviews and Remedies of DOCTOR (DOCument Tree ORganizer Tool) with distributed reviews support, in: IEEE International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India, 2008, pp. 210–214.
A. Lamersdorf, J. Munch, TAMRI: a tool for supporting task distribution in global software development projects, in: International Conference on Global Software Engineering 2009 (ICGSE 2009), Limerick, Ireland, 2009, pp. 322–327.
F. Lanubile, et al., Collaboration Tools for Global Software Engineering. Software Technology 27(2) (2010) 52–55.
Filippo Lanubile, Teresa Mallardo, Tool support for distributed inspection, in: Computer software and applications conference, annual international, 26th annual international computer software and applications conference, 2003, pp. 1071.
F. Lanubile, T. Mallardo, F. Calefato, Tool support for geographically dispersed inspection teams, Software Process: Improvement and Practice 8(4) (2003) 217–231.
L. Layman, et al., Essential communication practices for extreme programming in a global software development team, Information and Software Technology 48(9) (2006) 781–794.
M. Legenhausen, et al., RepoGuard: a framework for integration of development tools with source code repositories, in: Fourth IEEE International Conference on Global Software Engineering (ICGSE), Limerick, Ireland, 2009, pp. 328–331.
N.G. Lester, F.G. Wilkie, Evaluating UML tool support for effective coordination and communication across geographically disparate sites, in: 12 International Workshop on Software Technology and Engineering Practice (STEP'04), 2004, pp. 57–64.
I. Liccardi, CAWS: improving users' awareness in collaborative authoring activities, in: Group '07 Doctoral Consortium Papers, ACM, Sanibel Island, Florida, 2007, pp. 1–2.
R. Martignoni, Global sourcing of software development – a review of tools and services, in: Fourth IEEE International Conference on Global Software Engineering, Limerick, Ireland, 2009, pp. 303–308.
M. Meisinger, A. Rausch, M. Sihling, 4everedit – team- based process documentation management, Software Process: Improvement and Practice 11(6) (2006) 627–642.
B. Meyer, Design and code reviews in the age of the internet, Commun. ACM 51(9) (2008) 66–71.
R. Morgan, F. Maurer, MasePlanner: a card-based distributed planning tool for agile teams, in: IEEE International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 132–138.
Appendix A (*continued*)

List of primary studies in the systematic review

T. Niinimaki, A. Piri, C. Lassenius, Factors affecting audio and text-based communication media choice in global software development projects, in: IEEE International Conference on Global Software Engineering 2009 (ICGSE 2009), Limerick, Ireland, 2009, pp. 153–162.
S. Paul, et al., Impact of heterogeneity and collaborative conflict management style on the performance of synchronous global virtual teams, Information and Management 41(3) (2004) 303–321.
C. Pickering, et al., 3D global virtual teaming environment, in: Fourth International Conference on Creating, Connecting and Collaborating through Computing (C5'06), Berkeley, California, 2006, pp. 126– 135.
S. Salinger, et al., Saros: an eclipse plug-in for distributed party programming, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Cape Town, South Africa, 2010, pp. 48–55.
S. Sarkar, R. Sindhgatta, K. Pooloth, A collaborative platform for application knowledge management in software maintenance projects, in: Proceedings of the 1st Bangalore Annual Compute Conference, Bangalore, India, 2008, pp. 1–7.
A. Sarma, et al., Tesseract: Interactive visual exploration of socio-technical relationships in software development, in: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, Vancouver, BC, Canada, 2009, pp. 23–33.
F. Servant, et al., CASI: preventing indirect conflicts through a live visualization, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Cape Town, South Africa, 2010, pp. 39–46.
N. Seyff, et al., Enhancing GSS-based requirements negotiation with distributed and mobile tools, in: 14th IEEE International Workshops on Enabling

Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), Linkoping, Sweden, 2005, pp. 87–92.
V. Sinha, B. Sengupta, S. Chandra, Enabling Collaboration in Distributed Requirements Management, IEEE Software, 2006, pp. 52–61.
H. Spanjers, et al. Tool support for distributed software engineering, in: International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 187–198.
H. Su, S. Jodis, H. Zhang, Providing an integrated software development environment for undergraduate software engineering courses, J. Comput. Small Coll. 23(2) (2007) 143–149.
A. Tang, et al., A comparative study of architecture knowledge management tools, Journal of Systems and Software, in press, Corrected Proof., 2009, pp. 352– 370.
M.R. Thissen, et al., Communication tools for distributed software development teams, in: Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce, ACM, St. Louis, Missouri, USA, 2007, pp. 28–35.

Appendix B (*continued*)

Tool Designed to Evaluation

E. Trainer, et al., Analyzing a socio-technical visualization tool using usability inspection methods, IEEE Symposium on Visual Languages and Human-Centric Computing, 2008, pp. 78–81.
C. Treude, The role of emergent knowledge structures in collaborative software development, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering – Volume 2, ACM, Cape Town, South Africa, 2010, pp. 389–392.
C. Treude, M.-A. Storey, Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering – Volume 1, ACM, Cape Town, South Africa, 2010, pp. 365– 374.
C. Treude, M.-A. Storey, How tagging helps bridge the gap between social and technical aspects in software development, in: International Conference on Software Engineering 2009 (ICGSE 2009), Limerick, Ireland, 2009,
pp. 12–22.
J. Whitehead, Collaboration in software engineering: a roadmap, in: 2007 Future of Software Engineering, IEEE Computer Society, 2007, pp. 214–225.
W. Xiao, C. Chi, M. Yang, On-line collaborative software development via wiki, in: Proceedings of the 2007 International Symposium on Wikis, ACM, Montreal, Quebec, Canada, 2007, pp. 177–183.
D. Xu, et al., Distributed collaborative modeling support system associating UML diagrams with chat messages, in: 33rd Annual IEEE International Computer Software and Applications Conference, 2009, pp. 367–372.

References

K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), Bari, Italy, 2008.

ARENA Negotiate
requirements in a distributed manner EGRET Support global software development teams in collaborating on requirements management SoftFab Automate building and test processes
To evaluate the
distributed and mobile negotiation tools (ARENA II and ARENA-
M respectively) an initial evaluation study was performed to investigate whether stakeholders are able to successfully use the tools, to identify usability flaws, and to identify major differences in usage between them [63] The tool received positive reviews in various communities within IBM, including practitioners and tool builders. Reviewers felt that the persistence of ad hoc discussions with remote team members would enable ''knowledge logging'' while the use of traceability to communicate requirement changes would help ''enforce accountability'' [60]
A case study was
conducted to investigate the applicability of SoftFab in collaborations that involve multiple partners. This case study was used to show the problems that are encountered in such a collaboration, and how SoftFab is setup and
used [6]

D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using Mapping Studies in Software Engineering, in: PPIG, 2008, pp. 195–204.
P.J. Ågerfalk, B. Fitzgerald, H.H. Olsson, E.Ó. Conchúir, Benefits of global software development: the known and unknown, in: International Conference on Software Process, ICSP 2008, Leipzig, Germany, Springer, Berlin/Heidelberg, 2008, pp. 1–9.
K. Dullemond, B.v. Gameren, Technological Support for distributed agile development, in: Department of Software Technology, Delf University of Technology, Delf, 2009, p. 223.
A.A. Keshlaf, S. Riddle, Risk management for web and distributed software development projects, in: Fifth International Conference on Internet Monitoring and Protection, Barcelona, Spain, 2010, pp. 22–28.
H. Spanjers, M.t. Huurneç, B. Graaf, M. Lormans, D. Bendas, R. van, Tool support for distributed software engineering, in: International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 187–198.
C. Ebert, Global Software Engineering: Distributed Development, Outsourcing, and Supplier Management, Wiley, IEEE Computer Society Books, Los Alamitos, USA, 2010.
E. Carmel, Global Software Teams: Collaborating Across Borders and Time Zones, Prentice Hall PTR, 1999. 269.
J.D. Herbsleb, A. Mockus, T.A. Finholt, R.E. Grinter, Distance, dependencies and delay in a global collaboration, in: ACM Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, 2000.
R. Prikladnicki, L. Pilatti, Improving contextual skills in global software engineering: a corporate training experience, in: IEEE International Conference on Global Software Engineering (ICGSE'08), IEEE Computer Society, Bangalore, India, 2008, pp. 239–243.
K. Berkling, M. Geisser, T. Hildenbrand, F. Rothlauf, Offshore software development: transferring research findings into the classroom, in: S. Berlin (Ed.), Software Engineering Approaches for Offshore and Outsourced Development, Heidelberg, 2007, pp. 1–18.
B. Lutz, Linguistic challenges in global software development: lessons learned in an international SW development division, in: Fourth IEEE International Conference on Global Software Engineering (ICGSE'09), IEEE Computer Society, Limerick, Ireland, 2009, pp. 249–253.
D. Damian, A. Hadwin, B. Al-Ani, Instructional design and assessment strategies for teaching global software development: a framework, in: International Conference on Software Engineering (ICSE'06), Shanghai, China, ACM Press, New York, NY, USA, 2006.
J. Favela, F. Peña-Mora, An experience in collaborative software engineering education, IEEE Software 18 (2) (2001) 47–53.
D. Petkovic, G.D. Thompson, R. Todtenhoefer, Assessment and comparison of local and global SW engineering practices in a classroom setting, in: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, ACM, Madrid, Spain, 2008, pp. 78–82.
F. Lanubile, C. Ebert, C. Prikladnicki, A. Vizcaíno, Collaboration tools for global software engineering, IEEE Software 27 (2) (2010) 52–55.
B. Sengupta, S. Chandra, V. Sinha, A research agenda for distributed software development, in: Proceedings of the 28th International Conference on Software Engineering, ACM, Shanghai, China, 2006.
C. Laurent, A sensitivity analysis approach to select IT-tools for global development projects, in: Tool Support and Requirements Management in Distributed Project, Munich, Germany, 2007, pp. 38–42.
F.Q.B.d. Silva, C. Costa, A. Cesar C. França, R. Prikladinicki, Challenges and solutions in distributed software development project management: a systematic literature review, in: International Conference on Global Software Development (ICGSE 2010) Princeton, NJ, USA, 2010.
S. Jalali, Agile practices in global software engineering – a systematic map, in: 2010 5th IEEE International Conference on Global Software Engineering, Princeton, New Jersey, USA, 2010.
D. Šmite, C. Wohlin, T. Gorschek, R. Feldt, Empirical evidence in global software

Libra-on- chat
Distributed synchronous collaborative modeling support system for UML diagrams

It was evaluated through two experiments, the first to validate the effectiveness of association of conversations with model elements and the second to validate how well the system supports utilizing the stored conversation contents [51]

engineering: a systematic review, Empirical Software Engineering 15 (1) (2010) 91–118.
E. Hossain, M. Ali-Babar, H. Paik, Using scrum in global software development: a systematic literature review, in: Fourth IEEE International Conference on Global Software Engineering (ICGSE'09), IEEE Computer Society, Limerick, Ireland, 2009, pp. 175–184.
T.L. Friedman, The World is Flat: Brief History of the 21st Century, Farrar, Straus and Girou, New York, 2005.
B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, in EBSE Technical Report, 2007.
A. Abran, J.W. Moore, Guide to the software engineering body of knowledge (SWEBOK®), in: IEEE Computer Society 2004 Guide, 2004.
A. Fuggetta, A classification of CASE technology, Computer 26 (12) (1993) 25–38.
A. Boden, G. Avram, L. Bannon, V. Wulf, Knowledge management in distributed software development teams – does culture matter? in: International Conference on Global Software Engineering, Limerick, Ireland, 2009.

B. Al-Ani, E. Trainer, R. Ripley, A. Sarma, André v.d. Hoek, D. Redmiles, Continuous coordination within the context of cooperative and human aspects of software engineering, in: Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Leipzig, Germany, 2008, pp. 1–4.
J. Froehlich, P. Dourish, Unifying artifacts and activities in a visual tool for distributed software development teams, in: Proceedings of the 26th International Conference on Software Engineering, IEEE Computer Society, 2004, pp. 387–396.
L. Hattori, M. Lanza, Syde: a tool for collaborative software development, Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 2, ACM, Cape Town, South Africa, 2010, pp. 235–238.
J.T. Biehl, M. Czerwinski, G. Smith, G.G. Robertson, FASTDash: a visual dashboard for fostering awareness in software teams, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, San Jose, California, USA, 2007, pp. 1313–1322.
B. Bruegge, A.H. Dutoit, T. Wolf, Sysiphus: enabling informal collaboration in global software development, in: International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 139–148.
B. Bruegge, A.D. Lucia, F. Fasano, G. Tortora, Supporting distributed software development with fine-grained artefact management, in: Proceedings of the IEEE International Conference on Global Software Engineering, IEEE Computer Society, 2006, pp. 213–222.
F. Calefato, D. Gendarmi, F. Lanubile, Embedding social networking information into jazz to foster group awareness within distributed teams, in: Proceedings of the 2nd International Workshop on Social Software Engineering and Applications, ACM, Amsterdam, The Netherlands, 2009, pp. 23–28.
H. Bani-Salameh, C. Jeffery, J. Al-Gharaibeh, SCI: towards a social collaborative integrated development environment, Proceedings of the 2009 International Conference on Computational Science and Engineering, vol. 04, IEEE Computer Society, 2009, pp. 915–920.
A. Braun, A.H. Dutoit, A.G. Harrer, B. Brüge, iBistro: a learning environment for knowledge construction in distributed software engineering courses, in: Proceedings of the Ninth Asia-Pacific Software Engineering Conference, IEEE Computer Society, 2002, pp. 197–203.
A. Sarma, L. Maccherone, P. Wagstrom, J. Herbsleb, Tesseract: interactive visual exploration of socio-technical relationships in software development, in: Proceedings of the 31st International Conference on Software Engineering, IEEE Computer Society, 2009, pp. 23–33.
M. Cataldo, C. Shelton, Y. Choi, Y.-Y. Huang, V. Ramesh, D. Saini, L.-Y. Wang, CAMEL: a tool for collaborative distributed software design, in: International Conference on Global Software Engineering, Limerick, Ireland, 2009.
L. Aversano, A.D. Lucia, M. Gaeta, P. Ritrovato, S. Stefanucci, M.L. Villani, Managing coordination and cooperation in distributed software processes: the GENESIS environment, Software Process: Improvement and Practice 9 (4) (2004) 239–263.
B. Meyer, Design and code reviews in the age of the internet, Communications of the ACM 51 (9) (2008) 66–71.
N. Boulila, Group support for distributed collaborative concurrent software modeling, in: 19th IEEE International Conference on Automated Software Engineering (ASE'04), Linz, Austria, 2004, pp. 422–425.
M. Ali-Babar, The application of knowledge-sharing workspace paradigm for software architecture processes, in: Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge, ACM, Leipzig, Germany, 2008, pp. 45–48.
M. Ali-Babar, A. Northway, I. Gorton, P. Heuer, T. Nguyen, Introducing tool support for managing architectural knowledge: an experience report, in: Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, IEEE Computer Society, 2008, pp. 105–113.
C. Hill, R. Yates, C. Jones, S.L. Kogan, Beyond predictable workflows: enhancing productivity in artful business processes, IBM Systems Journal 45 (4) (2006) 663–682.
S. Sarkar, R. Sindhgatta, K. Pooloth, A collaborative platform for application knowledge management in software maintenance projects, in: Proceedings of the 1st Bangalore Annual Compute Conference, ACM, Bangalore, India, 2008, pp. 1–7.
A. Gupta, S. Seshasai, 24-hour knowledge factory: using Internet technology to leverage spatial and temporal separations, ACM Transactions on Internet Technology 7 (3) (2007) 14. M.R. Thissen, J.M. Page, M.C. Bharathi, T.L. Austin, Communication tools for distributed software development teams, in: Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce, ACM, St. Louis, Missouri, USA, 2007, pp. 28–35.
K. Bauer, M. Fokaefs, B. Tansey, E. Stroulia, WikiDev 20: discovering clusters of related team artifacts, in: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, ACM, Ontario, Canada, 2009, pp. 174–187.
M. Legenhausen, S. Pielicke, J. Ruhmkorf, H. Wendel, A. Schreiber, RepoGuard: a framework for integration of development tools with source code repositories, in: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering, IEEE Computer Society, 2009, pp. 328–331.
D. Winkler, S. Biffl, A. Kaltenbach, Evaluating tools that support pair programming in a distributed engineering environment, in: 14th International Conference on Evaluation and Assessment in Software Engineering (EASE), Keele University, UK, 2010.
D. Xu, J. Kurogi, Y. Ohgame, A. Hazeyama, Distributed collaborative modeling support system associating UML diagrams with chat messages, Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference, vol. 01, IEEE Computer Society, 2009, pp. 367–372.
M. Meisinger, A. Rausch, M. Sihling, 4everedit – team-based process documentation management, Software Process: Improvement and Practice. 11 (6) (2006) 627–642.
N.G. Lester, F.G. Wilkie, Evaluating UML tool support for effective coordination and communication across geographically disparate sites, in: Proceedings of the 12 International Workshop on Software Technology and Engineering Practice, IEEE Computer Society, 2004, pp. 57–64.
V. Garousi, J. Leitch, IssuePlayer: an extensible framework for visual assessment of issue management in software development projects, Journal of Visual Languages and Computing 21 (3) (2010) 121–135.
Y. Assogba, J. Donath, Share: a programming environment for loosely bound cooperation, in: Proceedings of the 28th International Conference on Human Factors in Computing Systems, ACM, Atlanta, Georgia, USA, 2010, pp. 961–970.
S. Kawaguchi, P.K. Garg, M. Matsushita, K. Inoue, MUDABlue: an automatic categorization system for open source repositories, Journal of Systems and Software 79 (7) (2006) 939–953.
T. Krishnamurthy, S. Subramani, Ailments of distributed document reviews and remedies of DOCTOR (DOCument Tree ORganizer Tool) with distributed reviews support, i:n IEEE International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India, 2008, pp. 210–214.
A. Fernández, B. Garzaldeen, I. Grützner, J. Münch, Guided support for collaborative modeling, enactment and simulation of software development processes, Software Process: Improvement and Practice. 9 (2) (2004) 95–106.
R.L. Edwards, J.K. Stewart, M. Ferati, Assessing the effectiveness of distributed pair programming for an online informatics curriculum, ACM Inroads 1 (1) (2010) 48–54.
V. Sinha, B. Sengupta, S. Chandra, Enabling collaboration in distributed requirements management, IEEE Software 23 (5) (2006) 52–61.
R. Bartholomew, Evaluating a networked virtual environment for globally distributed avionics software development, in: Proceedings of the 2008 IEEE International Conference on Global Software Engineering, IEEE Computer Society, 2008, pp. 227–231.
S.R. Haynes, A.L. Skattebo, J.A. Singel, M.A. Cohen, J.L. Himelright, Collaborative architecture design and evaluation, in: Proceedings of the 6th Conference on Designing Interactive Systems, ACM, University Park, PA, USA, 2006, pp. 219– 228.
S. Norbert, Enhancing GSS-based Requirements Negotiation with Distributed and Mobile Tools, 2005.
L. Layman, L. Williams, D. Damian, H. Bures, Essential communication practices for extreme programming in a global software development team, Information and Software Technology 48 (9) (2006) 781–794.
S. Salinger, C. Oezbek, K. Beecher, J. Schenk, Saros: an eclipse plug-in for distributed party programming, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Cape Town, South Africa, 2010, pp. 48–55.
F. Servant, J.A. Jones, A.v.d. Hoek, CASI: preventing indirect conflicts through a live visualization, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Cape Town, South Africa, 2010, pp. 39–46.