

Tools used in Global Software Engineering: A systematic mapping review

Javier Portillo-Rodríguez^{a,*}, Aurora Vizcaíno^a, Mario Piattini^a, Sarah Beecham^b

^aAlarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain

^bLero-The Irish Software Engineering Research Centre, University of Limerick, Ireland

ARTICLE INFO

Article history:

Received 8 June 2010

Received in revised form 24 February 2012

Accepted 28 February 2012

Available online 7 March 2012

Keywords:

Global Software Development
Distributed Software Engineering
Tool
Systematic Mapping Study

ABSTRACT

Context: This systematic mapping review is set in a Global Software Engineering (GSE) context, characterized 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 tool support in GSE.

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 communication, 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 percentage of tools (18.9%) are reported as having been tested outside the initial context in which they were developed.

Conclusion: The most common features in the GSE tools included in this study are: team activity and social awareness, support for informal communication, Support for Distributed Knowledge Management and Interoperability with other tools. Finally, there is the need for an evaluation of these tools to verify their external validity, or usefulness in a wider global environment.

© 2012 Elsevier B.V. All rights reserved.

Contents

1. Introduction	664
2. Systematic mapping review of tools to support GSE	665
2.1. Definition of research question	665
2.2. Conducting the search	665
2.3. Screening of papers and Keywording of Abstracts	665
2.4. Data/information extraction and mapping of studies	666
2.4.1. Validation of tool classification scheme	667
3. Results and discussion	668
3.1. Classification of features	668
3.2. Tool classification and description	671

* Corresponding author.

E-mail addresses: javier.portillo@uclm.es (J. Portillo-Rodríguez), aurora.vizcaino@uclm.es (A. Vizcaíno), mario.piattini@uclm.es (M. Piattini), sarah.beecham@lero.ie (S. Beecham).

3.2.1.	Tools and features used in each knowledge area	672
3.2.1.1.	Requirement Tools (SRTs)	672
3.2.1.2.	Design Tools (SDTs)	672
3.2.1.3.	Construction Tools (SCTs)	672
3.2.1.4.	Testing Tools (STTs)	672
3.2.1.5.	Maintenance Tools (SMTs)	672
3.2.1.6.	Configuration Management Tools (SCMTs)	672
3.2.1.7.	Engineering Management Tools (SEMTs)	675
3.2.1.8.	Engineering Process Tools (SEPTs)	675
3.2.1.9.	Quality Tools (SQTs)	675
3.2.1.10.	Miscellaneous Tool Issues (MTIs)	675
3.2.1.11.	Knowledge Management Tools (KMTs)	675
3.2.1.12.	Virtual Meeting Tools (VMTs)	675
3.2.1.13.	Socio-Cultural Tools (S-CTs)	675
3.2.2.	Common groups of features provided by the studied tools	675
3.2.3.	Tools' evaluation	676
4.	Threats to validity	677
5.	Conclusions and future work	678
	Acknowledgements	678
	Appendix A	678
	Appendix B	681
	References	684

1. Introduction

Global Software Engineering (GSE) has become a growing area of research, apart from being an expanding trend in the Information 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 principally come about as a result of the distance factor (temporal, geographic 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 collaboration with other partners and in different geographical locations is a great challenge for organizations [6,7].

Software tools for GSE should therefore help to alleviate problems 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 environment; (c) *Loss of Communication*; this is the case in this type of environment, if we consider that the richest communication medium 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 interaction 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 Engineering (DSE) often include interesting features that may be useful in a global environment. Moreover, according to the study presented in [17], collaborative software tools for distributed development 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 globally-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 mapping 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 summary (map) of classified features. The classification of the tools will be based on an extraction of common features across studies (principally related to communication, owing to its importance).

Companies, practitioners and researchers may find this work useful, 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 support. 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 studies, 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 conclusions obtained.

2. Systematic mapping review of tools to support GSE

The systematic mapping review which follows has been developed by using the guidelines presented in [1] and taking into account 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.

2.1. 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 developed 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 request 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 following results once the systematic mapping review had been completed. 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, composed 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.

2.2. 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 related 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 research area is widely dealt with, such as: Information and Software Technology, IEEE Software, Computer, Information and Management, and Systems and Software. Moreover, from IEEE we included the proceedings of the most relevant conference on GSE (*International 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 evolution 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.

2.3. Screening of papers and Keywording of Abstracts

The goal of this step was to identify the relevant papers, with regard both to the objectives of this review and to the scope of the research questions. The main difficulty in achieving this goal was that the terms used in the question led to results which were far too wide-ranging. For example, the word “tool” is broadly used in many kinds of publication, and a large amount of papers therefore appeared in the first results obtained, as is shown in Table 3.

The step used to classify the scheme (Keywording of Abstract) is not clearly described in [1] (the paper followed to perform this study). That being the case, in order to obtain the most valuable papers and to achieve a more in-depth analysis in relation to our research goal and the research question, we defined four stages

Table 1
Search string.

(tool or technology) AND ((global OR distributed) AND (software development OR software engineering))

Table 2
Search strings by source.

Source	Specific search string	Extra options included
Science Direct	pub-date > 1999 and ((tool or technology) and (“global software development” or “global software engineering” or “distributed software development” or “distributed software engineering”))	Selection of “Computer Science” as subject in <i>Expert Search menu</i>
Wiley InterScience	((tool or technology) and (“global software development” or “global software engineering” or “distributed software development” or “distributed software engineering”))	Selection of publications from the year 2000
IEEE CS	((tool or technology) and (“global software development” or “global software engineering” or “distributed software development” or “distributed software engineering”))	Unmarked searching in ACM Digital Library and selection of publications from the year 2000
ACM Digital Library	((tool or technology) and (“global software development” or “global software engineering” or “distributed software development” or “distributed software engineering”)) and (Publisher:ACM))	Selection of publications from the year 2000

Table 3
Selection of studies.

Phase	Inclusion criteria	Papers remaining			
		SD	WI	IEEE	ACM
Based on Search	English language, from the year 2000, containing the search string terms	174	43	836	385
Exclusion upon Title	Does not include proceedings frontmatter/backmatter, notes, comments, workshops or letters and repeated works	168	42	734	271
Exclusion upon Abstract	Discusses tool support for GSD or DSD	18	7	74	48
Exclusion upon Full Text	Presents specific tools or systems for GSD or DSD	8	5	30	23

(summarized in Table 3), following the indications provided in [24].

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 initial 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 eliminating 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 articles, workshop papers and conference papers. A result was considered 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 abstract was not sufficient, and that it would also be necessary to review 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 specific 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 (communication 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 primary 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 paper may include several tools.

What is more, in order to avoid missing any important information about a tool, we checked the information provided in the other papers that had been rejected. In all cases, the information provided 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.

2.4. Data/information extraction and mapping of studies

Once the primary studies had been chosen, the relevant information for the systematic review was obtained. The inclusion criterion 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 format 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.

Table 4

Data extraction form.

Information extraction from the study								
Study ID	20							
Title	CAMEL: A Tool for Collaborative Distributed Software Design							
Authors	Marcelo Cataldo, Charles Shelton, Yongjoon Choi, Yun-Yin Huang, Vytresh Ramesh, Darpan Saini, Liang-Yun Wang							
Publishing site	IEEE International Conference on Global Software Engineering 2009 (ICGSE 2009)							
Publishing date	July, 2009							
Context	Software design activities require rich communication channels in which developers can exchange information in multiple ways. It is well established that geographic distribution makes a negative impact on the effectiveness of design meetings							
Tool focused	Yes							
Name	URL	Description	Area	License	Collaboration features	Designed for GSE	Tested in GSE	Benefits for GSE
Tools discovered (list of 1 or more tools)								
CAMEL	http://code.google.com/p/cameldesigntool/	Collaborative tool to support the design process	Design	Research	<ul style="list-style-type: none"> – This tool allows synchronous design meetings – Includes a chat tool – Includes awareness features such as action awareness through color identification – Includes a team view to identify team members 	Yes	No	Helps to reduce geographical distance

Information about the studies was obtained by considering the following column: study id, to identify each study, title of the study, the authors, publishing site (journal, conference proceedings, etc.), its date, and the context of the study (mainly to discover in which area the tools can be used and whether the study is focused on tools or whether the tools are only a complementary topic of the study).

The following columns were deemed useful to obtain information about each tool: name of the tool, URL from which it can be downloaded (or in which it is described), a brief description of the tool and the area in which it is used, considering the areas defined in the SWEBOK [25] and three more areas (Knowledge Management Tools, Virtual Meeting Tools and Socio-Cultural Tools) not included in the SWEBOK (all the areas considered will be defined in Section 3.1).

The following were also obtained: information about the tools' "license" (commercial, research or free); which collaboration features (useful in GSE) it includes; whether the tool has been specifically designed for GSE or has not been designed for GSE but can be used in it; information about whether the tool has been tested in a GSE context; and finally, what particular benefits in terms of distance (geographical distance, temporal distance and socio-cultural distance) can be obtained by using the tool.

We consider features that help to confront GSE challenges to be those which focus on supporting distributed communication, those supporting distributed coordination or control and features that help to reduce socio-cultural distance between distributed team members. The goal of including each tool's characteristics is to obtain a set of common features that the GSE tools include. This set of feature categories will be useful in classifying the tools found.

Extracting information about each tool's characteristics was not a simple task, since in some of the primary studies the information about the tool features was insufficient. In these cases, we consulted the tool's website (where one existed) to extract information about features. Moreover, it was impossible for the information extracted to be as complete as we would have liked, since the tools might have features that could only be recognized by actually installing and using the tool, which was not within the scope of this study.

The information gathered by using the aforementioned table made it possible to extract an initial map to show how the primary studies are distributed, in terms of time and the type of tool presented in each one.

The map in Fig. 1 (right-hand side) shows the total number of selected papers. It will be noted 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 Management 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 Engineering Process Tools (SEPTs), are only dealt with in a few publications. 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 Commercial Tools (30 tools). However, some free, commercial or research 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.

2.4.1. 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 agreement 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 raters. 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' classification. 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.

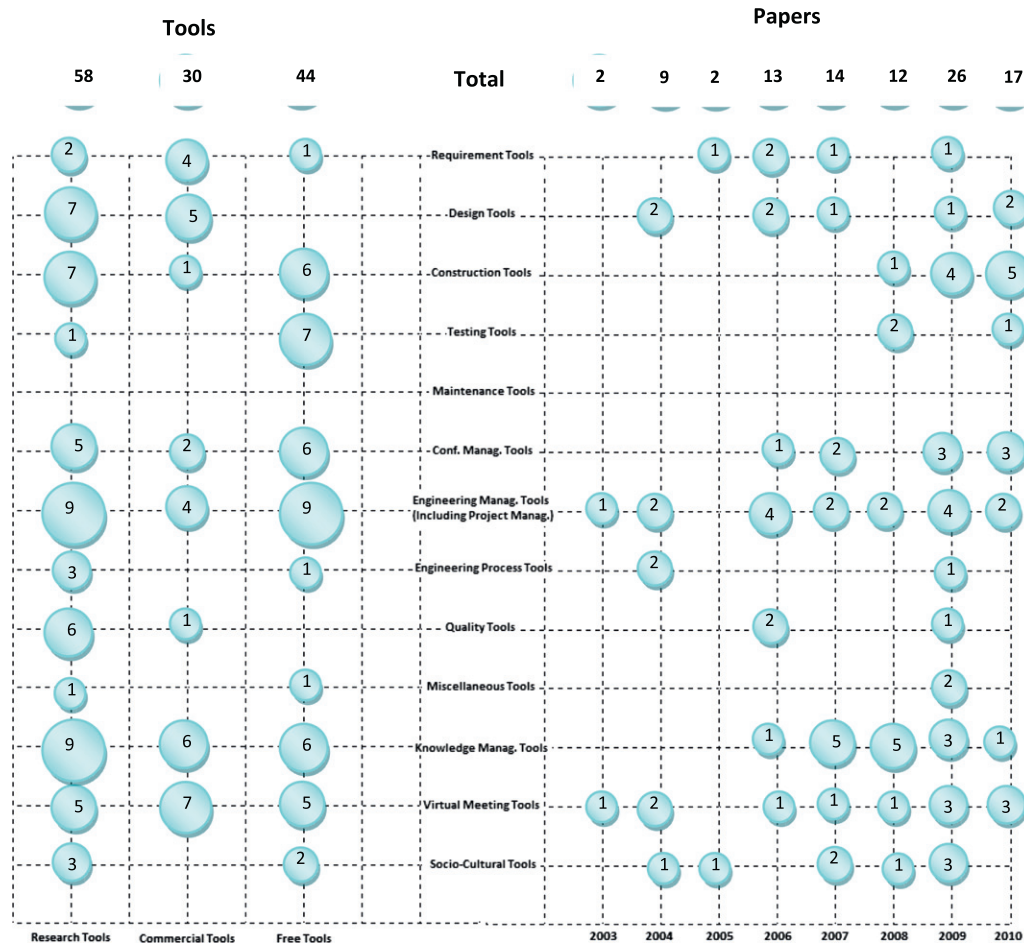


Fig. 1. Map by years.

Table 5 Inter-rater reliability analysis data and results.

	Categories	Subjects	Raters	Fleiss Kappa
Tools	13	132	3	0.952
Papers	13	95	3	0.966

3. Results and discussion

In this section we present the results obtained from the systematic 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 results 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 review 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 included in them could be categorized. These feature categories are part of the expected results and are presented in the following subsection.

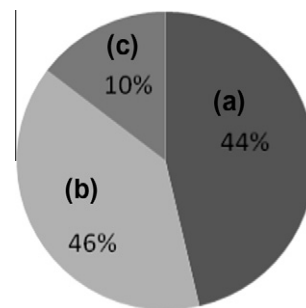


Fig. 2. Content of primary studies: (a) a tool for an area, (b) several tools for an area, and (c) several tools for several areas.

3.1. 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 described as a UML modeler, it will be classified in the design subject. This classification was carried out by considering the use of different 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

Table 6
SWEBOK subjects.

SWEBOK subjects	Abbreviation
Software Requirement Tools	SRT
Software Design Tools	SDT
Software Construction Tools	SCT
Software Testing Tools	STT
Software Maintenance Tools	SMT
Software Configuration Management Tools	SCMT
Software Engineering Management Tools	SEMT
Software Engineering Process Tools	SEPT
Software Quality Tools	SQT
Miscellaneous Tool Issues	MTI
<i>Extended subjects</i>	
Knowledge Management Tools	KMT
Virtual Meeting Tools	VMT
Socio-Cultural Tools	S-CT

[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 oriented 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 subjects not included in software engineering. For example, the subjects 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 knowledge 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 Process-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 *Integration 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 license is associated with each tool. We have thus defined three types of categories in relation to their licenses. These types are described 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 attain 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 domain 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 construction (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, videoconference 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-

Table 7
Distribution of primary studies by subject.

Subject	Primary studies	%
Software Requirement Tools (SRTs)	15, 54, 55, 63, 64	5.2
Software Design Tools (SDTs)	8, 14, 15, 21, 31, 36, 41, 66	8.4
Software Construction Tools (SCTs)	6, 9, 18, 29, 30, 36, 43, 50, 61, 63	10.5
Software Testing Tools (STTs)	4, 36, 56	3.1
Software Maintenance Tools (SMTs)	–	0
Software Configuration Management Tools (SCMTs)	11, 12, 33, 36, 40, 43, 53, 57, 62	9.4
Software Engineering Management Tools (SEMTs)	1, 11, 13, 16, 18, 21, 25, 26, 27, 29, 35,36, 39, 40, 46, 64, 66	17.8
Software Engineering Process Tools (SEPTs)	7, 32, 24	3.1
Software Quality Tools (SQTs)	38, 43, 37	3.1
Miscellaneous Tool Issues (MTIs)	22, 40	2.1
Knowledge Management Tools (KMTs)	3, 44, 59, 42, 51, 34, 66, 58, 5, 28, 36, 2, 43, 45, 65	15.7
Virtual Meeting Tools (VMTs)	61, 23, 48, 9, 17, 36, 47, 49, 10, 58, 38, 14	12.6
Socio-Cultural Tools (S-CTs)	57, 20, 18, 52, 54, 24, 19, 60	8.4

Table 8
Tool licenses.

License type	Abbreviation	Description
Commercial	C	This type of tool includes those whose license can be obtained through payment
Free	F	This type of tool includes those whose license can be obtained without payment. In this license type we include licenses such as Apache License, GPL, etc.
Research	R	This type of tool includes tools or prototypes developed in (and by) research groups/labs

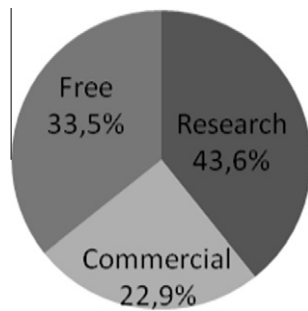


Fig. 3. Tool distribution by license.

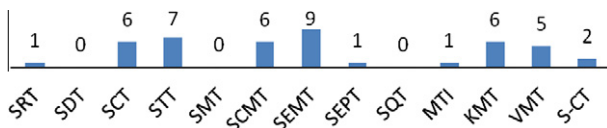


Fig. 4. Number of free tools supporting each area.

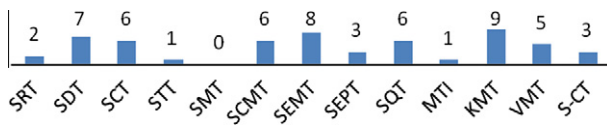


Fig. 5. Number of research tools supporting each area.

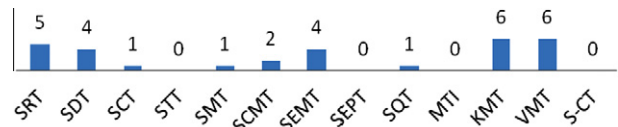


Fig. 6. Number of commercial tools supporting each area.

Table 9
Communication features.

Communication type	Abbreviation	Description
Synchronous	V	Video-conference
Synchronous	C	Chat
Synchronous	A	Audio-conference (include VoIP or similar)
Asynchronous	F	Forum or discussion thread
Asynchronous	E	E-mail
Asynchronous	Cmts	Comments (usually related to system items)

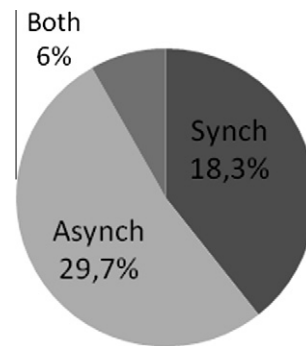


Fig. 7. Communication channel distribution.

nous communication (29.7%) (see Fig. 7). This would appear to be logical, since time differences commonly occur in globally distributed environments, making the constant use of synchronous communication difficult.

The fourth category is **Awareness**. With this category we have attempted to describe those features related to awareness management. We have therefore classified the awareness features into two groups, the first of which is *Presence Awareness*. This type is mainly related to showing who is connected to a session, or where a team member is located. Presence awareness can thus be considered as *Session Awareness* or *Global Awareness*, respectively. *Global Awareness* consists of showing where other team members are located geographically.

In this category we also consider another type of awareness, defined as *Change Awareness*. This type of awareness considers those features related to letting users know “who is doing what”, independently of whether team members are working synchronously or asynchronously. We have identified two main features to support 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 category includes features that principally assist managers with control 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 systems also assists in aspects of coordination, since managers have an overview of the project’s progress and are able to make coordination adjustments. In addition, these kinds of systems are complemented 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, knowledge 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 networks. 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.

3.2. 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 includes 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 possibilities 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 information 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-

Table 10
Awareness features.

Awareness type	Abbreviation	Description
Presence	G	Global Awareness
Presence	S	Session Awareness
Changes	V	Visual Awareness
Changes	EN	E-mail Notifications

Table 11
Control and coordination features.

Type of tool	Abbreviation
Issue Tracking System	ITS
Version Control System	VCS
Build Management System	BMS

Table 12
Knowledge management features.

Feature	Value
Wiki	W
Document Management System	DMS
Blog	B

Table 13
Socio-cultural features.

Feature	Value
Profile	P
Social Network	SN
Social Dependency Net	SDN

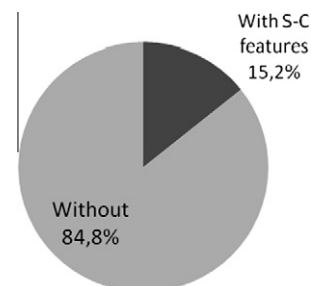


Fig. 8. Distribution by socio-cultural features.

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 features are included in each one. For example, for VMT, you could select *Webex*, *TeamSpace* and *Yahoo Messenger* as possible options, since they support the majority of the features presented in the table (i.e., they support audio, video and chat communication – values A, V, C in the table in the communication feature). Moreover, the tool shows when users start a new session, and/or it also displays 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 session 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 Management (SEMT) which include the maximum number of coordination and control features. This person could therefore select *ActiveCollab*, *Assembla* or *Rational Team Concert*, because they appear to be the most comprehensive tools that include communication features, such as Chat (C) or Forums (F). Moreover, they offer Version Control Systems (VCSs), Build Management Systems (BMSs) and Issue Tracking Systems (ITSs) as control and coordination mechanisms 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-cultural aspects focus mainly on studying social dependencies in social networks (SDNs), such as *Tesseract*, *CASOS* or *Ariadne*.

3.2.1. 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 commonly used by the studied tools in each knowledge area. Moreover, the lists of tools that can be used in each area are also provided.

3.2.1.1. 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 complemented with visual awareness techniques (value V in the Changes Awareness column), such as highlighting important aspects 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.

3.2.1.2. 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 distributed in this case), the majority of the tools use both visual awareness (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 Design Tools is: Artisan Studio, CAB, CAMEL, CoDesign, Creately, Gliffy, GroupUML, Libra-on-Chat, Rational Software Modeler, STEVE, Sysiphus and Together.

3.2.1.3. Construction Tools (SCTs). In the case of the construction tools, something similar to the Design Tools occurs, the most frequently provided feature being the awareness feature (session and changes awareness). However, the construction tools also usually 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, GitHub, Google Code, ICI, Moomba, SCI, Share, Syde, TagSEA and TUKAN.

3.2.1.4. 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, TestLink, Watir and WebTest.

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

3.2.1.6. 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.

3.2.1.7. 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 therefore mainly related to this type of tools. The main feature that this

Table 14
Tool classification and description.

	PS	Area	License	Communication		Awareness		Control & Coordination	KM	S-C
				Sync	Async	Presence	Changes			
4EverEdit	44	KMT	R	-	Cmt	-	V	VCS, ITS	DMS	-
ActiveCollab http://www.activecollab.com/	36	SEMT	C	-	F	S	EN	ITS, VCS	DMS	P
ADAMS	16	SEMT	R	-	-	-	EN	VCS, ITS	-	-
ADDSS	58	KMT	R	-	-	-	-	-	DMS	-
AISA	38	SQT	R	-	Cmt	-	-	-	DMS	-
ARENA	54	SRT	R	-	Cmt	-	-	ITS	-	-
Ariadne	60	S-CT	R	-	-	-	V	-	-	SDN
Artisan Studio http://www.artisansoftwaretools.com/products/artisan-studio	36	SDT	C	-	-	-	-	-	DMS	-
Assembla http://www.assembla.com/	18	SEMT	C	C	F	-	EN	ITS, VCS, BMS	W	P
Augur	26	SEMT	R	-	-	-	V	-	-	-
BSCW http://public.bscw.de/	42	KMT	F	-	Cmt	-	EN, V	VCS	B	-
Bugzilla http://www.bugzilla.org/	36	SEMT	F	-	-	-	-	ITS	-	-
byteMyCode http://www.bytemycode.com/	63	SCT	F	-	Cmt	-	-	-	B	P
CAB	31	SDT	R	-	Cmt	S	V	VCS	-	-
CAMEL http://code.google.com/p/cameldesigntool/	20	SDT	R	C	-	S	V	-	-	-
CASI	53	SCMT	R	-	-	-	V	-	-	-
CASOS	19	SC-T	R	-	-	-	V	-	-	SDN
CAWS	42	KMT	R	-	-	S	V	-	DMS, W, B	-
CheckStyle http://checkstyle.sourceforge.net/	43	SCT	F	-	-	-	-	-	-	-
Citrix GoToMeeting https://www2.gotomeeting.com/	59	VMT	C	C, A	-	S	V	-	-	-
CoDesign	8	SDT	R	-	Cmt	S	V	-	-	-
CollabDev	51	KMT	R	-	-	-	-	VCS	W, DMS	-
CollabVS	52	SCT	C	C, V, A	Cmt	S	V	ITS	-	-
ConnectNow https://www.adobe.com/acom/connectnow/	23	VMT	C	C, V, A	Cmt	S	V	-	DMS	-
Consensus@nyWARE	48	VMT	R	-	Cmt	-	V	ITS	-	-
COPPER	29	SCT	R	C	-	S	V	-	DMS	-
CodeBeamer http://www.intland.com/products/cb/overview.html	13	SEMT	C	-	F	-	EN	ITS, VCS, BMS	W, DMS	-
Creately http://creately.com/	36	SDT	C	-	Cmt	-	-	VCS	W	-
CruiseControl http://cruisecontrol.sourceforge.net/	36	SEMT	F	C	-	-	-	BMS	-	-
CVE http://cve.sourceforge.net/	9	VMT	F	C, A	-	-	-	-	-	-
Darcs http://darcs.net/	36	SCMT	F	-	-	-	-	VCS	-	-
DOCTOR	34	KMT	C	-	Cmt	-	EN	ITS, VCS	DMS	-
DOORS http://www01.ibm.com/software/awdtools/doors/	15	SRT	C	-	Cmt	-	-	ITS	-	-
DrProject https://www.drproject.org/	11	SEMT	F	-	Cmt	-	EN, V	VCS, ITS	DMS	-
eConference	17	VMT	R	C	-	S	V	-	-	-
EGRET	55	SRT	R	-	F	-	V	ITS	-	-
eRequirements http://www.erequirements.com/app	64	SRT	F	-	Cmt	-	V	ITS	DMS	-
FASTDash	12	SCMT	R	-	Cmt	S	V	ITS	-	-
"Fonseca Tool"	25	SEMT	R	-	-	-	EN	-	-	SN
FriendFeed http://friendfeed.com/	18	S-CT	F	-	-	-	-	-	-	SN
Galaxy Wiki	65	KMT	R	-	F	-	-	ITS	W, DMS, B	-
GatherSpace http://www.gatherspace.com/index.html	64	SRT	C	-	Cmt	-	V	ITS, VCS	DMS	-
GENESIS http://www.genesisist.org/english/default.htm	7	SEPT	F	C	F	-	EN	-	-	-
GForge http://gforge.org/gf/	36	SCT	F	-	F	-	EN	ITS, VCS, BMS	DMS	-
Git http://git-scm.com/	36	SCMT	F	-	-	-	-	VCS	-	-
Github http://github.com/	18	SCT	F	-	Cmt	-	-	ITS, VCS, BMS	B	SN
Gliffy http://www.gliffy.com/	36	SDT	C	-	-	-	-	ITS, VCS	W, DMS	-
Google Code http://code.google.com	18	SCT	F	-	-	-	-	ITS, VCS, BMS	DMS, W, B	SN
Google Docs docs.google.com	45	KMT	F	C	Cmt	-	V	VCS	B	-
Google Groups http://groups.google.com	57	KMT	F	-	F	-	EN	VCS	-	-
Google Wave http://wave.google.com/	36	VMT	F	C	F	-	-	-	DMS	P
GroupUML	14	SDT	R	C	-	S	V	-	-	-
Hobbes	32	SEPT	R	-	Cmt	S	V	VCS	DMS	-
HP Quality Center https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24_4000_100__	43	SQT	C	-	-	-	-	-	-	-
HttpUnit http://httpunit.sourceforge.net/	4	STT	F	-	-	-	-	-	-	-
Hypercode	15	SQT	R	-	Cmt	-	EN	-	-	-
IBIS	37	SQT	R	-	Cmt	S	EN	-	-	-
iBistro	5	KMT	R	C, V, A	Cmt	S	V	VCS	DMS	-
ICI	9	SCT	R	C, A	-	-	V	-	-	-
IssuePlayer	27	SEMT	R	-	-	S	V	-	-	-

(continued on next page)

Table 14 (continued)

	PS	Area	License	Communication		Awareness		Control & Coordination	KM	S-C
				Sync	Async	Presence	Changes			
Jira http://www.atlassian.com/software/jira/	36	SEMT	F	–	Cmt	–	EN	VCS, ITS	–	–
JWebUnit http://jwebunit.sourceforge.net/	4	STT	F	–	–	–	–	–	–	–
KNOWFACT	28	KMT	R	–	–	–	–	–	–	–
KnowledgeTree http://www.knowledgetree.com/	36	KMT	C	–	F	–	EN	VCS	DMS	–
Libra-on-chat	66	SDT	R	C	Cmt	S	V	–	–	–
LiveNet * http://linus.socs.uts.edu.au/~igorh/workspace/explore/livenet.htm	2	KMT	F	C	F	S	–	–	–	–
Lotus Sametime http://www01.ibm.com/software/lotus/sametime/	47	VMT	C	C, V, A	–	S, G	–	–	DMS	–
Lotus Quickr http://www01.ibm.com/software/lotus/products/quickr/	28	KMT	C	–	–	–	–	–	DMS	–
Mantis http://www.mantisbt.org/	40	SEMT	F	–	–	–	–	ITS	–	–
MasePlanner	46	SEMT	R	–	Cmt	–	–	–	–	–
Maven http://maven.apache.org/	36	SEMT	F	–	–	–	–	ITS, VCS, BMS	DMS	–
Mercurial http://mercurial.selenic.com/	36	SCMT	F	–	–	–	–	VCS	–	–
Merlin Toolchain http://merlintoolchain.sourceforge.net/index.html	22	MTI	F	–	–	–	–	ITS, VCS, BMS	–	–
Microsoft Office Communicator http://office.microsoft.com/es-es/help/HA102037153082.aspx	47	VMT	C	C, V, A	–	S	–	–	DMS	–
Microsoft Sharepoint http://sharepoint2010.microsoft.com/Pages/default.aspx	43	KMT	C	–	F, Cmt	–	–	–	W, B, DMS	–
MILOS ASE	29	SEMT	R	C, V, A	–	S	V	ITS	–	–
Miramar 2.0	49	VMT	R	C	Cmt	S	V, G	–	–	–
MoinMoin http://moinmo.in/	57	KMT	F	–	F	–	EN	ITS, VCS	W	–
Moomba	29	SCT	R	C, V, A	–	S	V	–	–	P
MPK20 https://lg3dwonderland.dev.java.net/	10	VMT	F	C, A	–	S	–	–	–	–
MSN Messenger messenger.live.com	57	VMT	F	C, V, A	–	S	V	–	–	P
MUDABlue	33	SCMT	R	–	–	–	V	VCS, ITS	–	–
MULTIMIND	28	KMT	R	–	–	–	–	–	–	–
OpenSTA http://opensta.org/	36	STT	F	–	–	–	–	ITS	–	–
P2PConference	38	VMT	R	C	–	S	–	–	–	–
PAKME	3	KMT	R	–	–	–	–	–	DMS	–
Palantír	62	SCMT	R	–	–	–	V	–	–	–
pcAnywhere http://www.symantec.com/business/pcanywhere	59	VMT	C	C	–	S	V	–	–	–
Perforce http://www.perforce.com/	40	SCMT	C	–	–	–	–	VCS, ITS	–	–
Rational Clearcase http://www01.ibm.com/software/awdtools/clearcase/	43	SCMT	C	–	–	–	–	VCS, BMS	–	–
Rational Requirements Composer https://jazz.net/projects/rational-requirements-composer/	63	SRT	C	–	Cmt	–	V	ITS	–	–
Rational Requisite Pro http://www01.ibm.com/software/awdtools/reqpro/	15	SRT	C	–	Cmt	–	EN	ITS	DMS	–
Rational Software Modeler http://www-01.ibm.com/software/awdtools/modeler/swmodeler/	36	SDT	C	–	–	–	–	ITS	–	–
Rational Team Concert https://jazz.net/projects/rational-team-concert/	63	SEMT	C	C	–	S	V	ITS, VCS, BMS	–	–
RepoGuard	40	MTI	R	–	–	–	–	ITS, VCS, BMS	–	–
Saperion ECM http://www.saperion.com/en/products/enterprise-content-management.html	43	KMT	C	–	–	–	–	–	DMS	–
Saros http://www.saros-project.org/	50	SCT	F	C	Cmt	S	V	–	DMS	–
SCARAB http://scarab.tigris.org/	57	SCMT	F	–	–	–	–	ITS	–	–
SCI	9	SCT	R	C, A	–	S	V	–	–	SN
Selenium http://seleniumhq.org/	36	STT	F	–	–	–	–	–	–	–
Share	6	SCT	R	–	Cmt	S	V	VCS	DMS	SN
SoftFab	56	STT	R	–	–	–	–	BMS	–	–
SPEARMINT	24	SEPT	R	–	–	–	–	–	DMS	–
STEVE	21	SDT	R	–	–	–	V	ITS, VCS	–	–
Subversion http://subversion.tigris.org/	36	SCMT	F	–	–	–	–	VCS	–	–
Syde	30	SCT	R	–	–	–	V	ITS, VCS	–	–
Sysiphus http://sysiphus.in.tum.de	15	SDT	R	–	Cmt	–	EN	–	W, DMS	P
TagSEA http://tagsea.sourceforge.net/	61	SCT	F	–	Cmt	–	–	–	–	–
TAMRI	35	SEMT	R	–	–	–	–	–	–	–
TeamSpace http://www.research.ibm.com/teampace/index.html	14	VMT	R	C, V, A	–	S	V	ITS	DMS	–
Tesseract	52	S-CT	R	–	–	–	V	ITS	–	SDN
TestLink http://blog.testlink.org/	36	STT	F	–	–	–	–	VCS	–	–
Together http://www.borland.com/us/products/together/index.html	41	SDT	C	–	Cmt	S	V	VCS, ITS	DMS	–
TortoiseSVN http://tortoisesvn.tigris.org/	57	SCMT	F	–	Cmt	–	–	ITS, VCS	–	–
Trac http://trac.edgewall.org/	65	SEMT	F	–	–	–	–	ITS	DMS, W	–
TUKAN http://www.darmstadt.gmd.de/concert	29	SCT	R	C, A	–	S	–	ITS, VCS	–	–
TWiki http://twiki.org/	65	KMT	F	–	F	–	–	ITS, VCS	DMS, W	–

Table 14 (continued)

	PS	Area	License	Communication		Awareness		Control & Coordination	KM	S-C
				Sync	Async	Presence	Changes			
Twitter http://twitter.com/	23	S-CT	F	–	Cmt	–	V, EN	–	–	SN
Watir http://watir.com/	4	STT	F	–	–	–	–	–	–	–
WebEx http://www.webex.com/	36	VMT	C	C, A, V	–	S	V	–	DMS	–
WebTest http://webtest.canoo.com/webtest/manual/WebTestHome.html	4	STT	F	–	–	–	–	–	–	–
WikiDev 2.0	11	SCMT	R	–	Cmt	S	V	VCS, ITS	W	P
WiP	38	SQT	R	–	–	–	–	–	DMS	–
WiT	38	SQT	R	–	Cmt	–	–	–	DMS	–
Workspace 3D http://www.tixeo.com/en/hosted-video-conferencing/Workspace3D-SE.htm	36	VMT	C	C, A, V	–	S	V	–	DMS	–
Workspace Activity Viewer http://www.cse.unl.edu/~asarma/research.html	1	SEMT	R	–	–	–	V	–	–	–
WorldView http://www.cse.unl.edu/~asarma/research.html	1	SEMT	R	–	–	G	–	–	–	P
XATI	38	SQT	R	–	–	–	–	–	DMS	–
XCHIPS	24	SEPT	R	C, V	–	–	–	–	–	–
Xerox DocuShare http://docushare.xerox.com/	43	KMT	C	–	–	–	–	VCS	DMS	–
XPlanner http://www.xplanner.org/	39	SEMT	F	–	Cmt	–	–	ITS	W	–
Yahoo Messenger http://messenger.yahoo.com/	57	VMT	F	C, V, A	–	S	V	–	–	P

kind of tools should include is Awareness. The majority of them offer 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 Tracking. 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, forums 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.

3.2.1.8. 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.

3.2.1.9. Quality Tools (SQTs). As the number of tools found in this topic is small, it is risky to draw conclusions. However, we can state that the main features are the document manager and the possibility 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.

3.2.1.10. 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.

3.2.1.11. 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.

3.2.1.12. 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 ascertain the state of each user (available, not available, disconnected, etc.). The most comprehensive tools also include a document manager to allow documents to be shared and edited in a virtual meeting. The list of VMT is: Connect Now, Consensus@nywhere, CVE, eConference, Google Wave, Lotus Sametime, MS Office Communicator, Miramar 2.0, MPK 2.0, MSN Messenger, P2P Conference, pcAnywhere, TeamSpace, WebEx, Workspace 3D, Yahoo Messenger.

3.2.1.13. Socio-Cultural Tools (S-CTs). These tools are directly related to social networks and their analysis. Two main features are therefore 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 interactions occur in a work team. This could, for example, help in the making of decisions related to the group structure. The second feature is the management of a social network (SN in the table). This feature is included in well-known tools such as Twitter and is commonly used to keep people close to each other, by sharing personal information. The list of S-CT is: Ariadne, CASOS, Friendfeed, Tesseract and Twitter.

3.2.2. 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 following groups of features:

Awareness: We identified that several types of awareness features 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 monitor 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 information 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 sometimes relatively easy to ignore the work of others in decoupled distributed 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 models 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 social awareness and communication in a collaborative development environment. Other examples are FriendFeed, which is used in [34] to integrate and disseminate personal information into development environments.

Other tools associate social dependencies (“dependency between 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 dependencies 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 incorporated 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 include 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 communication channels.

Knowledge Management: Architectural Knowledge (AK) Management (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, despite 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 CollabDev 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 commonly 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 simplify 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 distributed 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 addresses 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 performed 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 difficulty 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 considered to be a desirable feature.

3.2.3. Tools’ evaluation

Efficient tool selection and evaluation processes are key issues in software engineering if development efficiency is to be increased [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 visualization 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 originally built. In [52], the authors explain how the 4everedit tool was successfully evaluated in a large-scale industrial process engineering project, while in [39], the GENESIS tool was evaluated by two industrial partners of the project. There is, therefore, a certain degree 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 distributed 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 possible 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 appeared 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 evaluation 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.

Table 15
Tools evaluated I.

Type of evaluation	No. of tools	Percentage
External	25	18.9
Internal	9	6.9
None detected	98	74.2
Total number of tools	132	100

Table 16
Tools evaluated II.

Tool	Intended use
<i>Externally evaluated</i>	
SYSIPHUS [32], Together [53]	Distributed Design
ADAMS [33], Augur [29], RepoGuard [49], WikiDev [48]	Configuration Management
WorldView [28], Workspace Activity Viewer [28]	Visualize team structure
IssuePlayer [54]	Project Management
Syde [30], Share [55]	Software Construction
MUDABlue [56]	Categorizing Software Systems
DOCTOR [57], 4everedit [52], Google Docs [40]	Documentation Management
XCHIPS [58], GENESIS [39]	Process Management
iBistro [36]	Knowledge Management
Adobe Connect Now [59], WebEx [40]	Virtual Meetings
CollabDev [45]	Knowledge Management
Tesseract [37]	Socio-technical analysis
EGRET [60]	Requirement Management
LiveNet [42]	Knowledge Management
<i>Internally evaluated/validated</i>	
Ariadne [28]	Socio-Technical analysis
MPK20 [61]	Virtual Meetings
CAB [62], ARENA [63]	Requirements Management
XPlanner [64]	Project Management
Saros [65]	Collaboration Management
CASI [66], SoftFab [6]	Configuration Management
Libra-on-chat [51]	Distributed Design

Another important result is that, as only 39% of the studied research tools presented any kind of evaluation (with only 24 individual tools reporting a level of external validity), there is a need for studies in general to perform tool evaluation as a standard to provide 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.

4. Threats to validity

This section provides a discussion of the validity of the results presented. With regard to its external validity, which implies the possibility of generalizing the findings, this work is limited in two aspects.

The first issue consists of the limitation of the number of tools found. This work presents 132 tools, obtained, on the whole, from research works using the process we explained in Section 2. It is possible, however, that some existing tools have not been included in our study. Moreover, searching for tools through research sources and peer reviewed publications only, may exclude some commercial tools that are available in the market, as not all commercial tools are represented in this kind of work. However, in order to complete this study and provide more complete information about commercial tools, we have created a website (<https://sites.google.com/site/toolsgsd/>) where both commercial and non-commercial tools are described. Apart from the tools obtained in this review, the website also includes those tools that were not obtained by using systematic methods, and were attained by other means such as experts' recommendations, advertisements, web searches, etc. The goal of this website is to maintain an up-to-date tool database. Therefore, each time we discover new tools the website will be updated with new information.

The second limitation to this study relates to the external validity of the tools presented. Some of the primary studies offer only a brief description of these tools, and only where one single tool is presented in a given study is the description of the tool sufficiently thorough. In the worst case, the tools are simply named. All this implies that it is necessary to search for information about the tools on their websites or in similar places. That said, the information provided on these websites is, in some cases, incomplete, and possibly biased since it is often only trade or sales information and not many technical details are shown. In the majority of cases, however, the descriptions of the tools found on their websites have been sufficiently thorough to allow us to obtain the main features of each tool.

With regard to the construct validity, which is related to obtaining 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 level of distribution (those tools that can be used in a highly distributed 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 classification. This was done by 3 researchers with the same background and belonging to the same research group; a slightly different classification might therefore have been obtained if it had been done by other researchers from other groups. The number of results obtained from the searches might also have been different.

5. Conclusions and future work

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 features 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 Engineering 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 communication, 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 project 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*, *Software 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-cultural 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 research 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 social 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 commonly 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 inconsistencies, 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 generated 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 sufficient 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 different 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 particular GSE task(s) each tool supports. To carry out this task, we are currently performing a survey which includes structured interviews 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 information about the features that an integrated framework needs in order 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 (Ministerio de Ciencia e Innovación MICINN, TIN2009-13718-C02-01) and co-funded by FEDER. It is also supported by ENGLOBAS (PII2109-0147-8235) 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 research is also supported, in part, by Science Foundation Ireland Grant 10/CE/11855. 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 systematic review, sorted alphabetically:

List of primary studies in the systematic review

1. 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.
2. 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.
3. 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.
4. 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

5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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
17. 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

18. 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.
19. K.M. Carley, et al., Toward an interoperable dynamic network analysis toolkit, *Decision Support Systems* 43(4) (2007) 1324–1347.
20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. 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.
28. A. Gupta, S. Seshasai, 24-h knowledge factory: using Internet technology to leverage spatial and temporal separations, *ACM Trans. Internet Technol.* 7(3) (2007).
29. B. Hanks, Empirical evaluation of distributed pair programming, *International Journal of Human-Computer Studies* 66(7) (2008) 530–544.
30. 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.
31. 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.

(continued on next page)

Appendix A (continued)

List of primary studies in the systematic review

32. M. Held, W. Blochinger, Structured collaborative workflow design, *Future Generation Computer Systems* 25(6) (2009) 638–653.
33. S. Kawaguchi, et al., MUDABlue: An automatic categorization system for Open Source repositories, *Journal of Systems and Software* 79(7) (2006) 939–953.
34. 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.
35. 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.
36. F. Lanubile, et al., Collaboration Tools for Global Software Engineering, *Software Technology* 27(2) (2010) 52–55.
37. 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.
38. F. Lanubile, T. Mallardo, F. Calefato, Tool support for geographically dispersed inspection teams, *Software Process: Improvement and Practice* 8(4) (2003) 217–231.
39. 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.
40. 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.
41. 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.
42. I. Liccardi, CAWS: improving users' awareness in collaborative authoring activities, in: *Group '07 Doctoral Consortium Papers*, ACM, Sanibel Island, Florida, 2007, pp. 1–2.
43. 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.
44. M. Meisinger, A. Rausch, M. Sihling, 4everedit – team-based process documentation management, *Software Process: Improvement and Practice* 11(6) (2006) 627–642.
45. B. Meyer, Design and code reviews in the age of the internet, *Commun. ACM* 51(9) (2008) 66–71.
46. 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

47. T. Niinimäki, 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.
48. 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.
49. 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.
50. 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.
51. 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.
52. 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.
53. 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.
54. 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)*, Linköping, Sweden, 2005, pp. 87–92.
55. V. Sinha, B. Sengupta, S. Chandra, Enabling Collaboration in Distributed Requirements Management, *IEEE Software*, 2006, pp. 52–61.
56. 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.
57. 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.
58. 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.
59. 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 A (continued)

List of primary studies in the systematic review	
60.	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.
61.	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.
62.	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.
63.	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.
64.	J. Whitehead, Collaboration in software engineering: a roadmap, in: 2007 Future of Software Engineering, IEEE Computer Society, 2007, pp. 214–225.
65.	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.
66.	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.

Appendix B

GSD Tools that have been evaluated.

Tool	Designed to	Evaluation
Ariadne	Highlight the socio-technical relationships between source-code artifacts and the developers implementing those artifacts	Preliminary evaluation of Ariadne’s visualization using inspection methods appropriate for visual interfaces [28]
WorldView	Provide a central repository that can derive, retain and visualize the structure of distributed teams	Work in progress: Evaluation of the tool in real world setting with industrial partners. Controlled experiments conducted in order to assess the usefulness and gather information on how to improve it [28]
WorkSpace Activity Viewer	Visualize the developers and artifacts in a project	Visualize ongoing activities in an opensource project

Appendix B (continued)

Tool	Designed to	Evaluation
	using a 3D metaphor and give managers an overview of ongoing activities in a project	and observe the social aspects of development evolving. Finding: A particular developer started the project, but then become inactive. Later, new developers joined the project and one of them became the new leader on the project and attracted additional developers [28]
LiveNet	Provide different mechanisms for capturing and sharing explicit and tacit knowledge about processes, activities, and artifacts	The workspace networks supported by this tool are particularly applicable to capture and encourage sharing Architectural Knowledge in physically distributed teams [42]
iBistro	Provide meeting spaces for informal collaboration and communication acquiring informal knowledge	Tested and improved in a small distributed software development project at TU München and the National University of Singapore [36]
Share	Provide automatic code sharing, tracking copy and paste, visualizing relationships and allow explicit reference and linking of artifacts	Evaluated through a themed design/programming competition to scaffold the creation of a small scale community of practice providing loose associations and shared interests [55]
GENESIS	Cover the communication and coordination requirements within a software process that are necessary for the planning, execution and coordination of all task-related, spatially and temporally distributed activities	Evaluated by the two main industrial partners of the project, LogicDIS and SchlumbergerSema, acting as pilot users, providing real scenarios for the elicitation of requirements and validation of the platform [39]
MPK20	Provide a virtual 3D environment in which to collaborate by using avatars	Evaluated different kinds of verbal and non-verbal communication, application sharing, and (briefly) its potential for synchronizing

(continued on next page)

Appendix B (continued)

Tool	Designed to	Evaluation
WikiDev R	Provide a wiki as the central platform in which to integrate information about the various artifacts of interest, to cluster this information in clusters of relevant artifacts, and to present views on this information that cut across the individual tool boundaries	activities across time zones [61] Evaluated its effectiveness with a case study [48]
SYSIPHUS	Provide a uniform framework for system models, collaboration artifacts, and organizational models encouraging participants to make communication and issues explicit in the context of system models and become aware of relevant stakeholders	Used since 2000 both in project courses and within consulting projects for industry [32]
ADAMS	Provide an artifact management system, focusing on the support of high level documentation versioning	Experimented on from April 15th to July 20th 2005 in the Software Engineering courses of the Computer Science program at the University of Salerno (Italy). It has been used to coordinate the work of project teams [33]
Adobe Connect Now	Web conferencing software with screen sharing application, interactive whiteboard function, remote control desktop function and text, audio, and video chat options for synchronous communication	A set of students evaluated the tool by using it and rating it. The majority agreed that when performing pair programming as virtual collaboration, Connect Now was easy to use and made it easy to share files. The tool was considered as tolerable as a software solution for online pair programming [59]
XCHIPS	Support collaborative process modeling, where process models are constructed on the basis of a graphical and hypermedia structure	Scenario-based evaluations of the prototype with distributed participants from the application domain in the course of the e-QF project [58]

Appendix B (continued)

Tool	Designed to	Evaluation
Augur	Create visual representations of both software artifacts and software development activities and allow developers to explore the relationship between them	Informal evaluations with developers engaged in active development of multi-authored systems to check the effectiveness [29]
IssuePlayer	Visualize flows of incoming and handled artifacts, along with the people involved (team members) during the lifetime of a software development project	Two independent empirical studies with two industrial partners [54]
Syde	Share change and conflict information across developer's workspaces	The Inspector plug-in was used by an industrial team and the history was used to analyze code ownership. The Scamp plug-in was used by two teams of students to develop their course's project collaboratively and they assessed the usefulness of the plug-in through a qualitative study [30]
CAB	Support early requirements engineering and architecture evaluation	Used as the basis for distributed evaluation walkthroughs with S&R project stakeholders. Finding: It is possible to evolve an early, draft architectural diagram created from a single design stakeholder meeting to a more accurate representation by gathering comments and suggestions made by subject matter experts over time [62]
MUDABlue	Automatically categorize software systems	Experiment with 41 projects to show that the tool is able to make a categorization without any knowledge about target software systems [56]
DOCTOR	Improve the process of distributed software reviews	The tool was used in real projects to check the quality of the review. Findings: It

Appendix B (continued)

Tool	Designed to	Evaluation
		helps to reduce manual work for both the document owner and the reviewer and hence results in significant effort reduction for the review process. It also helps to identify and track a review that is being delayed and take corrective measures earlier [57]
XPlanner	Support project tracking	The tool was successfully used to analyze the team's velocity, which is the amount of effort (in hours) the team made in the previous iteration. The velocity was used to determine the number of features that should be scheduled for the next iteration [64]
RepoGuard	Link version control systems to other software development tools	The tool was used to validate the framework for Subversion and it was also used in many of the German Aerospace Center internal and external development projects [49]
Together	CASE tool supporting UML models	The developers of a company have benefitted greatly from the purchase and use of this tool. The reverse engineering facility was used to a great extent when documentation did not already exist for the code [53]
4everedit	XML based editor with a dynamic user interface, team-support through integration with the versioning system CVS and a mechanism for post-processing the edited documents	Evaluated in a large-scale industrial process engineering project. It was successfully used to support team-based editing, assure structure and consistency, enable post-processing and change document structure. Over a year, 26 editors, from more than five companies, have concurrently

Appendix B (continued)

Tool	Designed to	Evaluation
WebEx	Share screens	elaborated process documentation with 635 pages and 34 MB in size using it [52] Evaluated by using the tool in the development of EiffelStudio, a large integrated development environment (IDE). It was considered especially useful for running a demo of, say, a new proposal for a graphical-user-interface idea or another element [40]
Google Docs	Share documents	This tool was used for collaborative editing in the development of EiffelStudio, a large integrated development environment [40]
Saros	Manage and map means of collaboration in a distributed environment (which are normally performed in a co-located manner)	It was highlighted that this tool can be used in scenarios such as reviews or knowledge transfer [65]
CollabDev	Analyze applications in multiple languages and render various structural, architectural, and functional insights to the people involved in maintenance	The tool was used by 10 team members in two projects. One team used it for their maintenance and stated that “ the tool helped them in ensuring that there were no missing impacts” [45]
Tesseract	Utilize cross-linked displays to visualize the relationships between artifacts, developers, bugs, and communications	The tool was evaluated by demonstrating its feasibility with GNOME project data, assessing its usability via informal user evaluations, and verifying its suitability for the open source community via semi-structured interviews [37]
CASI	Inform developers of the changes that are taking place in a software project and the source code entities influenced by them	It was tested in some small examples and future work includes testing it in more projects of different sizes and in real situations with real

(continued on next page)

Appendix B (continued)

Tool	Designed to	Evaluation
ARENA	Negotiate requirements in a distributed manner	developers [66] 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]
EGRET	Support global software development teams in collaborating on requirements management	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]
SoftFab	Automate building and test processes	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]
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]

References

- [1] 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.
- [2] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using Mapping Studies in Software Engineering, in: PPIG, 2008, pp. 195–204.
- [3] 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.
- [4] K. Dullemond, B.v. Gamenen, Technological Support for distributed agile development, in: Department of Software Technology, Delf University of Technology, Delf, 2009, p. 223.
- [5] 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.
- [6] H. Spanjers, M.t. Huurnec, 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.
- [7] C. Ebert, Global Software Engineering: Distributed Development, Outsourcing, and Supplier Management, Wiley, IEEE Computer Society Books, Los Alamitos, USA, 2010.
- [8] E. Carmel, Global Software Teams: Collaborating Across Borders and Time Zones, Prentice Hall PTR, 1999, 269.
- [9] 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.
- [10] R. Prikładnicki, 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] J. Favela, F. Peña-Mora, An experience in collaborative software engineering education, IEEE Software 18 (2) (2001) 47–53.
- [15] 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.
- [16] F. Lanubile, C. Ebert, C. Prikładnicki, A. Vizcaino, Collaboration tools for global software engineering, IEEE Software 27 (2) (2010) 52–55.
- [17] 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.
- [18] 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.
- [19] F.Q.B.d. Silva, C. Costa, A. Cesar C. França, R. Prikładnicki, 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.
- [20] 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.
- [21] D. Šmite, C. Wohlin, T. Gorschek, R. Feldt, Empirical evidence in global software engineering: a systematic review, Empirical Software Engineering 15 (1) (2010) 91–118.
- [22] 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.
- [23] T.L. Friedman, The World is Flat: Brief History of the 21st Century, Farrar, Straus and Girou, New York, 2005.
- [24] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, in EBSE Technical Report, 2007.
- [25] A. Abran, J.W. Moore, Guide to the software engineering body of knowledge (SWEBOOK®), in: IEEE Computer Society 2004 Guide, 2004.
- [26] A. Fuggetta, A classification of CASE technology, Computer 26 (12) (1993) 25–38.
- [27] 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.

- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.
- [36] 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.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] B. Meyer, Design and code reviews in the age of the internet, Communications of the ACM 51 (9) (2008) 66–71.
- [41] 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.
- [42] 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.
- [43] 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.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [52] M. Meisinger, A. Rausch, M. Sihling, 4everedit – team-based process documentation management, Software Process: Improvement and Practice. 11 (6) (2006) 627–642.
- [53] 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.
- [54] 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.
- [55] 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.
- [56] 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.
- [57] T. Krishnamurthy, S. Subramani, Ailments of distributed document reviews and remedies of DOCTOR (DOCUMENT ORganizer Tool) with distributed reviews support, in: IEEE International Conference on Global Software Engineering (ICGSE 2008), Bangalore, India, 2008, pp. 210–214.
- [58] A. Fernández, B. Garzaldeen, I. Grütznier, J. Münch, Guided support for collaborative modeling, enactment and simulation of software development processes, Software Process: Improvement and Practice. 9 (2) (2004) 95–106.
- [59] 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.
- [60] V. Sinha, B. Sengupta, S. Chandra, Enabling collaboration in distributed requirements management, IEEE Software 23 (5) (2006) 52–61.
- [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.
- [62] 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.
- [63] S. Norbert, Enhancing GSS-based Requirements Negotiation with Distributed and Mobile Tools, 2005.
- [64] 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.
- [65] 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.
- [66] 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.