2010 IEEE International Conference on GLOBAL SOFTWARE ENGINEERING

23-26 August 2010 • Princeton, New Jersey, USA

Sponsored by

IEEE

Published by

IEEE computer society

CPS Conference Publishing Services

IEEE Computer Society Order Number P4122
ISBN-13: 978-0-7695-4122-8
BMS Part # CFP10ICG-PRT
Library of Congress Number 2010927211

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Patrick Kellenberger
Cover art production by Mark Bartosik
Printed in the United Stated of America by Applied Digital Imaging

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

Welcome Message
ICGSE 2010

The 5th International Conference on Global Software Engineering (ICGSE) brings together researchers and practitioners interested in exploring how globally distributed teams work and how the challenges posed by global software engineering can be met. This conference is a forum at the intersection of software engineering, communications, collaboration, business, and cultural aspects that influence human behaviors when faced with software development in global environments. This is the fifth in an annual series of international conferences on Global Software Engineering that began almost five years ago in Florianópolis, Brazil, in 2006. In our subsequent sessions we have learned from different cultures, where global software engineering is thriving: Germany, India, Ireland, and in our fifth edition: North-America.

The research agenda presented at the previous meetings has influenced the research field in global software engineering. We notice more formal modeling and empirical studies, and increased focus on the education of the new software engineers.

This year's technical program is as strong as ever. We received 55 submissions which were thoroughly reviewed by three or four reviewers from an expert program committee. From these 23 were accepted as research papers, 4 as industrial experience papers and 4 as educational papers. The Conference is structured in 9 sessions over 3 days in a single track. Our technical program attempts to address this broad area by presenting new insights into new tools, management, processes, human aspects, and teaching as applied to global software engineering.

The conference provides great opportunities for open discussion of issues and research directions, prompted by three workshops (PARIS'10, Knowing, REMIDI) and one panel. Our panel topic should provide an opportunity for us to travel virtually into the future and to speculate on how the state of the art in communication technologies will likely impact the workplace, specifically with the modern advances in communication technologies.

We offer our sincere thanks to the many individuals and organizations that helped make this year's Conference possible: the IEEE Computer Society, Siemens AG, Siemens Corporate Research, the ICGSE steering committee, the ICGSE 2010 program committee, and the ICGSE 2010 organizing committee. Finally, we would like to thank the authors, the tutorial and keynote speakers, and the workshop organizers and participants for making this year's ICGSE an exciting event.

Welcome to ICGSE 2010 in Princeton and enjoy the Fifth International Conference on Global Software Engineering!

August 2010

Alberto Avritzer, General Chair
Yael Dubinsky, Program Co-chair
Allen Milewski, Program Co-chairs

2010 International Conference on Global Software Engineering

ICGSE 2010

Table of Contents

Tools I: Support and Use

Processes and Practices

Management Environments I

Industry Experiences

Human Aspects

Preparing Software Engineers for Global Software Environments

Management Environments II: Assessing the Risk

Tools II: Emerging Tools

Management Environments III: Understanding Effort in a Global Context

PARIS'10: Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects

REMIDI'10: Fourth International Workshop on Toll Support Development and Management in Distributed Software Projects

KNOWING'10: Knowledge Engineering in Global Software Development

Doctoral Symposium

Tools to Support Global Software Development Processes: A Survey

Javier Portillo-Rodríguez,
Aurora Vizcaíno
Alarcos Research Group
University of Castilla- La Mancha
Ciudad Real, Spain
{javier.portillo,
aurora.vizcaino}@uclm.es

Christof Ebert
Vector Consulting Services
Stuttgart. Germany
christof.ebert@vector.com

Mario Piattini
Alarcos Research Group
University of Castilla- La Mancha
Ciudad Real, Spain
mario.piattini@uclm.es

***Abstract*—The current challenges in Global Software Development (GSD) necessitate support from software tools with special features. In this work, we explain which features are desirable for these tools in the context of GSD and how these features are related to the principal challenges in this environment. We shall therefore present a survey of the tools that provide these features. The tools included in the survey were classified through the use of the ISO/IEC 12207 standard processes to determine which process is supported by each tool. A classification of the tools according to their features is also included, in which the specific features of each tool are shown.**

Keywords-global software development, ISO/IEC 12207, tools, global software tools

I. INTRODUCTION

In recent years, Global Software Development (GSD) has become a growing research area and an expanding trend in the Information Technology (IT) industry [1]. Several features of GSD differ from traditional software development features, principally as a result of the distance factor (temporal, geographical, cultural and linguistic distance) [2]. Therefore, if an appropriate GSD work environment is to be created, software tools (management tools, development tools, etc) must support these special features [3].

A. GSD and its benefits

GSD is an ever-increasing practice in the software development industry ([1], [4]).

Developing software in a distributed setting evolves major important benefits in spite of the problems that this practice might cause [1]. In [1], [4] and [5] the main benefits that a GSD environment offers are detected as being:

- **Product Complexity:** Owing to the increasing complexity of software systems, companies engage into eco-systems of suppliers; distributed development permits complexity to be distributed throughout various organizations [5].
- **Access to a sufficiently large workforce**: Globally distributes development give organizations access to a larger pool of work teams and skills [5].
- **Acquisitions:** Distribution enables companies to acquire opportunities wherever they appear [4].
- **Global Presence:** Companies distributed throughout the world maintain a global presence and visibility [4].
- **Reduction in time-to-market and costs:** Distributing the work in countries with low labor costs and taking advantage of following-the-sun practices can reduce costs and time-to-market respectively [5].
- **Organization Scale:** The centralized management of a large number of development centers is more complex that distributing these centers and managing them from different management units [5].
- **Proximity to market:** The distribution of the organization enables it to be closer to customers [6].

However, in spite of these benefits, this kind of development approach implies the need to consider certain problems and/or challenges, which are outlined in the following section.

B. Challenges of GSD

The benefits of GSD are well known. However, the distance factor has created certain challenges that have led to a growing research area. In this case, the distance factor can be divided into different types of distance which cause different types of challenges (geographical, temporal and socio-cultural distance) [2].

These distance factors, which were first mentioned in [4], have led to the identification of five Problem Areas (PA) in which research efforts might be made. These problem areas are the following:

- **PA1.- Geographic dispersion:** Face-to-face team interactions create a set of advantages that are mostly lost when these teams have to interact in a geographically dispersed manner [4].

- **PA2.- Control and Coordination Breakdown:** Managing the control and coordination in a software project is difficult and this difficulty increases in a GSD environment [4].

978-0-7695-4122-8/10 $26.00 © 2010 IEEE
DOI 10.1109/ICGSE.2010.12

- **PA3.- Loss of communication:** Bearing in mind that face-to-face communication is the richest communication medium, there is a loss of communication richness in a GSD environment which implies problems related to collaboration or trust among team members [4].

- **PA4.- Loss of team spirit:** Teams create an appropriate work environment when there is trust among team members and these team members know each other. In GSD these teams are geographically dispersed, and there are different cultures and a low level of communication. Certain benefits of working in co-located teams are thus lost [4].

- **PA5.- Cultural Differences:** When people from different cultures have to work together it is possible that misunderstandings may occur which can have a negative influence on the team's performance [7].

After considering these problem areas, we have identified some general features that software tools for GSD should include to support the problem areas.

The survey presented in this work thus consists of a set of tools that include the aforementioned features and are considered to be useful tools for GSD.

Moreover, in order to classify the tools we have chosen the ISO/IEC 12207 standard as a classification frame, since it is one of the best known and most frequently used international software lifecycle process standards. In this case, the standard is used to classify each tool into a process, that is, we relate each tool with the ISO/IEC 12207 process that it supports.

The structure of this document is, therefore, as follows: In the next section we describe which features make a tool useful in GSD. In section three we present the ISO/IEC 12207 standard as a classification frame for the studied tools. In section four we list the set of studied tools, classifying them into the ISO/12207 process. Section five shows the table created to describe each tool's features and finally, in section six we present the conclusions obtained.

II. REQUIREMENTS TO GSD TOOLS

This work presents a survey of the tools used to support GSD processes. The starting point for this survey was therefore the definition of which tools would be considered to be GSD tools.

Bearing in mind the problem areas outlined in the previous section, the main idea was to search for software tools that would make it easier to confront these problems. The features considered to be useful in this area were divided into five feature groups (FG): *Online availability, Communication, Awareness, Knowledge Management* and *Control and Coordination.*

A. FG1.- Online/offline availability

A GSD tool can be used by different users on different sites. Users have to share information synchronously or asynchronously but this information should be available at anytime. This implies that one important feature of a GSD tool is its availability to any user, anywhere and anytime.

Different techniques are used to include this feature in a tool. The most common technique is to develop a web-based software tool. Web-based tools (which use web features such as web services) have the property of being available to all users via a simple web browser. This technology not only makes it easier to work simultaneously but is also useful when working asynchronously.

However, an important point that web-based tools may support is the offline availability. It is important to include the possibility of working in an offline mode when, for instance, the server is offline. This way, users are able to synchronize their work once the server is online again.

This feature is important for all the problem areas because often interactions will be "online" interactions, if physical and time distance are balanced.

B. FG2.- Communication

Working in a distributed environment has several implications in communication. In this case, communication between team members hardly ever occurs in a "face-to-face" context and must be supported by software tools.

The most common feature included in most of GSD tools is the *chat feature* which allows two or more members to exchange textual comments. In a synchronous context, there are other kinds of features such as handwriting or videoconference that are also useful in GSD.

In the other hand, if team members do not work at the same time, that is, asynchronously, GSD tools must include other features such as e-mail or forums.

This feature is especially important in tackling the challenges related to PA3.

C. FG3.- Awareness

When working in a distributed team a key factor is knowing which activity is being carried out by the rest of the team. Being aware of "who is doing what" can help team members to avoid work repetition, delays or confusions in general. It also helps to ensure team coordination and team control.

As was previously mentioned, teams that work in a GSD environment can work synchronously or otherwise. If the team works synchronously, GSD tools must include awareness features such as *color identification*, that is, each team member is identified by a color and user actions in the shared workspace are related to the user color. For

instance, if a team member (identified by the color blue) edits a piece of code, the text introduced is set in blue.

However, if team members work asynchronously, awareness features become more important because team members do not know which activities are being carried out by the other team members. In some cases, there are tools that include a *notification feature* with which to notify team members of important events by e-mail. In fact, *e-mail notifications* are a commonly included awareness feature in the GSD tools studied.

Another important awareness information type, included in many GSD tools, is *presence awareness*. This kind of awareness information consists of showing information concerning who is working in the current session (by showing user photos or a name list). Information about where each member is physically located is also included in this type of awareness information.

This feature group is useful to support the features outlined in PA2 in the previous section because being aware of who is doing what helps control and coordination.

D. FG4.- Knowledge Management

Distributed teams, as with other kinds of teams, use and generate an important amount of knowledge in their daily work. However, if the team is distributed, the knowledge is also distributed and cannot be shared or used without the *knowledge management features* (KM features) included in GSD tools.

In this respect, the GSD tools studied tend to implement certain knowledge management features such as wikis which team members can consult and add knowledge to. Another common KM feature is that of including forums in which member exchange knowledge. Some tools include *brainstorming support* to simulate knowledge exchange in "face-to-face" conversations.

From another point of view, in a global environment the offer of socio-cultural knowledge such as personal or cultural knowledge about partners will help to reduce the problems mentioned in PA4 and PA5.

E. FG5.- Control and Coordination

One of the challenges outlined in GSD is coordination and control breakdown (PA2). Tools for GSD must include features to support distributed coordination and control. In this case, it is common to include *tracking features* in the GSD tools. These tracking features allow a track list of important events or facts to be maintained, which can be consulted by team members in order to discover the actual situation when they start to work.

There are many tracking types which are used according to what it is important to track in each case. For instance, there is *bug tracking* which is important when working on software inspection or testing, there is *task tracking* when working in the project management area, etc.

The following section shows a classification of a set of tools that include at least one of these feature groups.

III. TOOLS CLASIFICATION FRAME

In order to make a classification of the tools found, we have chosen the ISO/IEC 12207 as our classification frame.

A. ISO/IEC 12207processes

ISO/IEC 12207 was created to define taxonomy for software lifecycle processes (SLPs). The aim of this standard is to be flexible, modulated and adaptable. These features allow software constructors to adapt this standard to their necessities.

The SLPs, defined in this standard, are divided into tasks and activities and are grouped into *primary processes*, *supporting processes* and *organizational processes*. They can be also grouped into *software specific processes* (SSPs) and *system context processes* (SCPs). This last group classification is used in this work.

SSPs are also divided into three groups (shown in Figure 1):

- *Software Implementation Processes*: These processes are defined to produce specific system elements implemented in software.

- *Software Support Processes*: These processes are designed to provide activities to support *Software Implementation Processes* by defining processes such as *documentation management process* or *configuration management process,* among others.

- *Software Reuse Processes*: These processes are designed to support an organization's ability to reuse software items across project boundaries.

SCPs are divided into four groups of processes (not shown in Figure 1 due to space limitations):

- *Agreement Processes:* These processes include operational activities to establish and maintain collaboration and agreement between two organizations.

- *Project Processes:* Include processes concerned with planning, assessment and control that can be applied in the area of organization management.

- *Technical Processes:* These processes include activities ranging from the definition of system requirements to the disposal of the product when it is withdrawn from service.

- *Organizational Support Processes:* These types of processes have been designed to manage the capability of acquiring and supplying products or services.

Bearing all these processes in mind, in the following section, we show which software tools support certain processes of the ISO/IEC 12207.

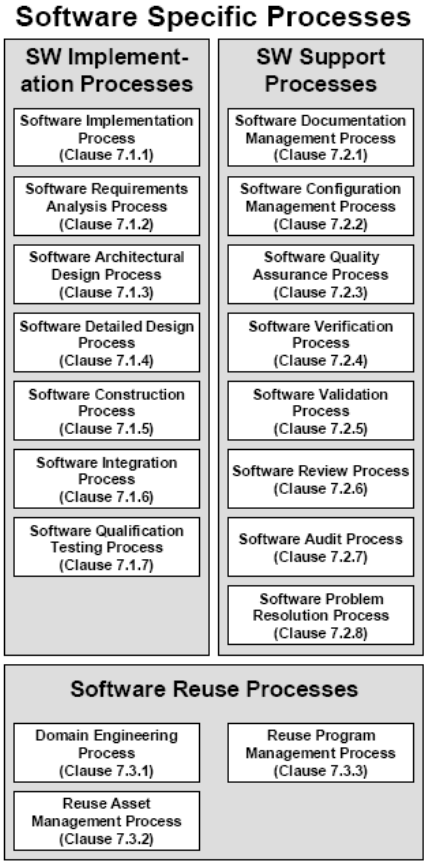

Figure 1. ISO 12207 Software Processes

IV. GSD TOOLS SUPPORTING ISO 12207 PROCESSES

In this section, we show a set of GSD tools mapped to the ISO/IEC 12207 processes that they support. This set includes those processes which are supported by existing specific software tools. We have not include specific tools of *Software Implementation Process* (see Figure1) because we think that is a very general process and the tools that could be used in this process are included in other more specific processes.

Moreover, from the systematic review we obtained that the majority of the tools were research tools, that is, tools that have been developed in research groups/labs. In order to improve our study, we performed a survey asking four multinational companies which work in GSD frequently, , about what tools they often utilize and in what tasks.

Finally, in order to select the set of tools presented in this paper we have chosen those tools including features (described in Section V) helping to reduce the different distances and the problems derived. Basically, all the selected tools include web-based or client-server access and communication control and coordination features for globally distributed teams.

Thus, we have divided the studied tools into different groups.

The first group is composed of tools to support Project Processes (see Figure 1). In this case, these tools principally support project management activities. These tools specifically integrate features to support the *Project Planning Process* and *Project Assessment and Control Process* of the ISO/IEC 12207.

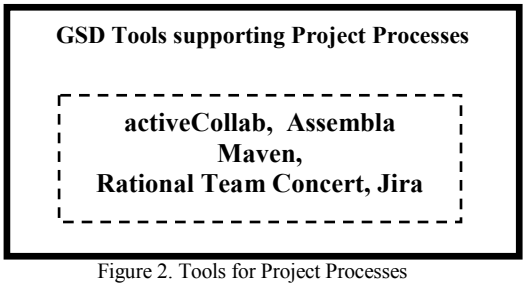

Figure 2. Tools for Project Processes

The following software tools have been included in this first group:

- **activeCollab** [8]: *activeCollab* is a commercial software tool that supports activities related to project management such as planning or progress tracking from a web-based platform. It offers events e-mail notifications and allows e-mail integration into project's workflow.
- **Assembla** [9]: This is a web-based tool which permits the activity stream of a project to be controlled and coordinated. It permits collaboration among team members through the use of web-based features such as wiki pages and the ability to view each team member's work. Assembla allows integration of different collaboration tools depending on the necessities of the project team to perform project activities. It also allows sharing selected project information with clients.
- **Maven** [10]: This tool provides a unified build system and quality project information. It is a free tool that can be integrated into IDEs such as *Eclipse* or *Netbeans*. It has been developed to help JAVA developers maintaining build processes and it allows the creation and integration of plug-ins in order to add new functionalities.
- **Rational Team Concert** [11]: This is an IBM *Jazz* tool that provides planning, source code management, work item management, building management, and project health. To carry out these tasks this tool integrates modules to support process and team awareness, work item tracking, agile planning providing tools to create plans for teams and continuous build integration.
- **Jira** [12]: This is a tool with a high level of integration through plug-ins which offers issue tracking and agile project management. It allows

easy bugs tracking and it offers full integration with popular IDEs such as Eclipse.

The second group is composed of tools supporting implementation processes (see Figure 3). The processes included in this group are *SW Requirements Analysis Process*, *SW Architectural Design Process*, *SW Detailed Design Process, SW Construction Process* and *SW Integration Process.*

SW Requirements Analysis tools: This group includes tools that permit distributed team members to manage software requirements. These tools are the following:

- **IBM Rational DOORS** [13]: This IBM tool permits collaboration among team members, thus allowing them to manage requirements, and can be integrated with other tools to provide a comprehensive traceability solution. It supports the Requirements Interchange Format, which enables suppliers and development partners to be directly involved in the development process. It provides full traceability through the entire lifecycle by the integration of other tools such as *IBM Rational Tau*.
- **eRequirements** [14]: This is a non commercial web-based tool that allows functional and non-functional requirements to be managed. It includes a PDF document generator. It allows capturing project's high level details or collaboratively exploring and building up the Use Cases that will define the system. It also includes an automatic requirement specification document generator.
- **Rational Requirements Composer** [16]: This is another IBM *Jazz* tool which allows the collaborative creation of requirements through web-based interface. It also permits comments to be inserted for communication purposes. Users are able to link requirements with work items in *Rational Team Concert* and tests in *Rational Quality Manager*. It supports collections, shared filters, tags, attributes, hyperlinks and advanced searches to help finding and organizing requirements and related information.

SW Architectural Design Process tools: This group includes web-based tools that support architectural design aspects. These tools are the following:

- **ADDSS:** *ADDSS* from *Architecture Design Decision Support System* is a web-based tool that provides traceability between requirements and design decisions [17]. This tool is able to visualize the evolution of the architecture over time.
- **PAKME:** This is also a web-based tool which uses a data model to characterize architectural constructs (such as design decisions, alternatives, rationale, and quality attributes) [17].

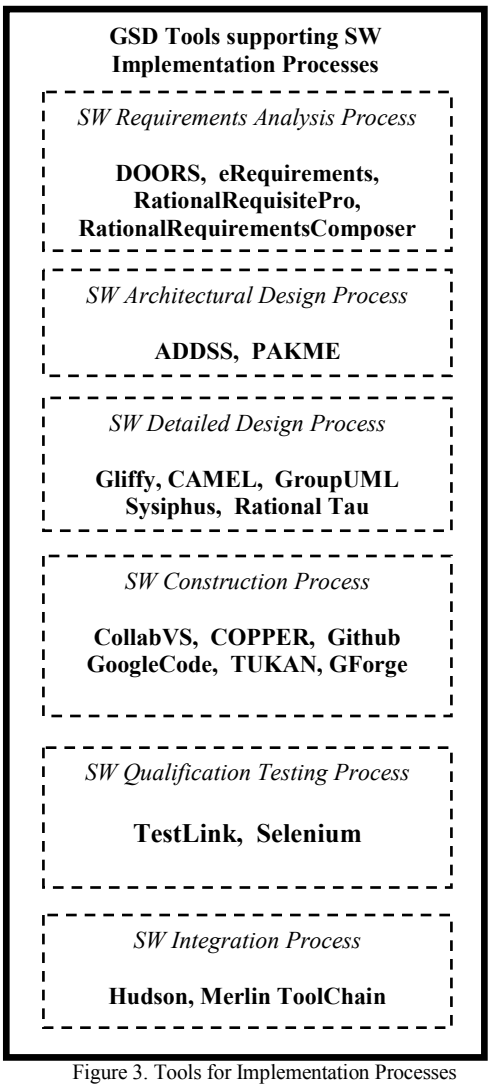

Figure 3. Tools for Implementation Processes

SW Detailed Design Process tools: This group includes tools that allow design activities to be performed collaboratively in a distributed team. These tools are the following:

- **Gliffy** [18]: This tools permits collaboration among distributed team members. It is possible to invite team members to a session in order to view or edit diagrams. It includes e-mail notifications, a revision control system and access control system. It also includes a document manager to organize diagrams allowing tagging.
- **CAMEL** [19]: This research tool provides an environment in which multiple diagrams (UML and free-style) can be sketched and discussed. It also supports mechanisms with which to manage the focus of the dialogue during meetings and provides mechanisms for capturing all the relevant information arising from design meetings.
- **GroupUML** [20]: This research tool supports distributed modeling with UML. It provides a

shared UML editor which is accessible from a Smart board.

- **Sysiphus** [21]: This research tool provides an environment for distributed modeling and collaboration. It also provides traceability and awareness mechanisms through which to facilitate collaboration.
- **Rational Tau** [22]: This tool is a UML 2.1-based environment that permits collaboration among teams, and includes automatic document generation and test case management. It integrates requirements modeling and embedded application development.

SW Construction Process tools: This group is composed of tools that insert collaborative aspects into the software construction process. These tools are the following:

- **CollabVS** [23]: This is a Microsoft research tool which introduces collaborative aspects into Visual Studio to support distributed software development. It includes communication features such as chat and information about online partner and which part of the code is editing everyone.
- **COPPER** [24]: COPPER **(**COllaborative Pair Programing EditoR) is a system to support distributed pair programming that includes features such as viewing which piece of code a partner is editing or independent scrolling.
- **Github** [25]: This tool includes a distributed source code repository in which partners can use the source code browser and insert comments. It provides pre-rolled post-commit hooks (IRC, Jabber, Email, Trac, Campfire, etc.).
- **Google Code** [26]: This web tool is mainly a central code repository which includes developer resources such as APIs or coding toolkits.
- **TUKAN** [24]**:** This tool allows distributed extreme programming. It includes a collaborative editor with separate cursors and insertion points, thus permitting the simultaneous edition of a file.
- **GForge** [27]: This tool is a very complete tool including collaborative source code management supported by trackers, task managers, document managers and forums, among other features. It allows creating and editing trackers, editing roles and observers to set access levels, creating tasks by using the *Task Manager* or linking changes to defects.

SW Qualification Testing Process tools: This group includes tools that support distributed test management. These tools are the following:

- **TestLink** [28]: This is a web-based tool that allows test case and tracking test results to be managed. It also generates test reports.
- **Selenium** [29]: Selenium is a web application testing system that allows test cases to be written and run in a web-browser by using add-ons. It also permits tests to be distributed across multiple servers.

SW Integration Process tools: The standard defines the main activity of this process as being the combination of the software units and software components, thus producing integrated software items. Moreover, this group includes tools that make easier the integration of changes to the project. We have therefore included the following tools:

- **Hudson** [30]: Is a web-based tool that can be considered as a continuous integration system and is mainly focused on building/testing software projects continuously and monitoring executions of externally-run jobs.
- **Merlin ToolChain** [31]*:* This tool offers a free solution to the integration of Project Management, Requirement Management, Configuration Management and Test Management tools .

The third group is composed of software tools that support some of the *SW Support Processes* (see Figure 4). In this group we have specifically included tools to support *SW Documentation Management* process and *SW Configuration Management* process.

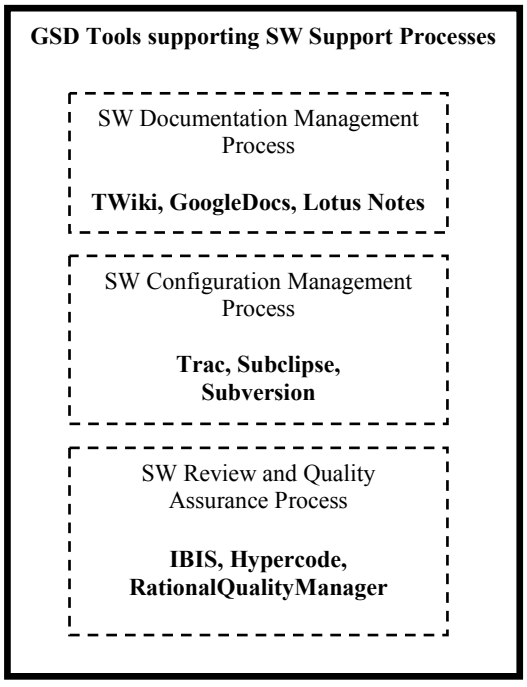

Figure 4. Tools for SW Support Processes

SW Documentation Management Process tools: This group includes tools that permit documentation or document management for distributed teams. These tools are the following:

- **TWiki** [32]: TWiki is a web application platform that uses wiki technology and can be

configured as a document management system.

- **GoogleDocs** [33]: This is a web-based platform that allows documents to be created, edited and shared. It also allows documents to be edited simultaneously.
- **Lotus Notes** [34]: Is a client/server collaboration system that allows sharing for instance documents that must be managed by different persons. Moreover, it allows sharing data bases, calendars or agendas and e-mail management.

SW Configuration Management Process: This group includes tools that are able to support the configuration management process by using web-based protocols. Several of the tools presented herein are based on the *Subversion* tool, and the other tools are issue tracking tools:

- **Subversion** [35]: This is a full version control system that permits the use of the Apache web server in order to provide repository-side network service.
- **Trac** [36]**:** This is a web-based wiki and issue tracking system that includes several plug-ins, thus permitting project tracking, team tracking or development tracking among other things. This tool allows wiki markup in issue descriptions and commit messages, creating links and seamless references between bugs, tasks, changesets, files and wiki pages.
- **Subclipse** [37]: This is an Eclipse plug-in to support Subversion features within the Eclipse IDE.

SW Review Process or Quality Assurance tools: This group includes tools related to the review process and the quality assurance process. These tools are software inspections tools (SIT). SIT are related to the review process because software inspections are considered to be a type of peer review for detecting and removing defects in earlier development stages, and SIT are also related to the quality assurance process because software inspections can be considered as the 'best practice' for improving software quality [38]. The SIT selected are the following:

- **IBIS** [38]: *Internet-Based Inspection System* (IBIS) is a web-based tool to support geographically dispersed inspection teams. IBIS was designed to minimize synchronous activities and coordination problems. The tool includes automatic e-mail notifications and forum discussion threads.
- **Hypercode** [39]: This is a web-based code inspection system that allows distributed teams to perform asynchronous code inspections through a web browser. This tool includes e-mail notifications.
- **Rational Quality Manager** [40]: This is a IBM *Jazz* tool which offers a web-based reporting capable of quantifying the impact of project decisions on business objectives for decision makers and quality professionals. It supports lifecycle quality management from requirements to build to test cases and defects.

V. DESCRIPTION OF TOOLS' FEATURES

As was mentioned in the introduction section, we outlined five problem areas derived from the distance factor that exists in global projects. This factor influences the way in which distributed teams perform the development processes (in this case ISO/IEC 12207 processes), leading to the set of problems previously explained. Moreover, the distance factor can be divided into geographical distance, temporal distance and socio-cultural distance.

In this section we summarize each tool's features, classifying them into the distance aspect that they help to reduce. It is important to note that we have considered that a tool includes a feature if the feature is included by the own tool or if the feature is included by adding a plug-in.

This way, tools reducing the different distance factors may reduce the five problem areas explained in the introduction since the source of these problems are the distance factor. We also make a distinction between the type of features, and two types of features are considered: *communication features (Com in Table 1)* and *coordination and control (C & C in Table 1) features*.

For example, we consider that a tool that includes a forum feature helps to reduce temporal distance with regard to communication because it helps to establish communication when team members work asynchronously (in different time zones). However, if team members are only geographically distant the chat feature can help to reduce this distance as regards communication. Some tools such as *activeCollab* include *Instant Messaging Information* (I.M. Info) in order to facilitate communication connections among team members. Moreover, some features help to reduce more than one distance type and will therefore appear in several columns (see Table 1). We have also included comment insertion as a means of reducing the temporal distance effect, considering the same situation of forums. *The session playback* feature (included in CAMEL) is considered to be a means to communicate what happened in a previous session to team members who work in a different time zone, and this feature has thus been considered to reduce temporal distance.

With regard to *Control and Coordination* features, we have considered that Issue Tracking Systems (ITS) and *e-mail notifications* are the most useful features by which to reduce both temporal and geographical distance. In this case, in Table 1, ITS have been included for a tool when the tool includes any kind of ITS or if the tool can integrate an external ITS.

TABLE I. TOOL FEATURES

Tool (License)	**12207 Process**	**Features**					
		Reducing Temporal Distance		**Reducing Geographical Distance**		**Reducing Socio-Cultural Distance**	
		Com	C&C	Com	C&C	Com	C&C
ActiveCollab (C)	Project	Forum, email notification	Project Tracks, email notification	I.M. Info, email notification	Project Track email notification	Profile Info	-
ADDSS (R)	Arch. Design	-	-	Doc. Manager	Doc. Manager	-	-
Assembla (C)	Project	Forum	Issue Tracks, VCS	Chat	Issue Tracks, VCS	Profile Info	-
CAMEL (R)	Detailed Design	Session Playback	VCS	Chat	Roles, Id by Color, VCS	-	-
CollabVS (R)	Construction	Comments	Issue Track	Chat	Issue Track	-	-
COPPER (R)	Construction	-	-	Chat	Doc. Manager	-	-
DOORS (C)	Requirement	Comments	Issue Track	-	Issue Track	-	-
eASEE (C)	Config. Man., Doc Man.	Comments, notification	Issue Track, Doc. Manager, VCS	Doc. Man., Backbone	Issue Track, Doc. / CM Man.	-	-
eRequirements (F)	Requirement	Comments	Issue Track	-	Issue Track, Doc. Manager	-	-
GForge (F)	Construction	Forum	Issue Track, Doc. Manager, VCS	-	Issue Track, Doc. Manager, VCS	-	-
Github (F)	Construction	Comments	-	-	Issue Track, VCS	Social Network	-
Gliffy (C)	Detailed Design	-	Issue Track, Doc. Manager, VCS	-	Issue Track, Doc. Manager, VCS	-	-
GoogleCode (F)	Construction	-	Issue Track, VCS, Doc. Manager	-	Issue Track, VCS, Doc. Manager	Social Network	-
GoogleDocs (F)	Doc. Man.	Comments	VCS	Chat	VCS	-	-
GroupUML (R)	Detailed Design	-	-	Chat	-	-	-
Hudson (C)	Integration	email notification	Issue Track, VCS, email notification	email notification	Issue Track, VCS, email notification	-	-
Hypercode (R)	Review & Q	Comments, email notification	email notification	email notification	email notification	-	-
IBIS (R)	Review & Q	Comments, email notification	email notification	email notification	email notification	-	-
Jira (F)	**Project**	-	Issue Track	-	Issue Track	-	-
Lotus Notes (C)	Doc. Manag.	**Comments**	Doc. Manager	**Chat**	Doc. Manager	-	-
Maven (F)	Project	-	Issue Track, Doc. Manager, VCS	-	Issue Track, Doc. Manager, VCS	-	-
Merlin Toolchain (F)	Integration	-	-	-	-	-	-
PAKME (R)	Arch. Design	-	-	-	Doc. Manager	-	-
Rational Quality Manager (C)	Review & Q	-	Issue Track, Doc. Manager	-	Issue Track, Doc. Manager	-	-
Rational Requirements Composer (C)	Requirement	Comments	Doc. Manager	-	Doc. Manager	-	-
Rational Tau (C)	Detailed Design	-	Doc. Manager	-	Doc. Manager		
Rational Team Concert (C)	Project	-	Issue Track	Chat	Issue Track	-	-
Selenium (F)	Testing	-	Issue Track	-	Issue Track	-	-
Subclipse (F)	Config. Man.	-	VCS	-	VCS	-	-
Subversion (F)	Config. Man.	-	VCS	-	VCS	-	
Sysiphus (R)	Detailed Design	Forum, email notification	Issue Track, email notification	Issue Track, email notification	Issue Track, email notification	-	-
TestLink (F)	Testing						
Trac (F)	Config. Man.	-	Issue Track	-	Issue Tracks	-	-
TUKAN (R)	Construction	-	Issue Track, VCS	Chat	Issue Track, VCS	-	-
TWiki (F)	Doc. Manag.	Forum	Issue Track, VCS, Doc. Manager	-	Issue Track, VCS, Doc. Manager	-	-

Legend:

Arch. Design: Architecture Design **Config. Man.:** Configuration Management **Doc. Manag.:** Documentation Management **Review & Q.:** Review and Quality
VCS: Version Control System **IM Info:** Instant Messaging Information **C:** Commercial **R:** Research **F:** Free

E-mail notifications help coordination and control because each member knows whether a task has been completed and another task can be initiated. E-mail notifications have not been considered to be a communication feature because, in this case, the e-mail notifications considered are those automatically generated by the tool or system and not those exchanged among team members, and we therefore believe that this type of notification is mainly relevant for control and coordination issues. Two types of socio-cultural features have been considered. The first and most common of these is including profile information about the distributed team members in the tool. The second type is including social network integration in the tool in which team members can create and manage work networks.

VI. CONCLUSIONS

In this paper we have presented a set of tools with special features that make them useful in GSD when distributed team members perform development process activities. These tools have been classified with the ISO/IEC 12207 standard by matching each tool with each process that it supports. Of course, many of these tools are able to support more than one process such as Gforge that supports *Construction* process, *Documentation Management* and *Project Management*, however we have include each tool in the process that is best supported. We have also created a table to describe the tools' features, relating each feature to the distance factor that it helps to reduce.

The information shown in the table thus permits certain conclusions to be drawn with regard to how these tools attempt to solve the challenges or problem areas presented. These conclusions are:

- All the tools attempt to reduce the problems caused by geographic distance (PA1) by offering a web-based environment which allows distributed team members at any location to interact. However, offering a web-based interface is not enough since these kind of tool may also offer an offline work mode in order to support collaboration when, for instance, the web server is offline. In this sense, some of the studied tools such as *Jira, Lotus Notes* or *Gliffy* allow working when the server is offline and updating information when the server is online but, other tools such as *activeCollab* or *eRequirements* needs to be online to allow collaboration.
- The tools studied attempt to reduce control and coordination breakdown (PA2), mainly by offering e-mail notifications, issue tracking systems and version control systems. In the case of e-mail notifications, it is important to integrate them into the workflow supported by the GSE tool. For instances, this is the case for activeCollab that integrates e-mail notifications into a project's workflow. However, this is a feature that is not usually supported.
- In order to reduce the loss of communication (PA3) these tools include communication features such as chats or forums for simultaneous synchronous and asynchronous communication. The majority of the studied tools use asynchronous communication features such as threaded forums since distributed team members do not often meet at same time. Only those tools that support synchronous collaborative activities such as *collabVS* or *CAMEL* include communication features such as chats to support synchronous communication.
- In order to reduce Loss of team spirit (PA4) and Cultural Differences (PA5) the tools include profile information and social network integration. However, as is shown in the table, socio-cultural features are only available in a few tools such as GoogleCode or GitHub. This may therefore be an open gap upon which researchers and companies must work in order to develop tools that also help to decrease aspects related to trust and cultural differences.
- Another detected problem is the lack of connection among tools. It is difficult to select a set of tools to support the complete software lifecycle because each tool has been developed without taking the next phase into account and users have to adapt information from a tool to be used in another tool. Only in the case of *Jazz* tools such as *Rational Requirements Composer* this connection exists with other IBM tools. Be aware however that close integration often builds upon proprietary data models which lock in users with one supplier. We thus recommend using Eclipse-based plug-ins or open data back-bones such as eASEE [15] and not embarking on a single vendor-driven tool-integration.

ACKNOWLEGMENTS

This work has been funded by the PEGASO/MAGO project (Ministerio de Ciencia e Innovación MICINN and Fondo Europeo de Desarrollo Regional FEDER, TIN2009-13718-C02-01). It is also supported by the ENGLOBAS (PII2I09-0147-8235) and MELISA (PAC08-0142-3315) projects, Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, in Spain, and also by the FABRUM project (PPT-430000-2008).

REFERENCES

[1] Ågerfalk, P.J., et al. Benefits of Global Software Development: The Known and Unknown. in International Conference on Software Process, ICSP 2008. 2008. Leipzig, Germany: Springer Berlin / Heidelberg, p.1-9.

[2] Casey, V. Leveraging or Exploiting Cultural Difference? in International Conference on Global Software Engineering. 2009. Limerick, Ireland, p.8-17.

[3] Dullemond, K. and B.v. Gameren, Technological Support for distributed agile development, in Department of Software Technology. 2009, Delf University of Technology: Delf. p. 223.

[4] Carmel, E., Global software teams: collaborating across borders and time zones. 1999: Prentice Hall PTR. 269.

[5] Ebert, C.: Global Software Engineering: Distributed Development, Outsourcing, and Supplier Management. Wiley, IEEE Computer Society Books, Los Alamitos, USA, 2010

[6] Shatz, H.J. and A.J. Venables, The geography of international investment. Policy Research Working Paper Series, 2000. 2338(The World Bank).

[7] Herbsleb, J.D., et al. Distance, dependencies and delay in a global collaboration. in ACM conference on Computer supported cooperative work. 2000. New York, NY, USA.: ACM, p. 319-328.

[8] A51. activeCollab. [cited 2010; Available from: http://www.activecollab.com/.

[9] Assembla. Assembla. [cited 2010; Available from: http://www.assembla.com/.

[10] Foundation, A.S., Maven [cited 2010; Available from: http://maven.apache.org/.

[11] IBM. Rational Team Concert [cited 2010; Available from: http://www-01.ibm.com/software/awdtools/rtc/.

[12] Atlassian, Jira [cited 2010; Available from: http://www.atlassian.com/software/jira/.

[13] IBM. Rational DOORS. [cited 2010; Available from: http://www-01.ibm.com/software/awdtools/doors/.

[14] Robinson, S., eRequirements. [cited 2010; Available from: http://www.erequirements.com/app.

[15] eASEE. PLM and data backbone. http://www.vector.com/easee

[16] IBM. Rational Requirements Composer. 2010; Available from: https://jazz.net/projects/rational-requirements-composer/.

[17] Tang, A., et al., A comparative study of architecture knowledge management tools. Journal of Systems and Software. In Press, Corrected Proof.

[18] Glifffy_Inc. Gliffy. 2009; Available from: http://www.gliffy.com/.

[19] Cataldo, M., et al. CAMEL: A tool for Collaborative Distributed Software Design. in International Conference on Global Software Engineering. 2009. Limerick, Ireland, p. 83-92.

[20] Boulila, N., Group Support for Distributed Collaborative Concurrent Software Modeling, in 19th IEEE International Conference on Automated Software Engineering (ASE'04) 2004: Linz, Austria. p. 422-425.

[21] Bruegge, B., A.H. Dutoit, and T. Wolf, Sysiphus: Enabling informal collaboration in global software development, in International Conference on Global Software Engineering (ICGSE'06) 2006: Florianopolis, Brazil p. 139-148.

[22] IBM. Rational Tau. 2010; Available from: http://www-01.ibm.com/software/awdtools/tau/.

[23] Microsoft, CollabVS. [cited 2010; Available from: http://research.microsoft.com/en-us/projects/collabvs/.

[24] Hanks, B., Empirical evaluation of distributed pair programming. International Journal of Human-Computer Studies, 2008. 66(7): p. 530-544.

[25] Inc., G. GitHub. 2010; Available from: http://github.com/.

[26] Google. Google Code. 2010; Available from: http://code.google.com.

[27] GForge. GForge. 2010; Available from: http://gforge.org.

[28] TestLink. TestLink. 2006; Available from: http://testlink.org/.

[29] Selenium. Selenium. 2010; Available from: http://seleniumhq.org/.

[30] Hudson. Hudson. [cited 2010; Available from: http://wiki.hudson-ci.org/display/HUDSON/Home.

[31] SourceForge. Merlin ToolChain. Available from: http://merlintoolchain.sourceforge.net.

[32] TWiki_Inc. TWiki. 2010; Available from: http://twiki.org/.

[33] Google. Google Docs. 2010; Available from: docs.google.com/.

[34] IBM. Lotus Notes. [cited 2010; Available from: http://www-01.ibm.com/software/lotus/products/notes/.

[35] Apache_Software_Foundation. Subversion. Available from: http://subversion.apache.org/.

[36] Software, E., Trac. 2009. Available from: http://trac.edgewall.org/.

[37] CollabNet. Subclipse. 2008; Available from: http://subclipse.tigris.org/.

[38] Lanubile, F., T. Mallardo, and F. Calefato, Tool support for geographically dispersed inspection teams. Software Process: Improvement and Practice, 2003. 8(4): p. 217-231.

[39] Perry, D.E., et al., Reducing inspection interval in large-scale software development. IEEE Trans. Softw. Eng., 2002. 28(7): p. 695-705.

[40] IBM. Rational Quality Manager. 2010; Available from: https://jazz.net/projects/rational-quality-manager/.