Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2009, Article ID 710971, 14 pages
doi:10.1155/2009/710971

Review Article

Challenges and Improvements in Distributed Software Development: A Systematic Review

Miguel Jiménez,[1] Mario Piattini,[2] and Aurora Vizcaíno[2]

[1] *Alhambra-Eidos, Technology Innovation Center, Paseo de la Innovación 1, 02006 Albacete, Spain*
[2] *Alarcos Research Group, Institute of Information Technologies & Systems, Escuela Superior de Informática, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain*

Correspondence should be addressed to Miguel Jiménez, miguel.jimenez@a-e.es

Received 12 November 2008; Accepted 6 March 2009

Recommended by Hossein Saiedian

Distributed Software Development (DSD) has recently evolved, resulting in an increase in the available literature. Organizations now have a tendency to make greater development efforts in more attractive zones. The main advantage of this lies in a greater availability of human resources in decentralized zones at less cost. There are, however, some disadvantages which are caused by the distance that separates the development teams. Coordination and communication become more difficult as the software components are sourced from different places, thus affecting project organization, project control, and product quality. New processes and tools are consequently necessary. This work presents the findings of a systematic review of the literature related to the challenges concerning Distributed Software Development, whose purpose is to identify the solutions and improvements proposed up to the present day.

1. Introduction

Recent years have seen the geographic distribution of software development. The software industry now tends to relocate its production units in decentralized zones in which a skilled workforce is more readily available, thus taking advantage of political and economic factors [1]. The main objective of this is to optimize resources in order to develop higher quality products at a lower cost than that of colocated developments. Software Factories [2] are therefore organizational structures which automate parts of software development by imitating those industrial processes that were originally linked to more traditional sectors such as those of the automobile and aviation industries, decentralize production units, and promote the reusability of architectures, knowledge and components.

Distributed Software Development (DSD) allows team members to be located in various remote sites during the software lifecycle, thus making up a network of distant sub-teams. In some cases, these teams may be members of the same organization; in other cases, collaboration or outsourcing involving different organizations may exist. Traditional face-to-face meetings are, therefore, no longer common, and interaction between members requires the use of technology to facilitate communication and coordination. Although this phenomenon began in the 90s, only during the last ten years has its strategic importance been recognized [3], and related studies are very recent [4].

The distance between the different teams can vary from a few meters (when the teams work in adjacent buildings) to different continents [5]. The situation in which the teams are distributed beyond the limits of a nation is called Global Software Development (GSD) [6]. This kind of scenario is interesting for several reasons, mainly because it enables organizations to abstract themselves from geographical distance, whilst having qualified human resources and minimizing cost [7], thus increasing the market area by producing software for remote clients and obtaining a longer workday by taking advantage of time differences [8]. However, a number of problems [9], caused mainly by distance, time, and cultural differences [10], must be confronted, and these depend largely on the specific characteristics of each organization.

In this context, "offshoring" refers to the transfer of an organizational function to another country, usually one in which human resources are cheaper. We refer to "nearshoring" when jobs are transferred to geographically closer countries, thus avoiding cultural and time differences between members and saving travel and communication costs. Outsourcing is a mean to contract an external organization, independently of its location, rather than developing in-house [11].

The aforementioned development practices have as a common factor both the problems arising from distance that directly affect the processes of communication and coordination, and control activities [12]. In these environments, communication is less fluid than in colocalized development groups, and problems related to coordination, collaboration, or group awareness therefore appear which negatively affect productivity and, consequently, software quality. These factors all influence the way in which software is defined, built, tested, and delivered to customers, thus affecting the corresponding stages of the software life cycle.

In order to mitigate these effects, and with the aim of achieving higher levels of productivity, organizations require new technologies, processes, and methods [13] through improvements related to the software life cycle, project planning, estimations, risks management, quality assurance, infrastructures, team skills, and the division of responsibilities with the aim of supporting collaboration, coordination, and communication among developers [14]. Iterative approaches are commonly used in contrast to traditional waterfall or sequential methods but these become more difficult to use consistently when teams are geographically distributed [15].

The Model Driven Development (MDD) approach is currently emerging in this field, providing reusability, maintainability, interoperability, and adaptability through different languages and platforms, and improving software quality and developers' productivity. Model Driven Architecture (MDA) [16] is the most frequently adopted MDD standard and provides concepts of separation in individual models and transformation techniques.

Reference [17] discusses the main ideas with regard to how MDA can be used within a collaborative environment to assist interenterprise business processes by using tools that are able to take several input models and produce different kinds of outputs. One representative example of the application of this approach is presented in [18] with a proposal for modeling enterprise organization and developing groupware applications under a concrete MDA-based development process, thus improving communication, collaboration, and coordination between distributed actors. Tools such as InterDOC [19] also exists, which serve as an example of the power of the approach to enable the authoring process when interoperability among different collaborative applications is necessary.

This work presents a systematic review of the literature dealing with efforts related to DSD and GSD with the purpose of discovering the aspects upon which researchers have focused until this moment, thus allowing us to analyze the issues and the solutions which have been contributed up to the present through information of a highly scientific and practical value.

The paper is organized as follows. Section 2 describes the systematic review procedure applied and the results obtained. Section 3 presents an analysis of the results presented in the previous section. The issues and solutions found relating to DSD and GSD are explained in Section 4. The main success factors necessary to carry out a distributed development are listed in Section 5. Finally, Section 6 provides some concluding remarks.

2. Systematic Review Procedure

A systematic review of literature [20] permits the identification, evaluation, and interpretation of all the available relevant studies related to a particular research question, topic area or phenomenon, thus providing results of a high scientific value by classifying studies into primary studies and secondary or relevant studies, by means of synthesizing existing work according to a predefined strategy.

This systematic review has been carried out within the context of the FABRUM project, whose main objective is the development of a process with which to manage the relationships between a planning and design center and a software production factory, this work serves as a starting point upon which to focus future research.

We have followed the systematic search procedure provided by Kitchenham [20], and the selection of primary studies method followed in [21].

2.1. Question Formularization. The research question that guided this systematic review was:

What are the initiatives carried out in relation to the improvement of DSD processes?

The keywords that guided the search to answer the research question were: *distributed*, *software*, *development*, *global*, *enterprise*, *organization*, *company*, *team*, *offshore*, *offshoring*, *outsource*, *outsourcing*, *nearshore*, *nearshoring*, *model*, *strategy,* and *technique*.

The ultimate goal of this systematic review consists of identifying the best procedures, models, and strategies employed, and to determine the most important improvement factors for the main problems found. The population will be composed of publications found in the selected sources which apply procedures or strategies related to DSD.

2.2. Sources Selection. The search strings (shown in Table 1) were established by combining the keyword list from the previous section through the logical connectors "AND" and "OR".

The studies were obtained from the following search sources: *Science@Direct* (http://www.sciencedirect.com), *Wiley Interscience* (http://www.interscience.wiley.com), *IEEE Digital Library* (http://www.computer.org), *and ACM Digital Library* (portal.acm.org/dl.cfm). The quality of these sources guarantees the quality of the studies. The basic search chains had to be adapted to the search engines of each source.

Table 1: Basic search strings.

	Basic search strings
1	("*distributed software development*" OR "*global software development*") AND ((*enterprise* OR *organization* OR *company* OR *team*) AND (*offshore* OR *offshoring* OR *outsource* OR *outsourcing* OR *nearshore* OR *nearshoring*)
2	("*distributed software development*" OR "*global software development*") AND (*model* OR *strategy* OR *technique*)

2.3. Studies Selection. The inclusion criteria for determining whether a study should be considered relevant (a potential candidate to become a primary study) were based on analyzing the title, abstract, and keywords from the studies retrieved by the search to determine whether they dealt with DSD as regards being orientated towards process improvement, quality, coordination, collaboration, communication, and related issues that carry out any improvement concerning the subject in question. In some cases it was necessary to read the entire document to determine its relevance.

After analyzing the results of the first iteration of the systematic review, we applied exclusion criteria to obtain the primary studies, excluding those studies which, despite addressing the issue of DSD, did not contribute to any significant improvement method. We also dismissed those studies which focused solely upon social issues, cultural or time differences or focused solely upon free software, although other papers that address these topics in a secondary manner have been taken into consideration.

The search procedure produced 768 initial studies, of which 497 were not repeated. 170 of these were selected as being relevant, and 78 were selected as primary studies (the complete list of primary studies is shown in the appendix. Table 2 shows the distribution of studies found according to the sources used.

2.4. Information Extraction. The process of extracting information from the primary studies followed an inclusion criterion based on obtaining information concerning the key success factors, improvement strategies employed, processes improved and the most important ideas in each study, thus establishing a categorization between objective and subjective results. All articles were categorized by paying close attention to the methodological study followed according to the models presented in [22]; these categorizations are as follows:

(i) case studies,

(ii) literature reviews,

(iii) experiments,

(iv) simulations,

(v) surveys.

The nonexperimental model for studies (which makes a proposal without testing it or performing experiments) was also applied.

Information corresponding to a specific template (including the type of study, methodology employed, affected processes, and a description of the approach) was extracted from each paper selected for analysis, with particular attention being paid to the problems dealt with and the solutions contributed.

3. Trends in Distributed Software Development Research

This section analyzes and discusses the content of the primary studies found in order to extract relevant information.

Figure 1(a) shows that the majority of the primary studies analyzed are case studies and experimental papers. Nonexperimental studies and surveys in which members involved in the development take part in outlining their difficulties have a significant representation.

However, as Figure 1(b) shows, the majority of primary studies are focused upon the business field but studies in the university environment also appear in which groups of students carried out developments in different locations. 38% of the studies did not indicate their field of work or their classification was not applicable owing to the nature of the study, while 6% were from organizations which did not specify their corporate or university environment.

3.1. Publications Tendency. After concentrating on the number of relevant studies found through the systematic search carried out, it can be concluded that the subject of DSD is evidently an area which was not widely studied until a few years ago, and that it is only recently that a greater number of publications have appeared; thus, as Figure 2 shows, 2006 is the year in which by far the greatest number of studies was published, bearing in mind that the data shown for 2008 only reflects the studies found before September of that year.

3.2. Standards Employed. Figure 3 presents the standards addressed by the articles analyzed. Based on the available data, it may be inferred that few studies indicate the use of specific standards. In part, this is attributable to the fact that the vast majority of studies deal with issues such as communication difficulties in which the standard used is not of importance. The standards supported by most primary studies are CMM, CMMI, and ISO 9001; it is common to jointly apply both. The majority of the studies which applied CMM and CMMI employed a maturity level of 2.

3.3. Improved or Analyzed Processes. Taking the primary studies analyzed as a reference, we carried out a classification in terms of processes in the software life cycle to which improvements were proposed or success factors or areas to be improved related to DSD were discussed. Primary studies were classified according to the improved or studied processes, in each case based on the ISO/IEC 12207 standard [23], with the aim of obtaining a vision of the process

Table 2: Distribution of studies found.

		Studies				
Sources	Search date	Found	Not repeated	Relevant	**Primaries**	%
Science@Direct	15/08/2008	175	143	53	**19**	23,8
Wiley InterScience	27/09/2008	30	20	17	**13**	16,3
IEEE Digital Library	17/08/2008	66	49	19	**14**	18,8
ACM Digital Library	16/08/2008	497	355	80	**32**	41,3
Total	—	768	567	170	**78**	100,0

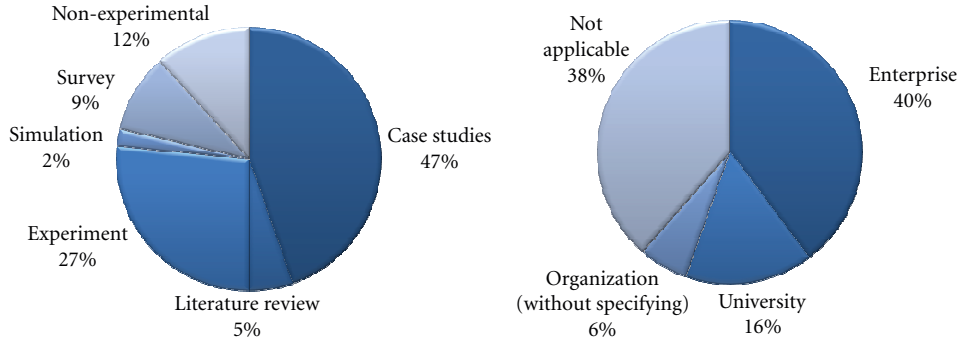

Figure 1: Type of articles analyzed (a), and environments of study development (b).

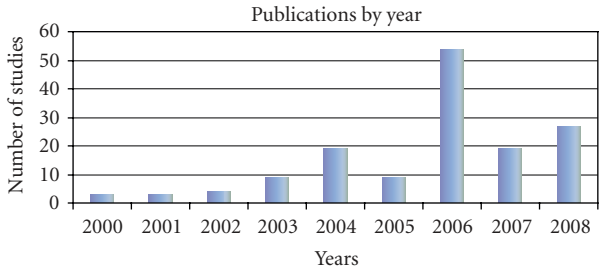

Figure 2: Trends in publications concerning DSD.

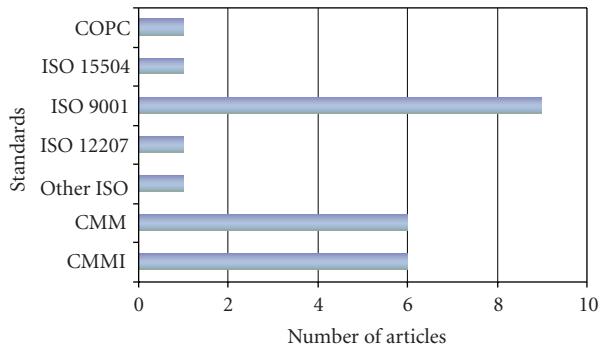

Figure 3: Standards employed in the studies.

life cycle that requires special attention when working in a distributed environment, and discovering the improvement efforts carried out until that moment.

The ISO 12207 standard establishes the activities that may be carried out during the software life cycle, which are grouped into main processes, support processes, and general processes. The results are presented graphically in Figure 4, which indicates frequency in function of the number of studies that address each process.

The results obtained indicate that greater efforts are focused on human resources, organizational management, infrastructure, organizational alignment, and project management. From these data we can infer that communication between team members is a critical factor. Most of the studies are centered on the organizational processes, and we thus believe that there is a need for more studies focused on the level of projects and technical aspects.

3.4. Contents of the Studies. Table 3 provides a schematic representation of the lines towards which the primary studies have focused. Most of the works study tools or models designed specifically for DSD which attempt to improve certain aspects related to development and coordination. Another large part of the studies are related to communication processes and the integration of collaborative tools, combining tools such as e-mail or instant messaging, and studying their application by means of different strategies. Most of the studies address the subject of communication difficulties in at least a secondary manner, presenting this aspect as being one of the most important in relation to the problematic nature of DSD.

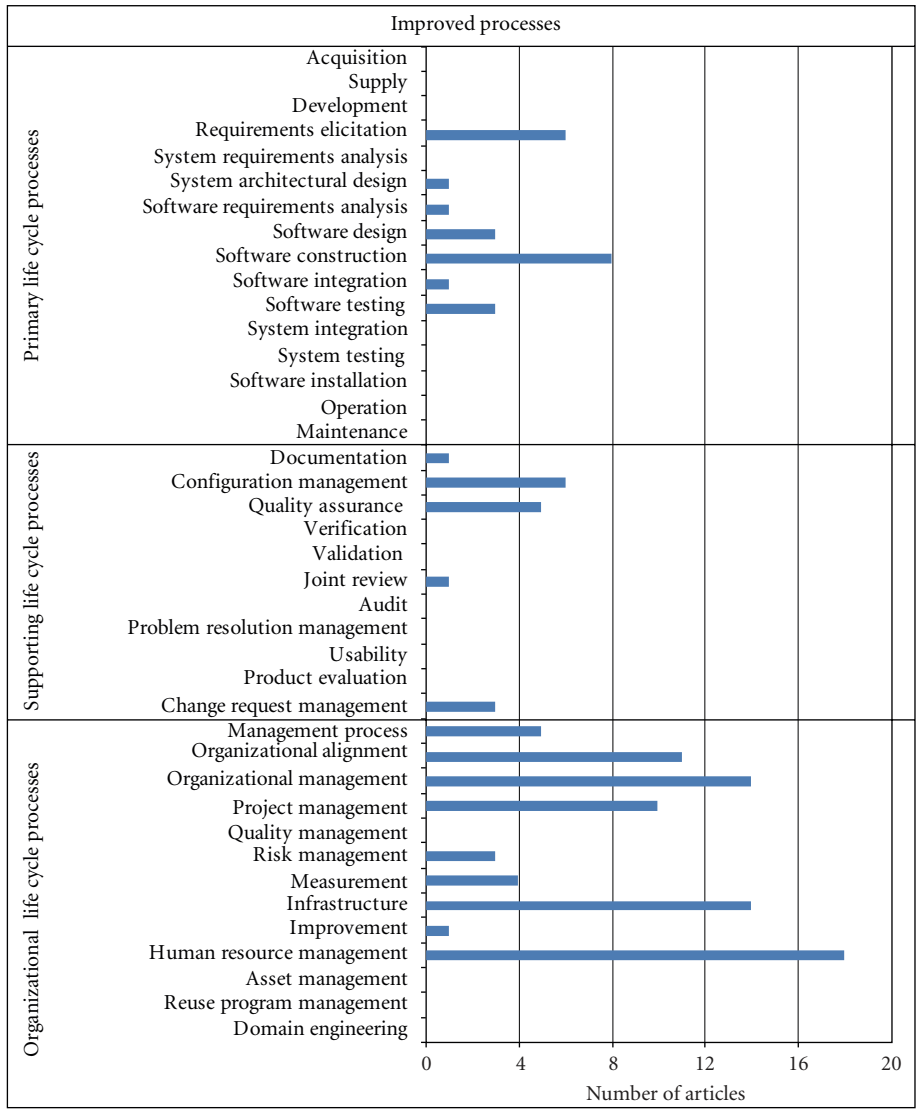

Figure 4: Processes improved or analyzed by the primary studies adjusted to ISO 12207.

Table 3: Thematic areas dealt with in the primary studies.

Thematic areas	Studies (%)
Process control, task scheduling, and project coordination	43.5
Collaborative tools, techniques, and frameworks	35.9
Configuration management	5.4
Multiagent systems	4.3
Knowledge management	7.6
Defects detection	2.2
Test management	1.1

4. Challenges and Improvements

In this section, we synthesize the challenges and proposed improvements identified through the systematic review, discussing the main subjects.

4.1. Communication. The software life cycle requires a great deal of communication between those members involved in the development who exchange a large amount of information through different tools and different formats without following communication standards, and who thus face misunderstandings and high response times. These drawbacks, combined with the complex infrastructure and the great size of personal networks which change over time, are summarized in a decrease in communication frequency and quality, which directly affects productivity. In order to decrease these effects, both methodologies and processes must be supported by collaborative tools, which are a means of avoiding ambiguity and face-to-face meetings without comprising the quality of the results, as is proposed by M. A. Babar et al. [PS56]. K. Mohan and B. Ramesh [PS40] discuss the need for user-friendly tools, and integrate collaborative tools and agents to improve knowledge integration. M. R. Thissen et al. [PS70] examine communication tools and

describe collaboration processes, dealing with techniques such as conference calls and e-mail.

Cultural differences imply different terminologies which may cause mistakes in messages and translation errors. Different levels of understanding of the problem domain also exist, as do different levels of knowledge, skills, and training between teams. The use of translation processes and codification guidelines is therefore useful [PS6].

Requirements should also be clearly defined and modeled in order to make them easily understood, and dependencies among modules should be identified in the architecture. G. N. Aranda et al. [PS34] propose a technique with which to reduce communication problems in the process of requirements elicitation by selecting a suite of groupware tools and techniques from the field of cognitive psychology.

The security of communications must also be taken into account. All the members involved must be able to work with several tools, and the human factor takes on more importance; the team members' communication skills are a critical factor.

4.2. Group Awareness. Members of a virtual team tend to be less productive due to feelings of isolation and indifference. Literature deals with the poor socialization and sociocultural differences which cause a lack of trust [PS39]. Developers need to have as much information as possible at their disposal, and to know the full status of the project and its past history, which will in turn allow them to create realistic assumptions about the project. Frequent changes in processes, lack of continuity in communications, and lack of collaborative tool integration cause remote groups to be unaware of what is important because they do not know what other people are working on. As a consequence, they cannot find the right person and/or timely information which will enable them to work together efficiently, resulting in misalignment, replanning, redesign, and rework.

M. A. D. Storey et al. [PS65] propose a framework for the comparison and understanding of visualization tools that provides awareness of software development activities, giving a solid grounding to the existing theoretical foundation of the field. Augur [PS14] similarly describes a visualization tool which supports DSD processes by creating visual representations of both software artefacts and software development activities, thus allowing developers to explore the relationships between them.

J. D. Herbsleb et al. [PS26] present a tool which provides a visualization of the changing management system, thus making it easy to discover who has experience in working on which parts of the code, and to obtain contact information for that person. In the same line, R. Holmes and R. J. Walker [PS25] present the YooHoo awareness system to help developers to keep apprised of code changes, providing notifications in a flexible manner.

Apart from using these tools, the development process must also be adapted to provide the team members with a better awareness of the project status. It must therefore be automated to provide notifications of actions and decisions to the roles involved.

4.3. Software Configuration Management. Distributed environments present problems derived from conflicts related to source code control. Coordination and synchronization become more complex as the degree of distribution of the team grows, and traceability is a critical factor. Source control systems must support access through Internet, thus confronting its unreliable and insecure nature and the higher response times.

To reduce these drawbacks, S. E. Dossick and G. E. Kaiser [PS11] propose CHIME, an Internet- and Intranet-based application which allows users to be placed in a 3D virtual world representing the software system. Users interact with project artifacts by "walking around" the virtual world, in which they collaborate with other users through a feasible architecture. B. Al-Ani et al. [PS12] present a similar tool which visualizes the developers and artifacts in a project using a 3D metaphor and give managers an overview of ongoing activities in the project. With the same purpose in mind, J. T. Biehl et al. [PS2] present FASTDash, a user-friendly tool that uses a spatial representation of the shared code base which highlights team members' current activities, allowing a developer to rapidly determine which team members have source files checked out, which files are being viewed, and what methods and classes are currently being changed, providing immediate awareness of potential conflict situations, such as two programmers editing the same source file.

B. Bruegge et al. [PS5] present ADAMS, an artefact-based process support system, supporting permissions definition, quality management and storing traceability links between artefacts.

4.4. Knowledge Management. The team members' experiences, methods, decisions, and skills must be accumulated during the development process through effective information-sharing mechanisms, so that each team member can use the experience of his/her predecessor and the experience of the team accumulated during development, thus saving costs and time by avoiding redundant work. Distributed environments must facilitate knowledge sharing by maintaining a product/process repository focused on well-understood functionality by linking content from sources such as e-mail and online discussions, and sharing metadata information among several tools.

To solve the drawbacks caused by distribution, M. A. Babar [PS23] proposes the application of an electronic workspace paradigm to capture and share knowledge to support the software architecture processes.

H. Zhuge [PS76] presents an approach that works with a knowledge repository in which information related to each project is saved by using internet-based communication tools, thus enabling a new team member to become quickly experienced by learning the knowledge stored.

K. Mohan and B. Ramesh [PS40] present an approach based on a traceability framework that identifies the key knowledge elements which are to be integrated, and a prototype system that supports the acquisition, integration, and use of knowledge elements, allowing the knowledge

fragments stored in diverse environments to be integrated and used by various stakeholders in order to facilitate a common understanding.

Change cannot be limited solely to tools, but must also take place in the organization and role distribution. Documentation must always be updated and structured to prevent assumptions and ambiguity, therefore facilitating the maintainability of the software developed.

4.5. Coordination. Coordination in multisite developments becomes more difficult in terms of articulation work, as problems derived from communication, lack of group awareness, and the complexity of the organization appear which influence the way in which the work must be structured and managed [PS3]. J. D. Herbsleb et al. [PS22] suggest that multisite communication and coordination require more people to participate which causes delays. Large changes involve multiple sites and greater implementation times. Changes in multiple distributed sites involve a large number of people. More progress reports, project reviews, conference calls, and regular meetings to take corrective action are therefore needed, thus minimizing task dependencies with other locations. Collaborative tools must support analysis, design and development to permit monitoring activities and managing dependencies, notifications, and implementation of corrective measures. P. Ovaska et al. [PS47] study the coordination of interdependencies between activities, including the figure of a chief architect to coordinate the work and maintain the conceptual integrity of the system.

S. Setamanit et al. [PS59] describe a simulation model to study different ways in which to configure global software development processes. Such models, based on empirical data, allow research into and calculation of the impact of coordination efficiency and its effects on productivity.

C. R. de Souza et al. [PS63] present the Ariadne tool which analyzes software projects for dependencies and helps to find coordination problems through a visual environment.

4.6. Collaboration. Software development is a collaborative activity in which business analysts, customers, system engineers, architects, and developers interact. The concurrent edition of models and processes requires synchronous collaboration between architects and developers who cannot be physically present at a common location. Software modeling requires concurrency control in real time, thus enabling geographically dispersed developers to edit and discuss the same diagrams, and improving productivity by providing a means through which to easily capture and model difficult concepts through virtual workspaces and the collaborative edition of artifacts by means of tools which permit synchronized interactions. S. Liu et al. [PS35] present an interesting approach which can support real-time collaborative UML-based modeling.

B. Bruegge et al. [PS4] describe SYSIPHUS, a distributed environment which provides a uniform framework for system models, collaboration artifacts, and organizational models, with services for exploring, searching, filtering, and analyzing the models.

A further approach is presented by J. Suzuki and Y. Yamamoto [PS16], [PS67] with the SoftDock framework which solves the issues related to software component modeling and their relationships, describing and sharing component models information, and ensuring the integrity of these models. Developers can therefore work by analyzing, designing, and developing software from component models and transfer them by using an exchange format, thus permitting communication between team members. S. Sarkar et al. [PS57] describe CollabDev, a human assisted collaborative knowledge tool with which to analyze applications in multiple languages and render various structural, architectural, and functional insights to the members involved in maintenance.

J. T. Biehl et al. [PS78] present IMPROMPTU, a framework for collaboration in multiple display environments, which allows users to share task information through displays via off-the-shelf applications.

In another direction, X. WenPeng et al. [PS75] study Galaxy Wiki, an online collaborative tool based on the wiki concept which permits the existence of a collaborative authoring system for documentation and coordination purposes, thus allowing developers to compile, execute, and debug programs in wiki pages.

The most valuable characteristics of these kinds of tools for an organization are their simplicity, usability, accessibility, adaptability, and broadband requirements. We therefore believe that proposals based on the wiki concept and Intranet web-based environments are more generic and easier to apply.

4.7. Project and Process Management. High organizational complexity, scheduling, task assignment, and cost estimation become more problematic in distributed environments as a result of volatile requirements, changing specifications, cultural diversity, and the lack of informal communication [PS7]. Managers must control the overall development process, improving it during the enactment and minimizing any factors that may decrease productivity, taking into account the possible impact of diverse cultures, identifying interrelated tasks, and minimizing dependencies among distributed groups.

The maturity of the process becomes a key success factor. M. Passivaara and C. Lassenius [PS48] propose incremental integration and frequent deliveries by following informing and monitoring practices.

H. Spanjers et al. [PS64] present *SoftFab*, an infrastructure which enables projects to automate the building and test process, and which manages all the tasks remotely though a control center.

G. Gousios et al. [PS17] propose a model for evaluating developers' contributions by combining traditional metrics with data mined from software repositories to extract contribution indicators. In the same line, N. Nagappan et al. [PS43] present a metric scheme to quantify organizational complexity.

R. J. Madachy [PS38] deals with economic issues, presenting a set of cost models to estimate distributed teams'

work, and taking into account different environmental characteristics of the teams, localized labor categories, calendars, compensation rates, and currencies for costing.

The automation of the process through an adaptable tool is consequently necessary in order to manage tasks and metrics through customizable reports managed by a central server and ensuring the application of the development processes in compliance with a predefined standard.

4.8. Process Support. Processes should reflect the direct responsibilities and dependencies between tasks, notifying the people involved of the changes that concern them, thus avoiding the information overload of team members. Process modeling and enactment should support the intersite coordination and cooperation of the working teams, offering automated support to distributed project management. Problems derived from process evolution, mobility, and tool integration appear within this context. Process engines have to support changes during enactment. Furthermore, distributed environments usually involve a large network of heterogeneous, autonomous and distributed models, and process engines, which requires the provision of a framework for process system interoperability.

In relation to these problems, A. Fernández et al. [PS13] present the SPEARMINT process modeling environment, which supports extensive capabilities for multiview modeling and analysis, and XCHIPS for Web-based process support which permits enactment and simulation functionalities.

S. Setamanit et al. [PS59] describe a hybrid computer simulation model of software development processes to study alternative ways in which to configure GSD projects in order to confront communication problems, control and coordination problems, process management, and time and cultural differences.

4.9. Quality and Measurement. The quality of products is highly influenced by the quality of the processes that support them. In DSD projects the impact of issues can be magnified when a problem is discovered, and it is more difficult to recover from this than in collocated projects. Organizations should introduce new quality assurance models and measures to obtain information which can be adapted to the distributed scenarios, thus ensuring that the requirements reflect the customer's needs. One of the most frequently recommended practices is that of automated code inspections [PS4] and the application of coding standards. With this aim, K. V. Siakas and B. Balstrup [PS27] propose the capability model eSCM-SP, which has many similarities with other capability-assessment models such as CMMI, Bootstrap or SPICE, and the SQM-CODE model, and considers the factors that influence software quality management systems from a cultural and organizational perspective.

J. D. Herbsleb et al. [PS21] work with several interesting measures, such as the *interdependence measure* which allows the degree of dispersion of work among sites to be determined by looking up the locations of all the individuals. F. Lanubile et al. [PS30] similarly propose metrics associated with products and processes oriented towards software defects such as: discovery effort, reported defects, defects density, fixed defects or unfixed defects.

Furthermore, software architecture evaluation usually involves a large number of stakeholders who need face-to-face evaluation meetings, and adequate collaborative tools are therefore needed, such as that proposed by M. A. Babar et al. [PS56].

We observed a lack of empirical studies that allow us to enumerate reliable measures, and more articles related to tests in distributed environments, which are directly related to software quality, are also necessary.

4.10. Risk Management. Risk management is a critical project management activity. In addition to all the known traditional issues connected with collocated environments [PS7], DSD development includes issues related to coordination, problem resolution, evolving requirements, knowledge, sharing and risk identification [14]. Software defects become more frequent due to the added complexity, and in most cases, this is related to communication problems and a lack of group awareness. Defects control must be adapted by making a greater effort in risk management activities. The use of adequate measures and the requirements definition is important key factors.

In an attempt to minimize these problems, F. Lanubile et al. [PS30] define a process, specifying roles, guidelines, forms and templates, and describe a web-based tool that adopts a re-engineered inspection process in order to minimize synchronous activities and coordination problems and thus support geographically dispersed teams.

R. Kuni and Navneet Bhushan [PS29] propose the WOOM methodology to provide measures and facilitate decision making, taking into account both the risks during various lifecycle phases and mitigation plans.

Rules and guidelines with which to organize the teams and their interactions become necessary. Teams must be continuously controlled in order to detect problems and take corrective actions.

5. Success Factors

From the experimental studies analyzed, we have extracted the following success factors of DSD. The primary studies referenced are listed in the appendix.

(i) Intervention of human resources by participating in surveys [PS56], [PS21].

(ii) Carrying out improvements based on the needs of the company, taking into account the technologies and methodologies used [PS1]. The tools employed at the present must be adapted and integrated [PS58].

(iii) Training of human resources in the tools and processes introduced [PS22].

(iv) Registration of activities with information on pending issues, errors and people in charge [PS2], and the provision of awareness of software development activities [PS65].

Table 4: Primary studies selected in the systematic review.

Number	Year	Source	Type of study	Reference
PS1	2004	Science Direct	Use Case	[24]
PS2	2007	ACM	Use Case	[25]
PS3	2006	ACM	Use Case	[26]
PS4	2006	IEEE	Experimental	[27]
PS5	2006	IEEE	Experimental	[28]
PS6	1998	Science Direct	Literature review	[29]
PS7	2006	IEEE	Use Case	[30]
PS8	2008	ACM	Experimental	[31]
PS9	2007	Science Direct	Use Case	[32]
PS10	2007	Science Direct	Use Case	[33]
PS11	1999	ACM	Nonexperimental	[34]
PS12	2008	ACM	Experimental	[35]
PS13	2004	Wiley Interscience	Use Case	[36]
PS14	2004	ACM	Use Case	[37]
PS15	2008	Wiley Interscience	Use Case	[38]
PS16	1996	Science Direct	Experimental	[39]
PS17	2008	ACM	Experimental	[40]
PS18	2006	ACM	Use Case	[41]
PS19	2008	Science Direct	Experimental	[42]
PS20	2006	IEEE	Experimental	[43]
PS21	2000	ACM	Survey	[44]
PS22	2001	ACM	Survey	[45]
PS23	2008	ACM	Experimental	[46]
PS24	2005	ACM	Survey	[47]
PS25	2008	ACM	Experimental	[48]
PS26	2006	IEEE	Use Case, Survey	[49]
PS27	2006	Wiley Interscience	Nonexperimental	[50]
PS28	2008	Science Direct	Use Case	[51]
PS29	2006	IEEE	Experimental	[52]
PS30	2003	Wiley Interscience	Use Case	[53]
PS31	2006	Science Direct	Use Case	[54]
PS32	2006	ACM	Survey	[55]
PS33	2006	ACM	Survey	[56]
PS34	2006	IEEE	Use Case	[57]
PS35	2006	IEEE	Experimental	[58]
PS36	2002	IEEE	Use Case	[59]
PS37	2008	Wiley Interscience	Survey	[60]
PS38	2008	Wiley Interscience	Nonexperimental	[61]
PS39	2008	Wiley Interscience	Use Case	[62]
PS40	2007	Science Direct	Use Case	[63]
PS41	2004	Science Direct	Use Case	[64]
PS42	2007	Science Direct	Use Case	[65]
PS43	2008	ACM	Use Case	[66]
PS44	1991	Science Direct	Use Case	[67]
PS45	2007	Wiley Interscience	Use Case	[68]
PS46	2008	ACM	Experimental	[69]
PS47	2003	Wiley Interscience	Use Case	[70]
PS48	2003	Wiley Interscience	Use Case, Survey	[71]
PS49	2006	ACM	Use Case	[72]

Table 4: Continued.

Number	Year	Source	Type of study	Reference
PS50	2004	ACM	Literature review	[73]
PS51	2003	Wiley Interscience	Use Case	[74]
PS52	2006	IEEE	Use Case	[75]
PS53	2007	ACM	Use Case	[76]
PS54	2005	Science Direct	Literature review	[77]
PS55	2008	ACM	Experimental	[78]
PS56	2006	Science Direct	Experimental	[79]
PS57	2008	ACM	Experimental	[80]
PS58	2003	ACM	Experimental	[81]
PS59	2007	Wiley Interscience	Simulation	[82]
PS60	2004	ACM	Experimental	[83]
PS61	2006	IEEE	Literature review	[84]
PS62	2006	Wiley Interscience	Use Case	[85]
PS63	2007	ACM	Experimental	[86]
PS64	2006	IEEE	Use Case	[87]
PS65	2005	ACM	Experimental	[88]
PS66	1999	IEEE	Nonexperimental	[89]
PS67	1995	Science Direct	Nonexperimental	[90]
PS68	2006	Science Direct	Experimental	[91]
PS69	2006	Science Direct	Use Case	[92]
PS70	2007	ACM	Use Case	[93]
PS71	2005	IEEE	Nonexperimental	[94]
PS72	2008	ACM	Experimental	[95]
PS73	2004	ACM	Use Case	[96]
PS74	2006	Science Direct	Nonexperimental	[97]
PS75	2007	ACM	Experimental	[98]
PS76	2002	Science Direct	Experimental	[99]
PS77	2006	ACM	Use Case	[100]
PS78	2008	ACM	Use Case	[101]

(v) Establishment of an efficient communication mechanism between the members of the organization, allowing a developer to discover the status and changes made within each project [PS67], [PS2].

(vi) Using a version control tool in order to control conflictive situations [PS49].

(vii) There must be a manner in which to permit the planning and scheduling of distributed tasks, taking into account costs and dependencies between projects, and the application of corrective measures and notifications [PS14], [PS38].

(viii) Application of maturity models and agile methodologies [PS32] based on incremental integration and frequent deliveries.

(ix) Application of MDD approaches to automate development tasks [PS64], [PS72].

(x) Systematic use of metrics tailored to the organization [PS22].

6. Conclusions

In this work we have applied a systematic review method in order to analyze the literature related to the topic of DSD within the FABRUM project context whose main objective is to create a new DSD model with which to manage the relationships between a planning and design center and a software production factory. This work serves as a starting point from which to establish the issues upon which subsequent research will be focused.

The results obtained from this systematic review have allowed us to obtain a global vision of a relatively new topic which should be investigated in detail. However, every organization has concrete needs which basically depend on its distribution characteristics, its activity and the tools it employs. These factors therefore cause this subject to be extremely wide-ranging, and lead to the necessity of adapting both the technical and organizational procedures, according to each organization's specific needs.

The proposals found in the analyzed studies were, in general, mainly concerned with improvements related to the use of collaborative tools, the integration of existing

tools, source code control, or the use of collaborative agents. Moreover, it should be stressed that the evaluation of the results obtained from the proposed improvements are often based on studies in a single organization, and sometimes only takes into account the developers' subjective perception.

On the other hand, it should be noted that maturity models such as CMM, CMMI, or ISO, which would be of particular relevance to the present investigation, represent only 17% of all analyzed works. The fact that almost all the experimental studies that employed CMMI and CMM applied a maturity level of 2 suggests that the cost of implementing higher maturity levels in distributed environments might be too high. However, there is a need for more studies related to the application of maturity models and metrics to quantify issues related to the process areas. The application of agile methodologies based on incremental integration and frequent deliveries, and frequent reviews of problems to adjust the process become important success factors. We also found an increasing interest in modeling in software development, and MDA approaches as a means to improve productivity, quality and understanding among members involved in the development process.

Finally, we must emphasize that the search was reduced to a limited number of search engines and excluded studies which addressed the subject of DSD but did not contribute any significant method or improvement in this research context. However, since this is such a wide area, some of these works present interesting parallel subjects for the development of this investigation, and their study would, therefore, be important in a future work. We also have found studies related to the business perspective or focused on the customer which may be useful for related works. Furthermore, many studies mainly related to tools which are not included in the context of DSD but are useful in fields related to communications or source control also exist.

Appendix

A. Primary Studies Selected

The selected primary studies in the systematic review are presented in Table 4.

Acknowledgments

The authors acknowledge the assistance of MELISA Project (PAC08-0142-3315), financed by the "Junta de Comunidades de Castilla-La Mancha" of Spain. This work is part of FABRUM Project (PPT-430000-2008-63), financed by "Ministerio de Ciencia e Innovación" of Spain and by Alhambra-Eidos (http://www.alhambra-eidos.es/).

References

[1] W. Aspray, F. Mayadas, and M. Y. Vardi, "Globalization and offshoring of software," Report of the ACM Job Migration Task Force, Association for Computing Machinery, New York, NY, USA, 2006.

[2] J. Greenfield, K. Short, S. Cook, S. Kent, and J. Crupi, *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, John Wiley & Sons, New York, NY, USA, 2004.

[3] R. Davison, "Offshoring information technology: sourcing and outsourcing to a global workforce," *Information Technology for Development*, vol. 13, no. 1, pp. 101–102, 2007.

[4] R. Prikladnicki, D. Damian, and J. L. N. Audy, "Patterns of evolution in the practice of distributed software development: quantitative results from a systematic review," in *Proceedings of the 12th Conference on Evaluation and Assessment in Software Engineering (EASE '08)*, Bari, Italy, June 2008.

[5] R. Prikladnicki, J. L. N. Audy, and J. R. Evaristo, "Distributed software development: toward an understanding of the relationship between project team, users and customers," in *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS '03)*, pp. 417–423, Angers, France, April 2003.

[6] J. D. Herbsleb and D. Moitra, "Global software development," *IEEE Software*, vol. 18, no. 2, pp. 16–20, 2001.

[7] W. Kobitzsch, D. Rombach, and R. L. Feldmann, "Outsourcing in India," *IEEE Software*, vol. 18, no. 2, pp. 78–86, 2001.

[8] C. Ebert and P. De Neve, "Surviving global software development," *IEEE Software*, vol. 18, no. 2, pp. 62–69, 2001.

[9] L. Layman, L. Williams, D. Damian, and H. Bures, "Essential communication practices for extreme programming in a global software development team," *Information and Software Technology*, vol. 48, no. 9, pp. 781–794, 2006.

[10] S. Krishna, S. Sahay, and G. Walsham, "Managing cross-cultural issues in global software outsourcing," *Communications of the ACM*, vol. 47, no. 4, pp. 62–66, 2004.

[11] S. McConnell, *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, Redmond, Wash, USA, 1996.

[12] D. Damian, F. Lanubile, and H. L. Oppenheimer, "Addressing the challenges of software industry globalization: the workshop on global software development," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 793–794, Portland, Ore, USA, May 2003.

[13] D. Damian and F. Lanubile, "The 3rd international workshop on global software development," in *Proceedings of the 26th International Conference on Software Engineering (ICSE '04)*, pp. 756–757, Edinburgh, UK, May 2004.

[14] R. Sangwan, M. Bass, N. Mullick, D. J. Paulish, and J. Kazmeier, *Global Software Development Handbook*, Auerbach Series on Applied Software Engineering Series, Auerbach, Boston, Mass, USA, 2006.

[15] M. A. Cusumano, "Managing software development in globally distributed teams," *Communications of the ACM*, vol. 51, no. 2, pp. 15–17, 2008.

[16] OMG, "MDA guide version 1.0.1," Tech. Rep. omg/2003-06-01, Object Management Group, Needham, Mass, USA, June 2003.

[17] L. Kutvonen, "Relating MDA and inter-enterprise collaboration management," in *Proceedings of the 2nd European Workshop on Model Driven Architecture (MDA) with an Emphasis on Methodologies and Transformations (EWMDA '04)*, pp. 84–88, University of Kent, Canterbury, UK, September 2004.

[18] J. L. Garrido, M. Noguera, M. González, M. V. Hurtado, and M. L. Rodríguez, "Definition and use of computation independent models in an MDA-based groupware development process," *Science of Computer Programming*, vol. 66, no. 1, pp. 25–43, 2007.

[19] R. S. P. Maciel, C. G. Ferraz, and N. S. Rosa, "An MDA domain specific architecture to provide interoperability among collaborative environments," in *Proceedings of the 19th Brazilian Symposium on Software Engineering (SBES '05)*, pp. 1–16, Uberlandia, Brazil, October 2005.

[20] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE-2007-001, Keele University & Durham University Joint Report, Staffordshire, UK, 2007.

[21] F. J. Pino, F. García, and M. Piattini, "Software process improvement in small and medium software enterprises: a systematic review," *Software Quality Journal*, vol. 16, no. 2, pp. 237–261, 2008.

[22] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23–31, 1998.

[23] ISO/IEC 12207:2002, "AMENDMENT 1: Information technology—Software life cycle processes," International Organization for Standardization, 2002.

[24] M. Akmanligil and P. C. Palvia, "Strategies for global information systems development," *Information & Management*, vol. 42, no. 1, pp. 45–59, 2004.

[25] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, "FASTDash: a visual dashboard for fostering awareness in software teams," in *Proceedings of the 25th SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pp. 1313–1322, San Jose, Calif, USA, April 2007.

[26] B. Brian, "Impact of organizational structure on distributed requirements engineering processes: lessons learned," in *Proceedings of the International Workshop on Global Software Development for the Practitioner (GSD '06)*, Shanghai, China, May 2006.

[27] B. Bruegge, A. H. Dutoit, and T. Wolf, "Sysiphus: enabling informal collaboration in global software development," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 139–148, Florianopolis, Brazil, October 2006.

[28] B. Bruegge, A. De Lucia, F. Fasano, and G. Tortora, "Supporting distributed software development with fine-grained artefact management," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 213–222, Florianopolis, Brazil, October 2006.

[29] J. M. Carey, "Creating global software: a conspectus and review," *Interacting with Computers*, vol. 9, no. 4, pp. 449–465, 1998.

[30] V. Casey and I. Richardson, "Project management within virtual software teams," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 33–42, Florianopolis, Brazil, October 2006.

[31] V. Clerc, "Towards architectural knowledge management practices for global software development," in *Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge (SHARK '08)*, Leipzig, Germany, May 2008.

[32] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, and J. Howison, "Self-organization of teams for free/libre open source software development," *Information and Software Technology*, vol. 49, no. 6, pp. 564–575, 2007.

[33] A. De Lucia, F. Fasano, G. Scanniello, and G. Tortora, "Enhancing collaborative synchronous UML modelling with fine-grained versioning of software artefacts," *Journal of Visual Languages and Computing*, vol. 18, no. 5, pp. 492–503, 2007.

[34] S. E. Dossick and G. E. Kaiser, "CHIME: a metadata-based distributed software development environment," in *Proceedings of the 7th European Software Engineering Conference, held jointly with the 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, Toulouse, France, September 1999.

[35] B. Al-Ani, E. Trainer, R. Ripley, A. Sarma, A. van der Hoek, and D. Redmiles, "Continuous coordination within the context of cooperative and human aspects of software engineering," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '08)*, Leipzig, Germany, May 2008.

[36] A. Fernández, B. Garzaldeen, I. Grützner, and J. Münch, "Guided support for collaborative modeling, enactment and simulation of software development processes," *Software Process: Improvement and Practice*, vol. 9, no. 2, pp. 95–106, 2004.

[37] J. Froehlich and 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 (ICSE '04)*, vol. 26, pp. 387–396, Edinburgh, UK, May 2004.

[38] P. J. Gomes and N. R. Joglekar, "Linking modularity with problem solving and coordination efforts," *Managerial and Decision Economics*, vol. 29, no. 5, pp. 443–457, 2008.

[39] I. Gorton and S. Motwani, "Issues in co-operative software engineering using globally distributed teams," *Information and Software Technology*, vol. 38, no. 10, pp. 647–655, 1996.

[40] G. Gousios, E. Kalliamvakou, and D. Spinellis, "Measuring developer contribution from software repository data," in *Proceedings of the International Working Conference on Mining Software Repositories*, pp. 129–132, Leipzig, Germany, 2008.

[41] C. A. Halverson, J. B. Ellis, C. Danis, and W. A. Kellogg, "Designing task visualizations to support the coordination of work in software development," in *Proceedings of the 20th Anniversary ACM Conference on Computer Supported Cooperative Work (CSCW '06)*, pp. 39–48, Banff, Canada, November 2006.

[42] B. Hanks, "Empirical evaluation of distributed pair programming," *International Journal of Human Computer Studies*, vol. 66, no. 7, pp. 530–544, 2008.

[43] T. Heistracher, T. Kurz, G. Marcon, and C. Masuch, "Collaborative software engineering with a digital ecosystem," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 119–123, Florianopolis, Brazil, October 2006.

[44] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, dependencies, and delay in a global collaboration," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 319–328, Philadelphia, Pa, USA, December 2000.

[45] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: distance and speed," in *Proceedings of the 23rd International Conference on Software Engineering*, pp. 81–90, Toronto, Canada, May 2001.

[46] 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 (SHARK '08)*, Leipzig, Germany, May 2008.

[47] J. D. Herbsleb, D. J. Paulish, and M. Bass, "Global software development at Siemens: experience from nine project," in

Proceedings of the 27th International Conference on Software Engineering (ICSE '05), pp. 524–533, St. Louis, Mo, USA, May 2005.

[48] R. Holmes and R. J. Walker, "Promoting developer-specific awareness," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '08)*, Leipzig, Germany, May 2008.

[49] H. Holmstrom, E. Ó. Conchúir, P. J. Ågerfalk, and B. Fitzgerald, "Global software development challenges: a case study on temporal, geographical and socio-cultural distance," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 3–11, Florianopolis, Brazil, October 2006.

[50] K. V. Siakas and B. Balstrup, "Software outsourcing quality achieved by global virtual collaboration," *Software Process: Improvement and Practice*, vol. 11, no. 3, pp. 319–328, 2006.

[51] J. Kotlarsky, P. C. van Fenema, and L. P. Willcocks, "Developing a knowledge-based perspective on coordination: the case of global software projects," *Information and Management*, vol. 45, no. 2, pp. 96–108, 2008.

[52] R. Kuni and N. Bhushan, "IT application assessment model for global software development," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 92–100, Florianopolis, Brazil, October 2006.

[53] F. Lanubile, T. Mallardo, and F. Calefato, "Tool support for geographically dispersed inspection teams," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 217–231, 2003.

[54] L. Layman, L. Williams, D. Damian, and H. Bures, "Essential communication practices for Extreme Programming in a global software development team," *Information and Software Technology*, vol. 48, no. 9, pp. 781–794, 2006.

[55] G. Lee, W. DeLone, and J. A. Espinosa, "Ambidextrous coping strategies in globally distributed software development projects," *Communications of the ACM*, vol. 49, no. 10, pp. 35–40, 2006.

[56] E. Lindqvist, B. Lundell, and B. Lings, "Distributed development in an intra-national, intra-organisational context: an experience report," in *Proceedings of the International Workshop on Global Software Development for the Practitioner*, Shanghai, China, May 2006.

[57] G. N. Aranda, A. Vizcaíno, A. Cechich, M. Piattini, and J. J. Castro-Sáchez, "Cognitive-based rules as a means to select suitable groupware tools," in *Proceedings of the 5th IEEE International Conference on Cognitive Informatics*, vol. 1, pp. 418–423, Beijing, China, July 2006.

[58] S. Liu, Y. Zheng, H. Shen, S. Xia, and C. Sun, "Real-time collaborative software modeling using UML with rational software architect," in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom '06)*, Atlanta, Ga, USA, November 2006.

[59] W. J. Lloyd, M. B. Rosson, and J. D. Arthur, "Effectiveness of elicitation techniques in distributed requirements engineering," in *Proceedings of the 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE '02)*, Essen, Germany, September 2002.

[60] J. Ma, J. Li, W. Chen, R. Conradi, J. Ji, and C. Liu, "A state-of-the-practice study on communication and coordination between Chinese software suppliers and their global outsourcers," *Software Process: Improvement and Practice*, vol. 13, no. 3, pp. 233–247, 2008.

[61] R. J. Madachy, "Cost modeling of distributed team processes for global development and software-intensive systems of systems," *Software Process: Improvement and Practice*, vol. 13, no. 1, pp. 51–61, 2008.

[62] N. B. Moe and D. Šmite, "Understanding a lack of trust in global software teams: a multiple-case study," *Software Process: Improvement and Practice*, vol. 13, no. 3, pp. 217–231, 2008.

[63] K. Mohan and B. Ramesh, "Traceability-based knowledge integration in group decision and negotiation activities," *Decision Support Systems*, vol. 43, no. 3, pp. 968–989, 2007.

[64] J. Van Moll, J. Jacobs, R. Kusters, and J. Trienekens, "Defect detection oriented lifecycle modeling in complex product development," *Information and Software Technology*, vol. 46, no. 10, pp. 665–675, 2004.

[65] B. E. Munkvold and I. Zigurs, "Process and technology challenges in swift-starting virtual teams," *Information and Management*, vol. 44, no. 3, pp. 287–299, 2007.

[66] N. Nagappan, B. Murphy, and V. R. Basili, "The influence of organizational structure on software quality: an empirical case study," in *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, pp. 521–530, Leipzig, Germany, May 2008.

[67] K. Narayanaswamy and N. M. Goldman, "A flexible framework for cooperative distributed software development," *The Journal of Systems and Software*, vol. 16, no. 2, pp. 97–105, 1991.

[68] R. M. De Araujo and M. R. S. Borges, "The role of collaborative support to promote participation and commitment in software development teams," *Software Process: Improvement and Practice*, vol. 12, no. 3, pp. 229–246, 2007.

[69] R. J. Ocker and J. Fjermestad, "Communication differences in virtual design teams: findings from a multi-method analysis of high and low performing experimental teams," *ACM SIGMIS Database*, vol. 39, no. 1, pp. 51–67, 2008.

[70] P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a coordination tool in multi-site software development," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 233–247, 2003.

[71] M. Paasivaara and C. Lassenius, "Collaboration practices in global inter-organizational software development projects," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 183–199, 2003.

[72] L. Pilatti, J. L. N. Audy, and R. Prikladnicki, "Software configuration management over a global software development environment: lessons learned from a case study," in *Proceedings of the International Workshop on Global Software Development for the Practitioner (GSD '06)*, Shanghai, China, May 2006.

[73] A. Powell, G. Piccoli, and B. Ives, "Virtual teams: a review of current literature and directions for future research," *ACM SIGMIS Database*, vol. 35, no. 1, pp. 6–23, 2004.

[74] R. Prikladnicki, J. L. N. Audy, and R. Evaristo, "Global software development in practice lessons learned," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 267–281, 2003.

[75] R. Prikladnicki, J. L. N. Audy, and R. Evaristo, "A reference model for global software development: findings from a case study," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 18–28, Florianopolis, Brazil, October 2006.

[76] N. Ramasubbu and R. K. Balan, "Globally distributed software development project performance: an empirical analysis," in *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM*

SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE '07), pp. 125–134, Dubrovnik, Yugoslavia, September 2007.

[77] S. Sakthivel, "Virtual workgroups in offshore systems development," *Information and Software Technology*, vol. 47, no. 5, pp. 305–318, 2005.

[78] R. S. Sangwan and J. Ros, "Architecture leadership and management in globally distributed software development," in *Proceedings of the 1st International Workshop on Leadership and Management in Software Architecture*, pp. 17–21, Leipzig, Germany, May 2008.

[79] M. A. Babar, B. Kitchenham, L. Zhu, I. Gorton, and R. Jeffery, "An empirical study of groupware support for distributed software architecture evaluation process," *Journal of Systems and Software*, vol. 79, no. 7, pp. 912–925, 2006.

[80] S. Sarkar, R. Sindhgatta, and K. Pooloth, "A collaborative platform for application knowledge management in software maintenance projects," in *Proceedings of the 1st Bangalore Annual Compute Conference*, Bangalore, India, January 2008.

[81] A. Sarma, Z. Noroozi, and A. Van der Hoek, "Palantír: raising awareness among configuration management workspaces," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 444–454, Portland, Ore, USA, May 2003.

[82] S.-O. Setamanit, W. Wakeland, and D. Raffo, "Using simulation to evaluate global software development task allocation strategies," *Software Process: Improvement and Practice*, vol. 12, no. 5, pp. 491–503, 2007.

[83] N. S. Shami, N. Bos, Z. Wright, et al., "An experimental simulation of multi-site software development," in *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, Markham, Canada, October 2004.

[84] B. Sengupta, S. Chandra, and V. Sinha, "A research agenda for distributed software development," in *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*, pp. 731–740, Shanghai, China, May 2006.

[85] D. Šmite, "Global software development projects in one of the biggest companies in Latvia: is geographical distribution a problem?" *Software Process: Improvement and Practice*, vol. 11, no. 1, pp. 61–76, 2006.

[86] C. R. de Souza, S. Quirk, E. Trainer, and D. F. Redmiles, "Supporting collaborative software development through the visualization of socio-technical dependencies," in *Proceedings of the International ACM Conference on Supporting Group Work*, pp. 147–156, Sanibel Island, Fla, USA, 2007.

[87] H. Spanjers, M. ter Huurne, B. Graaf, M. Lormans, D. Bendas, and R. van Solingen, "Tool support for distributed software engineering," in *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 187–198, Florianopolis, Brazil, October 2006.

[88] M.-A. D. Storey, D. Čubranić, and D. M. German, "On the use of visualization to support awareness of human activities in software development: a survey and a framework," in *Proceedings of the ACM Symposium on Software Visualization (SoftVis '05)*, pp. 193–202, St. Louis, Mo, USA, May 2005.

[89] J. Suzuki and Y. Yamamoto, "SoftDock: a distributed collaborative platform for model-based software development," in *Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA '99)*, Florence, Italy, September 1999.

[90] M. Baentsch, G. Molter, and P. Sturm, "WebMake: integrating distributed software development in a structure-enhanced Web," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 789–800, 1995.

[91] Y. Tamura, S. Yamada, and M. Kimura, "A reliability assessment tool for distributed software development environment based on Java and J/Link," *European Journal of Operational Research*, vol. 175, no. 1, pp. 435–445, 2006.

[92] L. Taxén, "An integration centric approach for the coordination of distributed software development projects," *Information and Software Technology*, vol. 48, no. 9, pp. 767–780, 2006.

[93] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin, "Communication tools for distributed software development teams," in *Proceedings of the ACM SIGMIS CPR Conference: The Global Information Technology Workforce*, pp. 28–35, Saint Louis, Mo, USA, April 2007.

[94] P. F. Tiako, "Collaborative approach for modeling and performing mobile software process components," in *Proceedings of the International Symposium on Collaborative Technologies and Systems*, pp. 40–47, Saint Louis, Mo, USA, May 2005.

[95] S. Vale and S. Hammoudi, "Towards context independence in distributed context-aware applications by the model driven approach," in *Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments*, Sorrento, Italy, July 2008.

[96] P. Wongthongtham, E. Chang, and T. S. Dillon, "Ontology-based multi-agent system to multi-site software development," in *Proceedings of the Workshop on Quantitative Techniques for Software Agile Process*, Newport Beach, Calif, USA, November 2004.

[97] P. Wongthongtham, E. Chang, T. S. Dillon, and I. Sommerville, "Ontology-based multi-site software development methodology and tools," *Journal of Systems Architecture*, vol. 52, no. 11, pp. 640–653, 2006.

[98] W. Xiao, C. Chi, and M. Yang, "On-line collaborative software development via wiki," in *Proceedings of the International Symposium on Wikis*, pp. 177–183, Montreal, Canada, October 2007.

[99] H. Zhuge, "Knowledge flow management for distributed team software development," *Knowledge-Based Systems*, vol. 15, no. 8, pp. 465–471, 2002.

[100] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?" *Communications of the ACM*, vol. 49, no. 10, pp. 41–46, 2006.

[101] J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski, "IMPROMPTU: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development," in *Proceedings of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems*, pp. 939–948, Florence, Italy, April 2008.