Kay Berkling
Mathai Joseph
Bertrand Meyer
Martin Nordio (Eds.)

Software Engineering Approaches for Offshore and Outsourced Development

LNBIP 16

Second International Conference, SEAFOOD 2008
Zurich, Switzerland, July 2008
Revised Papers

Lecture Notes
in Business Information Processing 16

Kay Berkling Mathai Joseph
Bertrand Meyer Martin Nordio (Eds.)

Software Engineering Approaches for Offshore and Outsourced Development

Second International Conference, SEAFOOD 2008
Zurich, Switzerland, July 2-3, 2008
Revised Papers

Springer

Volume Editors

Kay Berkling
Polytechnic University of Puerto Rico
00919 San Juan, Puerto Rico
E-mail: kay@berkling.com

Mathai Joseph
Tata Consultancy Services
Pune 411 001, India
E-mail: m.joseph@tcs.com

Bertrand Meyer
Martin Nordio
ETH Zurich
8092 Zurich, Switzerland
E-mail: {bertrand.meyer,martin.nordio}@inf.ethz.ch

Library of Congress Control Number: Applied for

ACM Computing Classification (1998): K.6, D.2

ISSN 1865-1348
ISBN-10 3-642-01855-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-01855-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12681969 06/3180 5 4 3 2 1 0

Preface

Major economic upheavals can have the sort of effect that Schumpeter foresaw 60 years ago as creative destruction. In science and technology, equivalent upheavals result from either scientific revolutions (as observed by Kuhn) or the introduction of what Christensen calls disruptive technologies. And in software engineering, there has been no technology more disruptive than outsourcing. That it should so quickly reach maturity and an unparalleled scale is truly remarkable; that it should now be called to demonstrate its sustainability in the current financial turmoil is the challenge that will prove whether and how it will endure. Early signs under even the bleak market conditions of the last 12 months are that it will not only survive, it will firmly establish its role across the world of business.

Outsourcing throws into sharp focus the entire software engineering lifecycle. Topics as diverse as requirements analysis, concurrency and model-checking need to find a composite working partnership in software engineering practice. This confluence arises from need, not dogma, and the solutions required are those that will have the right effect on the associated activities in the world of the application: e.g., reducing the time for a transaction or making the results of a complex analysis available in real-time. While the business of outsourcing continues to be studied, the engineering innovations that make it compelling are constantly changing. It is in this milieu that this series of conferences has placed itself.

SEAFOOD 2008, the Second International Conference on Software Engineering Approaches to Outsourcing and Offshore Development, was held in Zurich during July 2-3, 2008. There were outstanding invited talks by Ashish Arora (then at the Heinz School, Carnegie-Mellon University) and Dick Simmons (Texas A&M University), the first on how outsourcing has grown in countries as different as India, Israel and Ireland, and the second on the effects of outsourcing on software engineering in the past, the present and the future.

SEAFOOD 2008 received submissions spanning a wide range of topics, from processes, and risks to education in distributed software development. This volume includes 14 papers from the conference selected after review by the Program Committee. SEAFOOD 2008 received 50 submissions; the acceptance rate was 28%. Papers covered areas such as extreme programming and code review, predicting timelines in software development subject to changes and software process improvement in small companies. There was an outstanding panel discussion (not reported in this volume) organized by Peter Kolb with speakers from banking, insurance and engineering industries.

Many people contributed to SEAFOOD 2008. We thank the Program Committee and the external reviewers for their excellent work in reviewing and selecting papers. SEAFOOD 2008 was co-located with TOOLS 2008; we are grateful to Manuel Oriol and Marco Piccioni for their support and to Claudia Günthart

for once again providing with unwavering efficiency the organization that made SEAFOOD 2008 a success.

March 2009

Mathai Joseph
Bertrand Meyer
Martin Nordio

Organization

Program Chairs

Bertrand Meyer	ETH Zürich, Switzerland - Co-chair
Mathai Joseph	Tata Consultancy Services, India - Co-chair
Kay Berkling	Polytechnic University of Puerto Rico, Puerto Rico - Program Chair
Peter Kolb	Red Expel, Switzerland - Panel Chair

Program Committee

Gabriel Baum	La Plata National University, Argentina
Manfred Broy	Technische Universität München, Germany
Jean Pierre Corriveau	School of Computer Science, Carleton University, Canada
Barry Dwolatzky	South Africa
Kokichi Futatsugi	Japan Advanced Institute of Science and Technology, Japan
Victor Gergel	University of Nizhnyi-Novgorod, Russia
Amar Gupta	University of Arizona, USA
Pankaj Jalote	IIT Delhi, India
Koichi Kashida	SRA Key-Tech Lab, Japan
Philippe Kruchten	University of British Columbia, Canada
Mingshu Li	Chinese Academy of Sciences, China
Christine Mingins	Monash University, Australia
Jianjun Zhao	School of Software, Shanghai Jiao Tong University, China
Cleidson de Souza	Federal University of Para, Brazil

Local Organization

Martin Nordio	ETH Zürich, Switzerland
Claudia Günthart	ETH Zürich, Switzerland

External Reviewers

Dong, Fei
Fritzsche, Martin
He, Mei
Islam, Shareeful
Juergens, Elmar
Keil, Patrick
Kishida, Koichi
Kong, Weiqiang
Kuhrmann, Marco
Li, Yin
Liu, Dapeng
Mattarelli, Elisa
Nakamura, Masaki
Ogata, Kazuhiro
Pister, Markus
Smith, David
Sudaman, Fadrian
Wagner, Stefan
Wu, Shujian
Xie, Lizi
Yang, Da
Zundel, Armin

Table of Contents

Problems and Solutions in Distributed Software Development: A Systematic Review

Miguel Jiménez[1] and Mario Piattini[2]

[1] Alhambra-Eidos
Technological Innovation Center
Paseo de la Innovación 1, 02006, Albacete, Spain
Miguel.Jimenez@a-e.es
[2] University of Castilla-La Mancha
Alarcos Research Group
Institute of Information Technologies & Systems
Escuela Superior de Informática
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
Mario.Piattini@uclm.es

Abstract. Nowadays software development activity tends to be decentralized, thus expanding greater development efforts towards more attractive zones for organizations. The type of development in which the team members are distributed in remote sites is called Distributed Software Development (DSD). A variant of the DSD is Global Software Development (GSD), where the team is distributed beyond the borders of a nation. The main advantage of this practice is mainly that of having a greater availability of human resources in decentralized zones with less cost. On the other hand, some disadvantages appear due to the distance that separates the development teams. This article presents a systematic review of the literature related to the problems and the solutions proposed up to the present day in DSD and GSD with the purpose of obtaining a vision of the state-of-the-art which will allow us to identify possible new research lines.

Keywords: Distributed Software Development, DSD, Global Software Development, GSD, Offshore, Outsource, Nearshore, Systematic Review.

1 Introduction

Nowadays, many organizations, especially those dedicated to Information Technology (IT), and concretely the software industry, are tending to relocate their production units, mainly to take advantage of the greater availability of qualified labor in decentralized zones. The objective consists of optimizing resources in order to develop higher quality products at a lower cost. With the same purpose, "software factories" [1] attempt to imitate industrial processes originally linked to more traditional sectors such as those of the automobile and aviation, by decentralizing production units, and promoting the reusability of architectures, knowledge and components.

K. Berkling et al. (Eds.): SEAFOOD 2008, LNBIP 16, pp. 107–125, 2009.

Distributed Software Development (DSD) allows the team members to be located in various remote sites, thus making up a network of distant sub-teams. In this context the traditional face-to-face meetings are no longer common and interaction between members requires the use of technology to facilitate communication and coordination.

The distance between the different teams can vary from a few meters (when the teams work in adjacent buildings) to different continents. The special situation in which the teams are distributed beyond the limits of a nation is called Global Software Development (GSD). This kind of scenario is interesting for several reasons [2], mainly because it enables organizations to abstract themselves from geographical distance, whilst having qualified human resources and minimizing cost [3], increasing their market area by producing software for remote clients and obtaining a longer workday by taking advantage of time differences [4]. On the other hand we must confront a number of problems [5], caused mainly by distance and time and cultural differences [6], which depend largely on the specific characteristics of each organization.

In this context, GSD is experiencing a boom thanks to offshoring and nearshoring. Offshoring involves the transfer of an organizational function to another country, usually where 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.

The aforementioned development practices have as a common factor the problems arising from distance that directly affect the processes of communication as well as coordination and control activities [7]. In these environments, communication is less fluid than in colocalized development groups, as a consequence, problems related to coordination, collaboration or group awareness appear which negatively affect productivity and, consequently, software quality. All these factors 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, companies need to incorporate new technologies, processes and methods [8], and research into this field is therefore necessary.

This article presents a systematic review of the literature dealing with efforts related to DSD with the purpose of discovering the aspects upon which researchers have focused until this moment. The objective is to identify, evaluate, interpret and synthesize most of the important studies on the subject, by conducting a rigorous and objective review of literature which will allow us to analyze the issues and the solutions contributed up to the present in the fields of DSD and GSD with the aim of obtaining information with a high scientific and practical value through a rigorous systematic method.

2 The Importance of Systematic Reviews

A systematic review of literature [9] permits the identification, evaluation and interpretation of all the available relevant studies related to a particular research question, topic area or phenomenon, providing results with a high scientific value by classifying studies between 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, serving this work as a starting point to focus on future research to be done about DSD.

In order to carry out this study we have followed the systematic search procedure proposed by [9], and the selection of primary studies method followed in [10].

2.1 Question Formularization

The research question is: 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*.

During a first iteration, we also included the keywords CMM, CMMI, COBIT and ITIL in an attempt to obtain studies based on these standards, but due to the scarcity of good results these words were misestimated in subsequent iterations.

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

By combining the keyword list from the previous section through the logical connectors "AND" and "OR", we established the search strings shown in Table 1.

The studies were obtained from the search sources: *Science@Direct*, *Wiley Interscience*, *IEEE Digital Library*, *ACM Digital Library* and *EBSCO Host*. 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 if a study should be considered relevant (potential candidate to become a primary study) was based on analyzing the title, abstract

and keywords from the studies retrieved by the search to determine whether they dealt with the DSD subject orientated towards process improvement, quality, coordination, collaboration, communication and related issues that carry on any improvement about the subject.

Upon analyzing the results of the first iteration of the systematic review, we decided to exclude those studies which, despite addressing the issue of DSD, did not contribute to any significant improvement method, and we also dismissed those studies which focused solely upon social issues, cultural or time differences or focused solely upon free software, although we have taken into account other articles that address these topics in a secondary manner.

To obtain the primary studies we have followed the iterative and incremental model proposed by [10]. It is iterative because the search, retrieval and information visualization of results is carried out entirely through an initial search source and then repeats the same process on the rest. It is incremental because the document evolves incrementally, including new studies to complete the final version.

By applying the procedure to obtain the primary studies, 2224 initial studies were found, of which 518 were not repeated. From these, we selected 200 as relevant and 69 as primary studies (the complete list of primary studies is shown in Appendix A). Table 2 shows the distribution of studies found according to the sources employed.

Table 2. Distribution of studies found

	Studies					
Sources	Search date	Found	Not repeated	Relevant	**Primaries**	%
Science@Direct	07/11/2007	160	132	51	**18**	26,1
Wiley InterScience	08/11/2007	22	15	12	**9**	13,0
IEEE Digital Library	19/11/2007	60	30	30	**21**	30,4
ACM Digital Library	19/11/2007	1898	273	88	**15**	21,7
EBSCO Host	19/11/2007	84	68	19	**6**	8,7
	Total	2224	518	200	**69**	100,0

2.4 Information Extraction

The process of extracting information from the primary studies followed an inclusion criterion based on obtaining information about 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 attending to the methodology study followed according to the models presented in [11]. We used the following categories: case studies, literature review, experiment, simulation and survey. The nonexperimental model for studies which makes a proposal without testing it or performing experiments was also applied.

3 Trends in Distributed Software Development Research

This section analyzes and discusses proposals and success factors in order to extract relevant information from the information provided by the primary studies.

Figure 1 (*left*) shows that most of the primary studies analyzed are case studies and nonexperimental articles. Surveys also have a significant representation, in which members involved in the distributed development take part in outlining their difficulties.

On the other hand, as is shown in Figure 1 (*right*), the majority of primary studies are focused upon the enterprise field, but studies in the university environment also appear, in which groups of students carried out developments in different locations. Near the half of the studies did not indicate their field of work or their characterization did not proceed, while 10% were from organizations which did not specify their corporate or university environment.

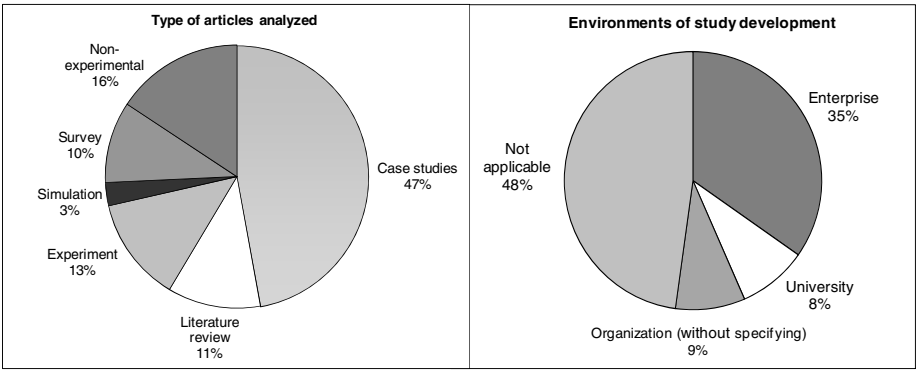

Fig. 1. Type of articles analyzed (*left*) and environments of study development (*right*)

3.1 Publications Tendency

After attending to 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 it is only recently that a greater number of publications have appeared; thus in Figure 2 we can see that 2006 is by far the year in which most studies were published, bearing in mind that the data shown for 2007 only reflects the studies found before the middle of November.

3.2 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 [12], with the aim of obtaining a vision of the processes life cycle that requires special attention when working in a distributed environment and discovering the improvement efforts carried out until that moment.

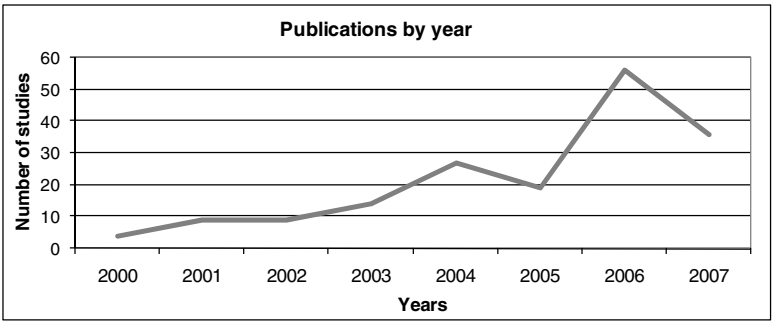

Fig. 2. Trends in publications about DSD

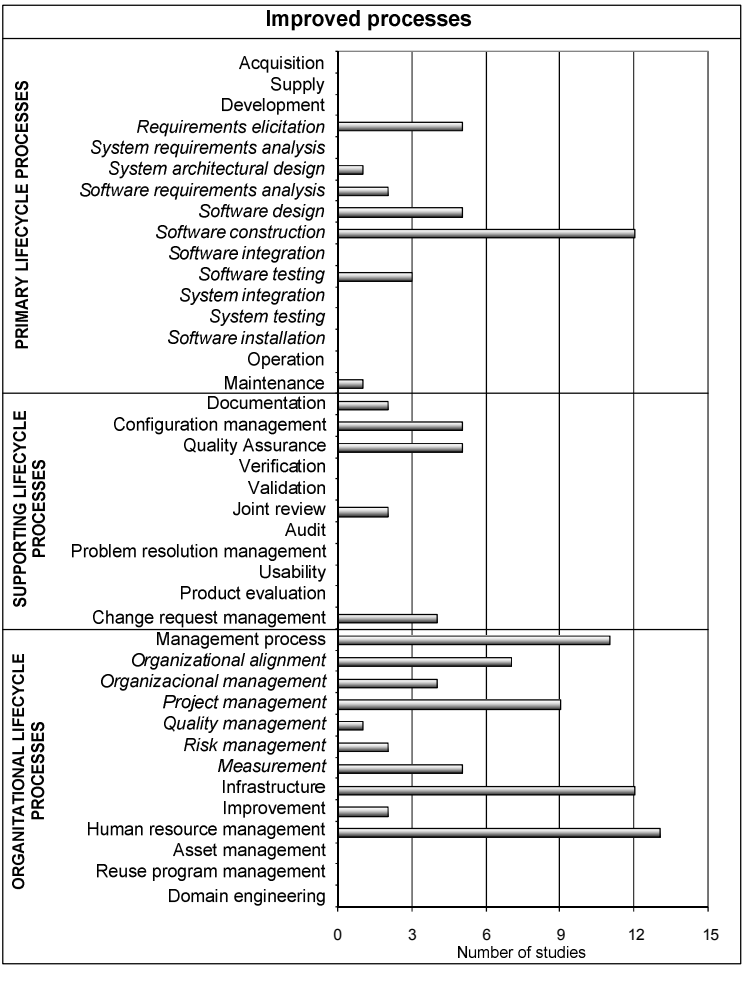

Fig. 3. Improved or analyzed processes by the primary studies adjusted to ISO 12207

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 3 where for every process, its frequency in function of the number of studies that address it is indicated.

The results obtained indicate that greater efforts are focused on human resources, infrastructure, software construction and management and project organization processes. From these data we can infer that communication between team members is a critical factor. On the other hand, other processes, such as software installation or usability are not mentioned in any study. This information will be useful in the focusing of future research efforts.

3.3 Employed Standards

Figure 4 presents the standards that the analyzed articles address. 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 great majority of studies deal with issues such as communication difficulties in which the standard used does not matter. The standards supported by most primary studies are CMM and ISO 9001, it being common to jointly apply both. All applications of CMM and CMMI studied employed a maturity level 2 with the exception of one which was certified at CMM level 5. No studies relative to ITIL or COBIT models were obtained.

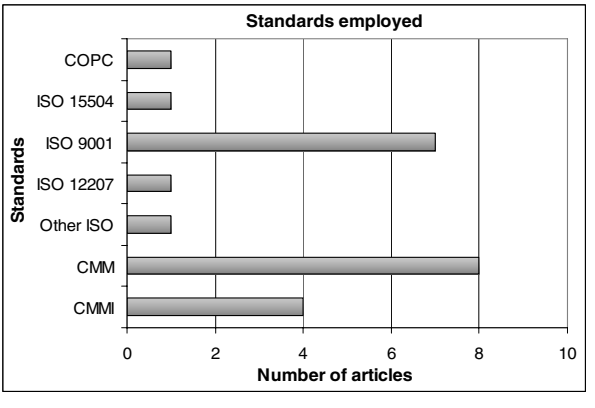

Fig. 4. Standards employed in the studies

3.4 Contents of the Studies

Table 3 shows in a schematic way 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 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.

On the other hand, 62% of the studies analyze or provide strategies, procedures or frameworks related to DSD. The remaining 38% study tools were designed specifically for distributed environments. As an example, tools such as FASTDash [13], Augur [14] or MILOS [15] may be of particular interest.

Table 3. Thematic areas dealt with in the primary studies

Thematic areas	Studies (%)
Collaborative tools, techniques and frameworks orientated towards communication and integration of existing tools	41,8
Process control, task scheduling and project coordination	34,2
Configuration management	6,3
Multi-agent systems	6,3
Knowledgement management	5,1
Test management	3,8
Defects detection	2,5

4 Problems and Solutions

In this section, we synthesize the problems and solutions identified through the systematic review, discussing the main subjects.

4.1 Communication

The software life cycle, especially in its early stages, requires a great deal of communication between members involved in the development who exchange a large number of messages through different tools and different formats without following communication standards and facing misunderstandings and high response times. These drawbacks, combined with the great size of personal networks which change over time, are summarized in a decrease in communication frequency and quality which directly affects productivity. To decrease these effects, both methodologies and processes must be supported by collaborative tools as a means of avoiding face-to-face meetings without comprising the quality of the results, as is proposed by M.A. Babar et al. [PS3]. K. Mohan and B. Ramesh [PS29] discuss the need for user-friendly tools, integrating collaborative tools and agents to improve knowledge integration. M.R. Thissen et al. [PS55] examine communication tools and describe collaboration processes, dealing with techniques such as conference calls and email.

Cultural differences imply different terminologies which cause mistakes in messages and translation errors. Different levels of understanding the problem domain exist, as do different levels of knowledge, skills and training between teams. The use of translation processes, and codification guidelines is therefore useful [PS10, PS65].

4.2 Group Awareness

Members who are part of a virtual team tend to be less productive due to feelings of isolation and indifference. They have little informal conversation across sites, and their trust is reduced. Developers need to know the full status of the project and past history which will allow them to create realistic assumptions about how work is done on other sites. Frequent changes in processes, lack of continuity in communications and lack of collaborative tool integration cause the 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, rework and other coordination problems.

M.A.S. Mangan et al. [PS35] present Odyssey, a middleware for collaborative applications that increases group and workspace awareness information available to developers, helping them to reuse existing applications. On the other hand, J. Froehlich and P. Dourish [PS17] describe Augur, a visualization tool that supports DSD processes by creating visual representations of both software artifacts and software development activities, thus allowing developers to explore relationships between them. In the same context, S. Dustdar and H. Gall [PS15] study current technologies such as peer-to-peer, workflow management and groupware systems.

J.D. Herbsleb et al. [PS26] present a tool that provides a visualization of the change management system, 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 this line C. Gutwin et al. [PS22] propose using social networks to discover the experts in a specific area and project documentation to provide direct information about activities and areas of work that must be kept up to date.

4.3 Source Control

Distributed environments present problems derived from conflicts caused by editing files simultaneously. Coordination and synchronization become more complex as the degree of distribution of the team grows. Source control systems must support access through internet, confronting its unreliable and insecure nature and the higher response times.

To reduce these drawbacks, S.E. Dossick and G.E. Kaiser [PS14] 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. With the same purpose, J.T. Biehl et al. [PS6] present FASTDash as a user-friendly tool that uses a spatial representation of the shared code base which highlights team members' current activities, allowing a developer to determine rapidly 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.

4.4 Knowledge Flow Management

The team members' experiences, methods, decisions, and skills must be accumulated during the development process, so that each team member can use the experience of his/her predecessor and the experience of the team accumulated during development, saving cost and time by avoiding redundant work. For this purpose, documentation must always be updated to prevent assumptions and ambiguity, therefore facilitating the maintainability of the software developed. 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, H. Zhuge [PS60] presents an approach that works with a knowledge repository in which information related to every project is saved, using internet-based communication tools and thus enabling a new team member to become quickly experienced by learning the knowledge stored.

K. Mohan and B. Ramesh [PS29] 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 knowledge fragments stored in diverse environments to be integrated and used by various stakeholders in order to facilitate a common understanding.

4.5 Coordination

Coordination can be interpreted as the management of the right information, the right people and the right time to develop an activity. Coordination in multi-site 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 managed. In this sense, more progress reports, project reviews, conference calls and regular meetings to take corrective action are needed, thus minimizing task dependencies with other locations. Collaborative tools must support analysis, design and development, allowing monitoring activities and managing dependencies, notifications and implementation of corrective measures [PS5]. We shall deal with many of these issues in the following sections.

P. Ovaska et al. [PS39] 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.S. Vibha et al. [PS66] propose a framework that enables a common understanding of the information from different tools and supports loose coupling between them. S. Setamanit et al. [PS50] 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.

J.D. Herbsleb et al. [PS26] suggest that multi-site communication and coordination requires more people to participate, which causes a delay. Large changes involve multiple sites and greater implementation times. Changes in multiple distributed sites involve a large number of people.

4.6 Collaboration

Concurrent edition of models and processes requires synchronous collaboration between architects and developers who cannot be physically present at a common location. Software modelling requires concurrency control in real time, 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.

A. De Lucia [PS62] proposes STEVE, a collaborative tool that supports distributed modelling of software systems which, provides a communication infrastructure to enable concurrent edition of the same diagram at the same time by several distributed developers.

A further approach is presented by J. Suzuki and Y. Yamamoto [PS51] with the SoftDock framework which solves the issues related to software component modelling and their relationships, describing and sharing component models information, and ensuring the integrity of these models. Developers can therefore work analyzing, designing, and developing software from component models and transfer them using an exchange format, thus enabling communication between team members.

In another direction, X. WenPeng et al. [PS69] study Galaxy Wiki, an on-line collaborative tool based on the wiki concept which enables a collaborative authoring system for documentation and coordination purposes, allowing developers to compile, execute and debug programs in wiki pages.

4.7 Project and Process Management

Due to high organizational complexity, scheduling and task assignment becomes more problematic in distributed environments because of volatile requirements, changing specifications, and the lack of informal communication and synchronization. Managers must control the overall development process, improving it during the enactment and minimizing the factors that may decrease productivity, taking into account the possible impacts of diverse cultures and attitudes.

In this context, S. Goldmann et al. [PS19] and S. Bowen and F. Maurer [PS7] explain the main ideas of MILOS, a system orientated towards planning and scheduling which supports process modeling and enactment.

N. Ramasubbu et al. [PS43] propose a process maturity framework with 24 key process areas which are essential for managing distributed software development and capabilities for a continuously improving product management applicable to the CMM framework.

The maturity of the process becomes a key factor for success. In this sense, M. Passivaara and C. Lassenius [PS36] propose incremental integration and frequent deliveries by following informing and monitoring practices. In the same mindset J. Cusick and A. Prasad [PS12] include a set of recommendations based on experience, such as limiting phase durations to maintain control by breaking large projects into medium-size bundles, requiring interim deliverables to ensure quality or enforcing quality through coding standards and verification.

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 information overload of team members. Process modeling and enactment should support inter-site 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. [PS2] present the process modelling environment SPEARMINT, which supports extensive capabilities for multi-view modelling and analysis, and XCHIPS for web-based process support which allows enactment and simulation functionalities. Y. Yang and P. Wojcieszak [PS58] propose a web-based visual environment to support process modelling for software project managers and process enactment for software developers in an asynchronous and/or synchronous manner.

S. Setamanit et al. [PS50] describe a hybrid computer simulation model of software development processes to study alternative ways to configure GSD projects in order to confront communication problems, control and coordination problems, process management and time and cultural differences.

N. Glasser and J-C. Derniane [PS18] analyse CoMoMAS, a multi-agent engineering approach that describes different view points in a software process, permitting the transformation of conceptual models into executable programs. In this context, the agents will be able to cover with the high mobility of the members involved in the development process, taking charge of the management of information and permitting artifacts to communicate both with each other and with human users.

4.9 Quality and Measurement

Quality of products is highly influenced by the quality of the processes that support them. Organizations need to introduce new models and metrics to obtain information adapted to the distributed scenarios that could be useful in improving products and processes. With this aim, K.V. Siakas and B. Balstrup [PS30] 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, which considers the factors that influence software quality management systems from a cultural and organizational perspective.

J.D. Herbsleb et al. [PS25] work with several interesting measures, such as the *interdependence measure* which allows the determination of the degree of dispersion of work among sites by looking up the locations of all the individuals. In this sense, F. Lanubile et al. [PS16] propose metrics associated with products and processes orientated towards software defects such as: discovery effort, reported defects, defects density, fixed defects or unfixed defects. D.B. Simmons [PS48] presents PAMPA 2 Knowledge Base to measure the effectiveness of virtual teams by gathering information from completed projects.

Furthermore, software architecture evaluation usually involves a large number of stakeholders, who need face-to-face evaluation meetings, and for this reason adequate collaborative tools are needed, such as propose M.A. Babar et al. [PS3].

4.10 Defects Detection

In distributed environments it is necessary to specify requisites with a higher level of detail. Software defects become more frequent due to the added complexity, and in most cases, this is related to communication problems and lack of group awareness. Defects control must be adapted by making a greater effort in relation to risk management activities.

To minimize these problems, F. Lanubile et al. [PS16] define a process, specifying roles, guidelines, forms and templates, and describe a web-based tool that adopts a reengineered inspection process to minimize synchronous activities and coordination problems to support geographically dispersed teams.

An adequate model cycle must allow the localization and recognition of defect-sensitive areas in complex product development. In this line, Jv. Moll et al. [PS37] indicate that transitions between constituent sub-projects are particularly defect-sensitive. By means of an appropriate modelling of the overall project lifecycle and by applying adequate defect detection measures, the occurrence of defects can be reduced. The goal is to minimize the amount of defects that spread to the subsequent phases early in the software life cycle, and reuse existing components or the application of third-party components, thus minimizing product quality risks by using tested components.

5 Success Factors

From the experimental studies analyzed, we have extracted the following success factors of DSD, in which the primary studies referenced are listed in the Appendix A:

- Intervention of human resources by participating in surveys [PS3], [PS25].
- Carrying out the improvement 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 [PS15].
- Training of human resources in the tools and processes introduced [PS26].
- Registration of activities with information on pending issues, errors and people in charge [PS6].
- 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 [PS4], [PS6].
- Using a version control tool in order to control conflictive situations [PS40].
- There must be a way to allow the planning and scheduling of distributed tasks, taking into account dependencies between projects, application of corrective measures and notifications [PS17].
- Application of maturity models [PS43] and agile methodologies [PS33] based on incremental integration and frequent deliveries.
- Systematic use of metrics tailored to the organization [PS26].

6 Conclusions and Future Work

In this article we have applied a systematic review method in order to analyze the literature related to the topic of DSD within the FABRUM project context, this work serving as a starting point from which to establish the issues upon which subsequent research will be focused.

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 are the factors that make this such a wide subject, and lead to the necessity of adapting both the technical and organizational procedures, according to each organization's specific needs.

Generally, the proposals found in the analyzed studies were mainly concerned with improvements related to the use of collaborative tools, integration of existing tools, source code control or use of collaborative agents. Moreover, it should be considered 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 subjective perception of developers.

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 27,5% of all analyzed works. The fact that almost all experimental studies that employed CMMI and CMM applied a maturity level 2 suggests that the cost of implementing higher maturity levels under distributed environments might be too high. 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.

Finally, we must emphasize that the search 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, which is why their study would be important in a future work.

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

References

1. Greenfield, J., Short, K., Cook, S., Kent, S., Crupi, J.: Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. John Wiley & Sons, Chichester (2004)
2. Herbsleb, J.D., Moitra, D.: Guest editor's introduction: Global software development. IEEE Software 18(2), 16–20 (2001)

3. Werner, K., Rombach, D., Feldmann, R.: Outsourcing in India. IEEE Software, 78–86 (2001)
4. Christof Ebert, P.D.N.: Surviving Global Software Development. IEEE Software 18(2), 62–69 (2001)
5. Layman, L., Williams, L., Damian, D., Bures, H.: Essential communication practices for Extreme Programming in a global software development team. Information & Software Technology 48(9), 781–794 (2006)
6. Krishna, S., Sundeep, S., Geoff, W.: Managing cross-cultural issues in global software outsourcing. Commun. ACM 47(4), 62–66 (2004)
7. Damian, D., Lanubile, F., Oppenheimer, H.: Addressing the Challenges of Software Industry Globalization: The Workshop on Global Software Development. In: ICSE 2003, pp. 793–794 (2003)
8. Damian, D., Lanubile, F.: The 3rd International Workshop on Global Software Development. ICSE 2004, pp. 756–757 (2004)
9. Kitchenham, B.: Procedures for performing systematic reviews (Joint Technical Report). Software Engineering Group, Department of Computer Science, Keele University and Empirical Software Engineering National ICT Australia Ltd. (2004)
10. Pino, F.J., García, F., Piattini, M.: Software Process Improvement in Small and Medium Software Enterprises: A Systematic Review. Software Quality Journal (in press, 2007)
11. Marvin, V.Z., Dolores, R.W.: Experimental Models for Validating Technology, pp. 23–31 (1998)
12. ISO/IEC 12207: 2002/FDAM 2. Information technology - Software life cycle processes. Geneva: International Organization for Standardization (2004)
13. Biehl, J.T., Czerwinski, M., Smith, G., Robertson, G.G.: FASTDash: a visual dashboard for fostering awareness in software teams. In: Proceedings of the SIGCHI conference on Human factors in computing systems, San Jose, California, USA, pp. 28–35. ACM Press, New York (2007)
14. Froehlich, J., Dourish, P.: Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams, pp. 387–396 (2004)
15. Goldmann, S., Münch, J., Holz, H.: A Meta-Model for Distributed Software Development, pp. 48–53 (1999)

Appendix A: Primary Studies Selected

In this section the selected primary studies in the systematic review are presented.

Table 4. Primary studies selected in the systematic review

List of primary studies selected in the systematic review	
PS1	**Strategies for global information systems development.** Information & Management 42(1). Published in 2004. Pages: 45-59. Akmanligil M, Palvia PC.
PS2	**Guided support for collaborative modeling, enactment and simulation of software development processes.** Software Process: Improvement and Practice 9(2). Published in 2004. Pages: 95-106. Fernández A, Garzaldeen B, Grützner I, Münch J.
PS3	**An empirical study of groupware support for distributed software architecture evaluation process.** Journal of Systems and Software 79(7). Published in 2006. Pages: 912-925. Babar MA, Kitchenham B, Zhu L, Gorton I, Jeffery R.

PS4	**WebMake: Integrating distributed software development in a structure-enhanced Web.** Computer Networks and ISDN Systems 27(6). Published in 1995. Pages: 789-800. Baentsch M, Molter G, Sturm P.
PS5	**Coordinating Management Activities in Distributed Software Development Projects.** Published in 1998. Pages: 33-38. Bendeck F, Goldmann S, Kötting B.
PS6	**FASTDash: a visual dashboard for fostering awareness in software teams.** Proceedings of the SIGCHI conference on Human factors in computing systems. Published in 2007. Pages: 28-35. Biehl JT, Czerwinski M, Smith G, Robertson GG.
PS7	**Designing a Distributed Software Development Support System Using a Peer-to-Peer Architecture.** Published in 2002. Pages: 1087-1092. Bowen S, Maurer F.
PS8	**Supporting Agent-Based Distributed Software Development through Modeling and Simulation.** Published in 2003. Pages: 56-59. Cai L, Chang CK, Cleland-Huang J.
PS9	**How distribution affects the success of pair programming.** International Journal of Software Engineering & Knowledge Engineering 16(2). Published in 2006. Pages: 293-313. Canfora G, Cimitile A, Lucca GAD, Visaggio CA.
PS10	**Creating global software: A conspectus and review.** Interacting with Computers 9(4). Published in 1998. Pages: 449-465. Carey JM.
PS11	**Self-organization of teams for free/libre open source software development.** Information and Software Technology 49(6). Published in 2007. Pages: 564-575. Crowston K, Li Q, Wei K, Eseryel UY, Howison J.
PS12	**A Practical Management and Engineering Approach to Offshore Collaboration.** IEEE Software 23(5). Published in 2006. Pages: 20-29. Cusick J, Prasad A.
PS13	**Global software development projects in one of the biggest companies in Latvia: is geographical distribution a problem?** Software Process: Improvement and Practice 11(1). Published in 2006. Pages: 61-76. Darja m.
PS14	**CHIME: a metadata-based distributed software development environment.** Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering. Published in 1999. Pages: 464-475. Dossick SE, Kaiser GE.
PS15	**Process Awareness for Distributed Software Development in Virtual Teams.** Published in 2002. Pages: 244-251. Dustdar S, Gall H.
PS16	**Tool support for geographically dispersed inspection teams**. Software Process: Improvement and Practice 8(4). Published in 2003. Pages: 217-231. Lanubile, F, Mallardo T, Calefato F.
PS17	**Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams.** Published in 2004. Pages: 387-396. Froehlich J, Dourish P.
PS18	**Software Agents: Process Models and User Profiles in Distributed Software Development.** Published in 1998. Pages: 45-50. Glaser N, Derniame J-C.
PS19	**A Meta-Model for Distributed Software Development.** Published in 1999. Pages: 48-53. Goldmann S, Münch J, Holz H.
PS20	**Issues in co-operative software engineering using globally distributed teams.** Information and Software Technology 38(10). Published in 1996. Pages: 647-655. Gorton I, Motwani S.
PS21	**Coordinating Distributed Software Development Projects with Integrated Process Modelling and Enactment Environments.** Published in 1998. Pages: 39-44. Grundy J, Hosking J, Mugridge R.
PS22	**Group awareness in distributed software development.** Proceedings of the 2004 ACM conference on Computer supported cooperative work. Published in 2004. Pages: 72-81. Gutwin C, Penner R, Schneider K.

PS23	**Designing task visualizations to support the coordination of work in software development.** Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. Published in 2006. Pages: 39-48. Halverson CA, Ellis JB, Danis C, Kellogg WA.
PS24	**An Empirical Study of Speed and Communication in Globally Distributed Software Development.** IEEE Transactions on Software Engineering 29(6). Published in 2003. Pages: 481-492. Herbsleb JD, Mockus A.
PS25	**Distance, dependencies, and delay in a global collaboration.** Proceedings of the 2000 ACM conference on Computer supported cooperative work. Published in 2000. Pages: 319-328. Herbsleb JD, Mockus A, Finholt TA, Grinter RE.
PS26	**An empirical study of global software development: distance and speed.** Proceedings of the 23rd International Conference on Software Engineering. Published in 2001. Pages: 81-90. Herbsleb JD, Mockus A, Finholt TA, Grinter RE.
PS27	**Global software development at siemens: experience from nine projects.** Proceedings of the 27th international conference on Software engineering. Published in 2005. Pages: 524-533. Herbsleb JD, Paulish DJ, Bass M.
PS28	**Working Group Report on Coordinating Distributed Software Development Projects.** Published in 1998. Pages: 69-72. Holz H, Goldmann S, Maurer F.
PS29	**Traceability-based knowledge integration in group decision and negotiation activities.** Decision Support Systems 43(3). Published in 2007. Pages: 968-989. Mohan K., Ramesh B.
PS30	**Software outsourcing quality achieved by global virtual collaboration.** Software Process: Improvement and Practice 11(3). Published in 2006. Pages: 319-328. Siakas K.V., Balstrup B.
PS31	**Global software development: technical, organizational, and social challenges.** SIGSOFT Softw Eng Notes 28(6). Published in 2003. Pages: 2-2. Lanubile F, Damian D, Oppenheimer HL.
PS32	**Essential communication practices for Extreme Programming in a global software development team.** Information and Software Technology 48(9). Published in 2006. Pages: 781-794. Layman L, Williams L, Damian D, Bures H.
PS33	**Ambidextroux coping strategies in globally distributed software development projects.** Communications of the ACM 49(10). Published in 2006. Pages: 35-40. Lee G, Delone W, Espinosa JA.
PS34	**Distributed development in an intra-national, intra-organisational context: an experience report.** Proceedings of the 2006 international workshop on Global software development for the practitioner. Published in 2006. Pages: 80-86. Lindqvist E, Lundelll B, Lings B.
PS35	**A Middleware to Increase Awareness in Distributed Software Development Workspaces.** Published in 2004. Pages: 62-64. Mangan MAS, Borges MRS, Werner CML.
PS36	**Collaboration practices in global inter-organizational software development projects.** Software Process: Improvement and Practice 8(4). Published in 2003. Pages: 183-199. Paasivaara, M, Lassenius C.
PS37	**Defect detection oriented lifecycle modeling in complex product development.** Information and Software Technology 46(10). Published in 2004. Pages: 665-675. Moll Jv, Jacobs J, Kusters R, Trienekens J.
PS38	**Process and technology challenges in swift-starting virtual teams.** Information & Management 44(3). Published in 2007. Pages: 287-299. Munkvold BE, Zigurs I.
PS39	**Architecture as a coordination tool in multi-site software development**. Software Process: Improvement and Practice 8(4). Published in 2003. Pages: 233-247. Ovaska, P, Rossi M, Marttiin P.

PS40	**Software configuration management over a global software development environment: lessons learned from a case study.** Proceedings of the 2006 international workshop on Global software development for the practitioner. Published in 2006. Pages: 45-50. Pilatti L, Audy JLN, Prikladnicki R.
PS41	**Virtual teams: a review of current literature and directions for future research.** SIGMIS Database 35(1). Published in 2004. Pages: 6-36. Powell A, Piccoli G, Ives B.
PS42	**Global software development in practice lessons learned. Software Process**: Improvement and Practice 8(4). Published in 2003. Pages: 267-281. Prikladnicki, R, Audy JLN, Evaristo R.
PS43	**Leveraging Global Resources: A Process Maturity Framework for Managing Distributed Development.** IEEE Software 22(3). Published in 2005. Pages: 80-86. Ramasubbu N, M. S. Krishnan, Kompalli P.
PS44	**Can distributed software development be agile?** Communications of the ACM 49(10). Published in 2006. Pages: 41-46. Ramesh B, Cao LAN, Mohan K, Peng XU.
PS45	**The role of collaborative support to promote participation and commitment in software development teams.** Software Process: Improvement and Practice 12(3). Published in 2007. Pages: 229-246. Renata Mendes de Araujo MRSB.
PS46	**Virtual workgroups in offshore systems development.** Information and Software Technology 47(5). Published in 2005. Pages: 305-318. Sakthivel S.
PS47	**An experimental simulation of multi-site software development.** Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research. Published in 2004. Pages: 255-266. Shami NS, Bos N, Wright Z, Hoch S, Kuan KY, Olson J, Olson G.
PS48	**Measuring and Tracking Distributed Software Development Projects.** Published in 2003. Pages: 63-69. Simmons DB.
PS49	**A research agenda for distributed software development.** Published in 2006. Pages: 731-740. Sinha V, Chandra S, Sengupta B.
PS50	**Using simulation to evaluate global software development task allocation strategies**. Software Process: Improvement and Practice. Published in 2007. Pages: n/a. Setamanit, S, Wakeland W, Raffo D.
PS51	**Leveraging Distributed Software Development.** 32(9). Published in 1999. Pages: 59-64. Suzuki J, Yamamoto Y.
PS52	**A flexible framework for cooperative distributed software development.** Journal of Systems and Software 16(2). Published in 1991. Pages: 97-105. Narayanaswamy. K, Goldman, NM.
PS53	**A reliability assessment tool for distributed software development environment based on Java and J/Link.** European Journal of Operational Research 175(1). Published in 2006. Pages: 435-445. Tamura Y, Yamada S, Kimura M.
PS54	**An integration centric approach for the coordination of distributed software development projects.** Information and Software Technology 48(9). Published in 2006. Pages: 767-780. Taxen L.
PS55	**Communication tools for distributed software development teams.** Proceedings of the 2007 ACM SIGMIS CPR conference on 2007 computer personnel doctoral consortium and research conference: The global information technology workforce. Published in 2007. Pages: 28-35. Thissen MR, Page JM, Bharathi MC, Austin TL.
PS56	**Ontology-based multi-agent system to multi-site software development.** Proceedings of the 2004 workshop on Quantitative techniques for software agile process. Published in 2004. Pages: 66-75. Wongthongtham P, Chang E, Dillon TS.
PS57	**Ontology-based multi-site software development methodology and tools.** Journal of Systems Architecture 52(11). Published in 2006. Pages: 640-653. Wongthongtham P, Chang E, Dillon TS, Sommerville I.

PS58	**Supporting Distributed Software Development Processes in a Web-Based Environment.** Published in 1999. Pages: 292-295. Yang Y, Wojcieszak P.
PS59	**Project Management Model: Proposal for Performance in a Physically Distributed Software Development Environment.** Engineering Management Journal 16(2). Published in 2004. Pages: 28-34. Zanoni R, Audy JLN.
PS60	**Knowledge flow management for distributed team software development.** Knowledge-Based Systems 15(8). Published in 2002. Pages: 465-471. Zhuge H.
PS61	**Empirical evaluation of distributed pair programming.** International Journal of Human-Computer Studies, In Press. Accepted Manuscript 2007. Hanks B.
PS62	**Enhancing collaborative synchronous UML modelling with fine-grained versioning of software artefacts.** Journal of Visual Languages & Computing 2007 18(5). Published in 2007. Pages: 492-503. De Lucia A., Fasano F., Scanniello G., Tortora G.
PS63	**An Evaluation Method for Requirements Engineering Approaches in Distributed Software Development Projects.** International Conference on Software Engineering Advances (ICSEA 2007). Published in 2007. Pages: 39-45. Michael G., Tobias H., Franz R., Colin A.
PS64	**1st International Workshop on Tools for Managing Globally Distributed Software Development (TOMAG 2007).** International Conference on Software Engineering Advances (ICSEA 2007). Published in 2007. Pages: 278-279. Chintan A., Jos van H., Frank H.
PS65	**Distributed Software Development: Practices and challenges in different business strategies of offshoring and onshoring.** Published in 2007. Pages: 262-274. Rafael P., Jorge Luis N. A., Daniela D., Toacy C. d. O.
PS66	**An Adaptive Tool Integration Framework to Enable Coordination in Distributed Software Development.** International Conference on Software Engineering Advances (ICSEA 2007). Published in 2007. Pages: 151-155. Vibha S. S., Bikram S., Sugata G.
PS67	**Coordination Practices in Distributed Software Development of Small Enterprises.** International Conference on Software Engineering Advances (ICSEA 2007). Published in 2007. Pages: 235-246. Alexander B., Bernhard N., Volker W.
PS68	**Globally distributed software development project performance: an empirical analysis.** Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering 2007. Published in 2007. Pages: 125-134. Narayan R., Rajesh Krishna B.
PS69	**On-line collaborative software development via wiki.** Proceedings of the 2007 international symposium on Wikis 2007. Published in 2007. Pages: 177-183. WenPeng X., ChangYan C., Min Y.

Author Index