Distributed Information System Development: Review of Some Management Issues

Deepti Mishra and Alok Mishra

Department of Computer Engineering
Atilim University, Ankara, Turkey
deepti@atilim.edu.tr, alok@atilim.edu.tr

Abstract. Due to the proliferation of the Internet and globalization, distributed information system development is becoming popular. In this paper we have reviewed some significant management issues like process management, project management, requirements management and knowledge management issues which have received much attention in distributed development perspective. In this literature review we found that areas like quality and risk management issues could get only scant attention in distributed information system development.

Keywords: Global software development, Distributed information system development, Distributed Software Development.

1 Introduction

More and more organizations have distributed their software development projects in different sites around the globe. As the software community appreciates, the economy of merging diverse development skills and domain expertise, and as communication media become more sophisticated, the cost and technology pushes more companies toward global software development [45]. With the need of reduced software product development cost and time and improved software quality, software products are increasingly developed via collaborations across people, organizations, and countries [5][56]. The accounts about cheaper work and "follow the sun" approaches are fading, while factors like proximity to the markets, access to specific expertise, productive friction and innovation capability tend to take the lead in driving the trend toward global software development [2]. This change is having a profound impact on the way products are conceived, designed, constructed, tested and delivered to customers [45][27][13]. Thus, the structure needed to support this kind of development is different from the ones used in collocated environments [45]. Therefore it is important to take into account this evolving paradigm of distributed information system development from different perspectives.

It is significant to note that although software implementation tasks may be moved from countries with high labor costs to countries with low cost relatively easily due to the Internet and common programming environments, tasks requiring intensive customer interaction , such as requirement analysis and specification, software design,

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 282–291, 2009.

and verification and validation, are hard to migrate [34]. 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 lack of informal communication [10].

Due to many conflicts, global software development projects are often more complex to manage. Engineers, managers and executives face formidable challenges on many levels, from the technical to the social and cultural [22]. Environments and processes in typical software development are not fully adapted to the needs of global software development [41]. In particular, they do not have all the capabilities necessary for cross-site collaboration [41].

In their recent report, Smite et al. [50] concluded that the state of practice in the GSE (Global Software Engineering) area is still unclear and evolving. There are very few review studies [50][24] in literature. Although systematic literature reviews provide a summary of research and references but still lack in presenting coherent and similar views on emerging sub areas. After an extensive literature survey, we found that knowledge management, process management, project management and requirements management have received much attention among researchers while studies on risk management, quality management, strategic management issues are very limited in distributed information system development. Jimenez et al. [24] have also supported in their recent study that maturity models such as CMMI or ISO represent only 17% of all analyzed works. Therefore this paper presents cohesive and recent trends in these areas which are getting much attention among researchers and also suggest those areas which require further research in software development in distributed environment.

The paper is organized as follows. The next section describes significant management issues based on literature review for distributed software development (Knowledge Management, Process Management, Project Management and Requirements Management). Finally, it concludes with discussions.

2 Management Issues in Distributed Information System Development

2.1 Knowledge Management

The knowledge-intensive nature of global software development efforts poses interesting challenges for organizations. Knowledge must be managed in all stages of software development from the encapsulation of design requirements, to program creation and testing, to software's installation and maintenance-and even extends to the improvement of organizational software development processes and practices [14]. More often, distributed projects require them to coordinate and integrate multiple knowledge sources, often under severe time pressure and resource and budgetary constraints. For workers to succeed in these engagements, the organization must have a robust knowledge management program [14]. Knowledge management issues becomes exponentially salient in the context of distributed and global software development efforts as knowledge is spread across locations and coordinating and synthesizing this knowledge is challenging [15]. A recurring theme in distributed

software development is that informal communication is critical for rapidly disseminating implicit knowledge, in particular, with respect to change and unforeseen events [8] which increases dramatically as the size and complexity of the software increases [31]. Without effective information and knowledge-sharing mechanisms, managers cannot exploit GSDs benefits. For instance, they might fail to promptly and uniformly share information from customers and the market among the development teams [22]. Common knowledge management problems in global software efforts include the inability to seek out relevant knowledge, poor aggregation and integration procedures for knowledge synthesis, and delays or blockages of knowledge transfer [14]. Unless organizations manage knowledge and leverage it appropriately in global software development efforts, the vitality of such efforts is compromised, and they become a burden rather than a competitive advantage [14]. Desouza and Evaristo [15] found three strategies that are currently in use for setting up knowledge management programs:

- In a headquartered-commissioned-and-executed strategy
- In a headquarters-commissioned-and regionally-executed strategy
- A regionally-commissioned-and-locally-executed strategy

Desouza et al. [14] mentioned that once the strategy is adopted it is important to setup a knowledge management system (KMSs) which can be from the client-server model, the peer-to-peer model, and the hybrid model. In their recent paper, Lopez et al. [35] suggested that in the Knowledge Management and Awareness area more research should be done, as 100% of its safeguards are proposed. Studies focusing on global software team performance point to the importance of knowledge sharing in building trust and improving effectiveness, while recognizing the additional complexity in sharing knowledge across geographically dispersed sites, not least because of the tacit dimension of so much knowledge [29][43]. Despite the fact that most of the problems reported in global software projects are fundamentally to do with information and knowledge, past research not focused on the role of knowledge sharing and social aspects of global software projects [30]. They found in their studies and suggested that managers should consider their organization in terms of knowledge processes, not just information flows and must focus on how coordination mechanisms facilitate knowledge processes. They argued that coordination mechanisms become knowledge management instruments, with a focus on their contribution to the coherence of knowledge processes and activities in achieving a coordinated outcome. Kotlarsky et al. [30] proposed a research model on coordination for a knowledge-based perspective and concluded that technologies are most useful for amplifying knowledge management processes to allow knowledge sharing while organization design facilitates knowledge flows across organizations and teams. Work-based mechanisms make knowledge explicit and accessible, while social mechanisms are needed to build social capital and exchange knowledge and ideas. 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 [24]. To overcome drawbacks caused by distribution, Babar [3] proposes the application of an electronic workspace paradigm to capture and share metadata information among several tools. Zhuge [57] introduced 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. Mohan and Ramesh [36] provided 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 fragements stored in diverse environment to be integrated and used by various stakeholders in order to facilitate a common understanding.

2.2 Process Management

The selection of the right development approach is necessary to address issues such as lack of users' knowledge of application, lack of users' involvement, missing, incorrect and evolving requirements, and risks associated with new technology and poor quality [7]. The software development process is considered one of the most important success factors for distributed projects [45].

Unclear requirements and new technologies make the waterfall model unsuitable for offshore developing strategic systems [47]. Although the use of a spiral model is uncommon in the development of business information systems, the strategic importance of a system warrants detailed planning, experimentation, verification, validation and risk management provided by the spiral model [47].

According to Simons [49], an iterative model seems to work well in distributed environment and can eliminate some of the problems that distribution brings. Moreover, iterative development with frequent deliveries provides increased visibility into project status, which makes it easier for project managers and customers to follow project progress [49]. Fowler [18] and Simons [49] emphasize that the main benefits of using agile development in their projects have been the high responsiveness to change and fast delivery of business value and these benefits have outweighed the challenges of distributed development.

Layman et al. [33] conducted an industrial case study to understand how a globally distributed team created a successful project in a new problem domain using eXtreme Programming (XP) that is dependent on informal face-to-face communication. They found that many of the hallmarks of the face-to-face communication advocated in XP are adquately approximated through the team's use of three essential communication practices: an email listserv, globally-available project management tool and an intermediary development manager who played a strong role in both groups. Nisar and Hameed [42] and Xiaohu et al. [54] describe their experiences in using XP in offshore teams collaborating with onshore customers. Both papers discuss projects where the development work is done in offshore teams only but the onshore customer is tightly involved in project communication. They concluded that the reported projects have been very successful and XP principles they have followed have proven to work [42][54]. Karlsson [26] and Farmer [17] found the XP practices useful but hard to implement in distributed projects. Sangwan and Laplante [48] reported that geographically distributed development teams in large projects can realize Test Driven Development's (TDD) considerable advantages. With good communication and judicious use of automated testing, they can overcome many problems [48]. The transition from unit to system level testing is challenging for TDD, as in general TDD

is not intended to address system and integration testing – certainly not for globally distributed development teams at any rate [48]. Still developers can realize the advantages of TDD through increased informal and formal communication, facilitated by appropriate change management and notification tools [48]. Test frameworks like the Xunit family are also appropriate at the systems level if developers are sufficiently disciplined in developing test cases [48]. Finally, the use of commercial or open source test workbenches and bug repositories to track test case dependencies and test failures is essential [48]. The maturity of the process becomes a key success factor. Passivaara and Lassenius proposed incremental integration and frequent deliveries by following informing and monitoring practices [44]. Recently, SoftFab tool infrastructure which enables projects to automate the building and test process and which manages all the tasks remotely by a control center was given by Spanjers et al. [51]. 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 [24]. There is still scope towards defining the process framework and maturity level standards like CMMI, SPICE etc. for distributed software development towards quality.

2.3 Project Management

Global software development literature brings up many challenges related to distributed development, e.g., interdependencies among work items to be distributed, difficulties of coordination, difficulties of dividing the work into modules that could be assigned to different locations, conflicting implicit assumptions that are not noticed as fast as in collocated work and communication challenges [40]. Managing a global virtual team is more complex than managing a local virtual team [39]. Effort estimation in software development is a major problem of project management [25]. Currently available tools may not provide an accurate estimation of development efforts in virtual environment [47].

It has been reported that multi-site software development introduces additional delay and thus takes much longer than single-site development [21]. However, the projects studied do not seem to have utilised the time difference between sites and thus have not exploited an ‘around the clock’ work pattern [52]. The global software development model opens up the possibility of 24-hour software development by effectively utilizing the time zone differences [23]. To harness the potential of the 24-hour software development model for reducing the overall development time, a key issue is the allocation of project tasks to the resources in the distributed teams [23]. The project tasks have vaious constraints for its execution: operational constraints, skill constraints, resource constraints etc. [23]. In some projects there may be some tasks that have collocation requirements i.e. engineering working on these tasks should be together [16]. Clearly, to maximize the reduction in completion time through the 24 hour model, task assignment needs to be done carefully, so that the project can be completed in minimum execution time while satisfying all the constraints [23].

Literature suggests dividing the work into separate modules that then can be distributed to different sites to be developed [20]. These modules should be

independent in order to minimize communication between sites [20]. Frequent deliveries are also recommended as a successful GSD collaboration practice [44].

The project manager has a greater role to co-ordinate, communicate, facilitate the group process and ensure participation of all members [4]. They act as mentors, exhibit empathy, are assertive without being overbearing, articulate role relationships, build relationships and trust and provide detailed and regular communications with the group members [28]. In addition, they define the interaction protocol for guiding the group behavior in virtual environment [9] and ensure prompt interactions in asynchronous communications [47].

Layman et al. [33] suggested that high process visibility is important for the customers to guide the project effectively while the developers are working on a new and unfamiliar problem. A major study of a global virtual group has many recommendations for an effective virtual group process [38]. The effectiveness depends on a match between the message complexity in the group process and the form of interaction medium [47]. Increased task interdependence in a group process will increase the interaction frequency, and increased task complexity will increase the messge complexity, which in turn, affects the choice of the interaction medium [47]. Computer-mediated asynchronous communication is suitable for structured and well-defined tasks [37][38] such as coding according to system specifications. Such type of global software development tasks still needs early face-to-face communication to define team members roles and to enhance the effectiveness of later computer-mediated communications [32].

2.4 Requirement Management

Most requirements are difficult to identify and identified requirements are unclear and not well organized [13]. These problems are exacerbated in global software development [13]. A key issue to achieve success in GSD is overcoming the pitfalls which GSD poses to RE [35]. Cheng and Atlee [11] recently identified globalization as a hot research topic regarding Requirements Engineering (RE). This new paradigm implies that many of the RE activities have to undergo changes, as it is no longer possible to carry them out as if the participants were collocated.

Initial studies show that distributed quality assurance of software code is feasible [6][19], but distributed quality assurance of requirements definition that need high user involvement would have a high task coupling until requirement definitions are formalized and made user-friendly [47].

During software requirement elicitation, lack of fluent communication is one of the most important challenges in discussion [55]. Therefore various factors affecting communication should be considered when analyzing virtual teams working on requirements definition, minimizing communication problems towards ensuring quality of the software in construction [1].

As requirements management activities are primarily impacted by distance [12], GSD projects benefit from requirements specifications well defined at the onset of the project [53], thus avoiding the need to reiterate feature understanding. These specifications are often used in planning processes, important in the organization and managing of distributed projects [46]. Layman et al [33] suggested that an identifiable customer authority is necessary to manage and resolve requirements related issues in a globally distributed team using XP.

3 Discussions and Conclusion

In this paper, we have reviewed some significant management issues like process and project management, requirements management and knowledge management issues from a global/distributed software/information system development perspective. Distributed software development is an evolving area. It is important to know the latest development through rigorous literature review to learn further new knowledge in different significant dimensions. As a future work, we would like to extend this review on various other attributes, dimensions and comparisons. Further we would like to include that area which could get only scant attention in distributed software development, for instance quality, and risk management issues. Various case studies may be included to enrich these management perspectives related with distributed information system development. However it is interesting to note as observed by Smite et al. [50] that these studies present results from significantly different contexts and backgrounds and may not applicable as a general standard in all contexts. They further viewed that unclarities inherited in reported studies may not only burden the reader's understanding, but also drive "lessons learned" applied in to be wrong way. Therefore there is a need to build the body of knowledge on how to manage global software development projects which will classify experiences and practices that will help to achieve positive results, in order to understand circumstances and also contexts.

References

1. Aranda, G.N., Vizcaino, A., Cechich, A., Piattini, M.: Technology Selection to Improve Global Collaboration. In: Proceedings of the IEEE international Conference on Global Software Engineering, ICGSE, October 16-19, pp. 223–232. IEEE Computer Society, Washington (2006)
2. Avram, G.: Of Deadlocks and Peopleware - Collaborative Work Practices in Global Software Development. In: Proceedings of the international Conference on Global Software Engineering, ICGSE, August 27- 30, pp. 91–102. IEEE Computer Society, Washington (2007)
3. Babar, M.A.: 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 2008, Leipzig, Germany, May 13, pp. 45–48. ACM, New York (2008)
4. Bal, J., Foster, P.: Managing the virtual team and controlling effectiveness. International Journal of Production Research 38(17), 4019–4032 (2000)
5. Bass, M., Paulish, D.: Global software development process research at Siemens. In: Proc. of Intl. workshop on global software development, Edinburgh, Scotland
6. Bianchi, A., Caivano, D., Lanubile, F., Visaggio, G.: Defect Detection in a Distributed Software Maintenance Project. In: Proc. of the International Workshop on Global Software Development (GSD 2003), Portland, Oregon, USA, May 2003, pp. 48–52 (2003)
7. Boehm, B.W.: Software risk management. IEEE computer society Press, New York (1989)
8. Bruegge, B., Dutoit, A.H., Wolf, T.: Sysiphus: Enabling informal collaboration in global software development. In: Proceedings of the IEEE international Conference on Global Software Engineering, ICGSE, October 16-19, pp. 139–148. IEEE Computer Society, Washington (2006)

9. Cascio, W.: Managing a virtual workplace. Academy of Management Executive 14(3), 81–90 (2000)
10. Casey, V., Richardson, I.: Project Management Within Virtual Software Teams. In: 2006 IEEE International Conference on Global Software Engineering, ICGSE 2006. Proceedings, Florianopolis, Brazil, October 16-19, pp. 33–42 (2006)
11. Cheng, B.H., Atlee, J.M.: Research directions in requirements engineering. In: Future of Software Engineering, FOSE 2007, pp. 285–303 (2007)
12. Damian, D.E., Zowghi, D.: RE challenges in multi-site software development organisations. Requirements Engineering Journal 8, 149–160 (2003)
13. Damian, D.E., Zowghi, D.: The Impact of Stakeholders? Geographical Distribution on Managing Requirements in a Multi-Site Organization. In: Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering, RE, September 9-13, pp. 319–330. IEEE Computer Society, Washington (2002)
14. Desouza Kevin, C., Awazu, Y., Baloh, P.: Managing Knowledge in Global Software Development Efforts: Issues and Practices. IEEE Software 23(5), 30–37 (2006)
15. Desouza, Evaristo: Managing Knowledge in distributed projects. Communications of the ACM 47(4), 87–91 (2004)
16. Ebert, C., De Neve, P.: Surviving Global Software Development. IEEE Softw. 18(2), 62–69 (2001)
17. Farmer, M.: DecisionSpace Infrastructure: Agile Development in a Large, Distributed Team. In: Proceedings of the Agile Development Conference, ADC, June 22-26, pp. 95–99. IEEE Computer Society, Washington (2004)
18. Fowler, M.: Using agile software process with offshore development, http://martinfowler.com/articles/agileOffshore.html
19. Hedberg, H., Harjumaa, L.: Virtual Software Inspections for Distributed Software Engineering Projects. In: Proceedings of International Workshop on Global Software Development (Co-located with ICSE 2002), Orlando, Florida (2002)
20. Herbsleb, J.D., Grinter, R.E.: Architectures, Coordination, and Distance: Conway's Law and Beyond. IEEE Softw. 16(5), 63–70 (1999)
21. Herbsleb, J.D., Mockus, A., Finholt, T.A., Grinter, R.E.: An empirical study of global software development: distance and speed. In: Proceedings of the 23rd international Conference on Software Engineering, Toronto, Ontario, Canada, May 12-19, pp. 81–90. IEEE Computer Society, Washington (2001)
22. Herbsleb, J.D., Moitra, D.: Guest Editors' Introduction: Global Software Development. IEEE Softw. 18(2), 16–20 (2001)
23. Jalote, P., Jain, G.: Assigning Tasks in a 24-Hour Software Development Model. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference, APSEC, November 30 - December 3, pp. 309–315. IEEE Computer Society, Washington (2004)
24. Jimenez, M., Piattini, M., Vizcaino, A.: Challenges and Improvements in Distributed Software Development: A Systematic Review. Advances in Software Engineering 2009, 1–14 (2009)
25. Jones, C.: Why software fails. Software Development 4(1), 49–54 (1996)
26. Karlsson, E., Andersson, L., Leion, P.: Daily build and feature development in large distributed projects. In: Proceedings of the 22nd international Conference on Software Engineering, ICSE 2000, Limerick, Ireland, June 4-11, pp. 649–658. ACM, New York (2000)
27. Karolak, D.W.: Global software development managing virtual teams and environments. IEEE Computer Society, Los Alamitos (1998)

28. Kayworth, T.R., Leidner, D.E.: Leadership Effectiveness in Global Virtual Teams. J. Manage. Inf. Syst. 18(3), 7–40 (2002)
29. Kotlarsky, J., Oshri, I.: Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. Eur. J. Inf. Syst. 14(1), 37–48 (2005)
30. Kotlarsky, J., Van Fenema, P.C., Willcocks, L.P.: Developing a knowledge-based perspective on coordination: the case of global software projects. Information and management 45(2), 96–108 (2008)
31. Kraut, R.E., Streeter, L.A.: Coordination in software development. Commun. ACM 38(3), 69–81 (1995)
32. Kruempel, K.: Making the right (interactive) moves for knowledge-producing tasks in computer-mediated groups. IEEE Transactions on Professional Communication 43, 185–195 (2000)
33. Layman, L., Williams, L., Damian, D., Bures, H.: Essential communication practices for extreme programming in a global software development team. Information and Software Technology 48(9), 781–794 (2006)
34. Liu, X.: (Frank) Collaborative Global Software Development and Education. In: Proceedings of the 29th Annual International Computer Software and Applications Conference, COMPSAC 2005 (2005)
35. Lopez, A., Nicolas, J., Toval, A.: Risks and Safeguards for the Requirements Engineering Process in Global Software Development. In: 4th International Conference on Global Software Engineering. IEEE Press, Los Alamitos (2009)
36. Mohan, K., Ramesh, B.: Traceability-based Knowledge Integration in group decision and negotiation activities. Decision Support Systems 43(3), 968–989 (2007)
37. Majchzak, A., Rice, R.E., King, N., Malhotra, A., Ba, S.: Computer-mediated inter-organizational knowledge-sharing: Insights from a virtual team innovating using a collaborative tool. Inf. Resour. Manage. J. 13(1), 44–53 (2000)
38. Maznevski, M.L., Chudoba, K.M.: Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness. Organization Science 11(5), 473–492 (2000)
39. McDonough, E.F., Kahn, K.B., Barczak, G.: An investigation of the use of global, virtual, and collocated new product development teams. Journal of Product Innovation Management 18(2), 110–120 (2001)
40. Mockus, A., Herbsleb, J.: Challenges of global software development. In: Software Metrics Symposium, METRICS 2001. Proceedings. Seventh International, pp. 182–184 (2001)
41. Mullick, N., Bass, M., Houda, Z., Paulish, P., Cataldo, M.: Siemens Global Studio Project: Experiences Adopting an Integrated GSD Infrastructure. In: Proceedings of the IEEE international Conference on Global Software Engineering, ICGSE, October 16-19, pp. 203–212. IEEE Computer Society, Washington (2006)
42. Nisar, M.F., Hameed, T.: Agile methods handling offshore software development issues. In: Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International, pp. 417–422 (2004)
43. Orlikowski, W.J.: Knowing in Practice: Enacting a Collective. Capability in Distributed Organizing 13(3), 249–273 (2002)
44. Paasivaara, M., Lassenius, C.: Collaboration practices in global inter-organizational software development projects. Software Process: Improvement and Practice 8(4), 183–199 (2004)
45. Prikladnicki, R., Audy, J.L.N., Evaristo, R.: A Reference Model for Global Software Development: Findings from a Case Study. In: Proc. IEEE Int'l Conference on Global Software Engineering (ICGSE 2006), Florianópolis, Brazil, pp. 18–25. IEEE Computer Society Press, Los Alamitos (2006)

46. Prikladnicki, R., Audy, J.L.N., Evaristo, R.: Global software development in practice lessons learned. Software Process: Improvement and Practice 8(4), 267–281 (2003)
47. Sakthivel, S.: Virtual workgroups in offshore systems development. Information and Software Technology 47(5), 305–318 (2005)
48. Sangwan, R.S., LaPlante, P.A.: Test-Driven Development in Large Projects. IT Professional 8(5), 25–29 (2006)
49. Simons, M.: Internationally Agile. InformIT (March 15, 2002)
50. Smite, D., Wohlin, C., Feldt, R., Gorschek, T.: Reporting Empirical Research in Global Software Engineering: A Classification Scheme. In: 2008 IEEE International Conference on Global Software Engineering, ICGSE, pp. 173–181 (2008)
51. Spanjers, H., ter Huurne, M., Graaf, M., Lormans, M., Bendas, D., van Solingen, R.: Tool support for distributed software engineering. In: Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE 2006), Florianopolis, Brazil, October 2006, pp. 187–198 (2006)
52. Taweel, A., Brereton, P.: Modelling software development across time zones. Information and Software Technology 48(1), 1–11 (2006)
53. Tiwana, A.: Beyond the Black Box: Knowledge Overlaps in Software Outsourcing. IEEE Softw. 21(5), 51–58 (2004)
54. Xiaohu, Y., Bin, X., Zhijun, H., Maddineni, S.R.: Extreme programming in global software development. In: Canadian Conference on Electrical and Computer Engineering, 2004, vol. 4, pp. 1845–1848 (2004)
55. Young, R.: Recommended Requirements Gathering Practices. CROSSTALK The Journal of Defence Software Engineering, 9–12 (2002)
56. Zamiska, N.: Quality lures software outsourcing. The Wall Street Journal (May 5, 2005)
57. Zhuge, H.: Knowledge flow management for distributed team software development. Knowledge-Based Systems 15(8), 465–471 (2002)