Architecting in Global Software Engineering

Bedir Tekinerdogan
Bilkent University, Turkey
bedir@cs.bilkent.edu.tr

Semih Cetin
Cybersoft, Turkey
semih.cetin@cs.com.tr

Muhammad Ali Babar
IT University of Copenhagen, Denmark
malibaba@itu.dk

Patricia Lago
VU University Amsterdam, The Netherlands
p.lago@vu.nl

Juho Mäkiö
Heilbronn University, Germany
juho.maekioe@hs-heilbronn.de

Abstract

This paper summarizes the results of the First Workshop on Architecting in Global Software Engineering (GSE), which was organized in conjunction with the 6th International Conference on Global Software Engineering (ICGSE 2011). The workshop aimed to bring together researchers and practitioners for defining and advancing the state-of-the-art and state-of-the practice in architecture design of global software development systems.

Keywords: Global Software Engineering, Software Architecture, Workshop

1. Introduction

Current trends in software engineering show that large software projects have to operate with teams that are working in globally distributedlocations. The reason behind this globalization of software development stems from clear business goals such as reducing cost of development, solving local IT skills shortage, and supporting outsourcing and offshoring [1]. There is ample reason that these factors will be even stronger in the future, and as such we will face a further globalization of software development [8]. To cope with these problems, we have witness the emergence of Global Software Engineering (GSE) paradigm [10]. GSE is a relatively new paradigm of software development that can be considered as the coordinated activity of software development that is not localized and central but geographically distant. Figure 1 shows the conceptual architecture for GSE systems. A GSE architecture usually consists of several nodes, or sites, on which different teams are working to develop a part of a system. The teams could include development teams, testing team, and management team. Usually each site will also be responsible for following a particular process. In addition, each site might have its own local data storage.

Despite its envisaged benefits, GSE is not a trivial undertaking and has to cope with different challenges in different domains including software architecture, eliciting and communicating requirements, setting up suitable environments and tools, and orchestration of GSE [10]. A close analysis of the literature in GSE shows that little attention has been paid on the software architecting process and software architecture as an artifact in the context of GSE. As a consequence of this situation, the problems related to architecting the large and complex systems required in GSE have not been explicitly addressed.

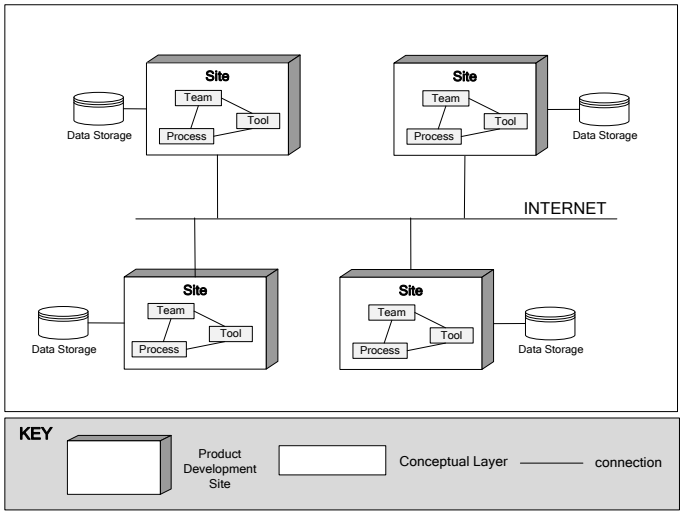

Figure 1. Conceptual Architecture for GSE[29]

Yet, it is generally accepted that software architecture design plays a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software. Since GSE projects very often have to deal with large systems, software architecture seems to be even more important for GSE.

Research on architecture design in the last two decades has resulted in different useful techniques and approaches. Different architectural modeling approaches for representing multiple views of the architecture have been proposed. Multiple architectural patterns have been introduced in literature to support the quality of architecture. A broad range of architecture analysis approaches have also been proposed to analyze the architecture before it is implemented. Despite the significant research and development output from the software architecture design community, we can observe that the endeavor of software architecting seems to have been mainly focused on architecting systems to be mainly developed and evolved by following the pre-GSE era.

Considering the two domains of global software engineering and software architecture design, we can identify different but related research perspectives:

(1) How can software architecture be used to support GSE?

It is well-known that software architecture design plays an essen-

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900

tial role in coping with the complexities of software systems, but we are interested, in particular in the way software architecture can deal with the specific GSE concerns.

(2) How does GSE impact software architecture?

Distributed development as defined in GSE together with its specific concerns will have an impact on the software architecture.

The viewpoints introduced above have been used to structure the first workshop on Architecting in Global Software Engineering (AGSE 2011). The workshop aimed to bring together researchers and practitioners for defining and advancing the state-of-the-art and state-of-the practices in architecture design of global software development systems. In this paper, we report on the results of this workshop and define an outline for future research.

The remainder of the paper is organized as follows. In Section 2, we define the workshop topics. Section 3 defines the workshop organization plan and activities. The subsequent sections report on the results of the discussions during the workshop including, the motivation for software architecture GSE, challenges for architecting in GSE, and requirements for proper GSE architecture.

2. Conceptual Model

Figure 2 shows, a conceptual model representing the relation between architecture and GSE. The rectangles represent the concepts, the arrows represent conceptual relations.

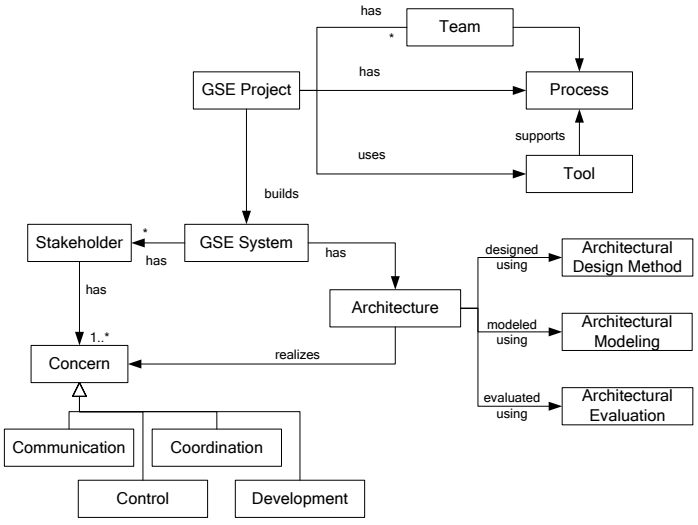

Figure 2. Conceptual Model defining relation between architecture and GSE

GSE Project consists of *Team*, *Process* and *Tool*. *GSE Project* develops a *GSE System*, that has an *Architecture*. *GSE System* has a set of stakeholders who have a stake in the system, and which have *Concerns*. The key concerns in GSE are the following [14][15]:

Development - the software development activities typically using a software development process. This includes activities such as requirements analysis, design, implementation and testing. Each site will address typically a subset of these activities.

Communication – the communication mechanisms within and across sites. Typically the different sites need to adopt a common communication protocol to support distributed development.

Coordination – coordination of the activities within and across sites to develop the software according to the requirements. Coordination will be necessary to align the workflows and schedules of the different sites. An important goal could be to optimize the development using appropriate coordination mechanisms.

Control – systematic control mechanisms for analyzing, monitoring and guiding the development activities. This does not only include controlling whether the functional requirements are performed but also which and to what extent quality requirements are addressed.

Architecture is designed using an *Architecture Design Method*, modeled using *Architectural Modeling* approach, and evaluated using *Architectural Evaluation.*

3. Workshop Topics and Activities

AGSE solicited submissions dealing with topics in the following list:

- Software Architecture Modeling of GSE
- Modeling Software Architectures for GSE
- Software Architecture Viewpoints for GSE
- Software Architecture Description Languages for GSE
- Software Architecture Patterns for GSE
- Documenting Software Architectures of GSE
- Architectural Requirements Analysis of GSE
- Analysis and evaluation of Software Architectures of GSE
- Quality Models for Software Architectures of GSE
- Tools for Designing and Analyzing GSE
- Experiences in Architecting GSE
- Managing architectural independencies in GSE teams
- Role of architecture in GSE project and process governance
- Architectural knowledge as control mechanism
- Sharing and using distributed architectural knowledge
- Approaches for early stages of architecture design in GSE projects
- Architectural styles and patterns for supporting GSE teams
- Cultural influence on architectural processes and artefacts

AGSE has been organized as a very interactive event including ample time for lively discussions. The discussions were organized around the following three questions:

1. What is the rationale for software architecture in GSE?
2. What are the (additional) challenges for software architecture design in GSE?
3. What are the roles of an architect in GSE projects?

Each question was discussed in groups of two and later on during the plenary session. In the following sections we provide the results of the discussions for each of these questions.

4. Rationale for Software Architecture in GSE

The ISO/IEC/IEEE 42010 International Standard on Architecture Description of Software-Intensive Systems provides the following definition for software architecture [16]:

Architecture is the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles guiding its design and evolution.

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900

Different motivations have been given for software architecture in the literature [1]. In the following, we list the important points together with a discussion on the motivation for the importance of architecture in GSE.

- *Architecture supports communication among stakeholders*

Software architecture represents a common abstraction of a system that can be used as a basis for communication among the stakeholders. Stakeholders are all those people who need to be considered in achieving a project's goals and whose participation and support are crucial to the project's success. As such, the identification of all stakeholders is an important activity to ensure project success. The notion of *stakeholder* is defined as "an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system". Besides *local-stakeholders* in GSE, *global-stakeholders* must specify and manage requirements across cultural, time-zone, and organizational boundaries. This results in a more elaborate set of stakeholders than in the case of single site development. The list of stakeholders that we have defined is listed in TABLE 1.

- *Architecture defines early design decisions*

Software architecture is one of the earliest artifacts in the development life cycle and likewise represents the earliest set of design decisions about a system. In general, these early decisions have the largest impact and cannot be changed easily. The software architecture defines the structure and constraints on subsequent artefacts in the life cycle, including detailed design and code. At the architecture design level, it can be decided whether the system to be developed will be able to meet the selected quality concerns. In the context of GSE, it is important to provide a common medium for defining the design decisions that impact the system. Unlike local systems, the impact is even broader in the GSE environment and will have even a larger impact. Quality concerns need to be explicitly considered and realized in the architecture to mitigate risks in GSE projects.

- *Architecture shapes organization structure*

Architecture very often impacts also the structure of the organization. The high level decomposition that it provides can be used to divide the overall work into portions and assign these to different groups in the project. This so-called work breakdown structure on its turn mandates the units of planning, scheduling and budget, inter-team communication channels, configuration control and file system organization, and integration and test plans [1]. Regarding GSE, we could state that the management of the concerns for development, communication, coordination and control somehow are related to the proper architecture. Likewise the architecture plays an important role in GSE to define the work breakdown structures.

- *Architecture permits early analysis of quality concerns*

Since early design decisions regarding quality concerns are made at the architecture design level, the architecture design permits also the analysis of these quality concerns. Quality concerns such as performance, adaptability, scalability, and reuse can all be analyzed before the system is implemented. In GSE, besides these quality concerns, cost and schedule estimates are also important to control and guide the project. The total cost of a GSE project could be estimated using the architecture.

TABLE 1. STAKEHOLDERS FOR GLOBAL SOFTWARE DEVELOPMENT

Stakeholder	Concern
Customer	• requirements traceability • cost
Local Business Analyst	• proper interpretation of business rules and requirements for technical system • business process modeling
Global Business Analyst	• proper interpretation of business rules and requirements for technical system • business process modeling • Efficient global development
Local Requirements Engineering	• complete and consistent specification of functional and non-functional requirements on local site
Global Requirements Engineering	• complete and consistent specification of functional and non-functional requirements across sites
Local System Architect	• complete consistent architecture • requirements traceability • support for trade-off analysis
Global System Architect	• Overall architecture
Local Developer	• modularity and easy to develop of local system
Global Developer	• modularity and easy to develop of functionality across systems
Local Database Analyst	• mapping entities to data
Global Database Analyst	• Datamodeling across sites
Local Tester	• Testability of system locally
Global Tester	• Global testability
Local Maintainer	• compatibility with existing systems • adaptability of the system
Global Maintainer	• Consist global updates • Adaptability across sites
End-user	• functional requirements and performance
Local Manager	• how long it will take to build the product, how much it will cost, and what are the potential problems.
Global Manager	• Allocation of tasks over different sites

- *Architecture supports management of evolution*

Software systems are usually not fixed but evolve over their lifetimes. The change can be in different ways. An architectural change is usually systemic in nature and requires changes all over the system. An effective architecture anticipates on and provides mechanisms for change. In the GSE setting because of the larger scope and the broader differences of the components and sites, the change is likely to happen more often and in a more difficult way. Architecture can be helpful to control the change by anticipating the changes, analyzing the consequences of the required changes, prioritizing the requested changes, and executing the changes.

- *Gross-level reuse*

Software reuse is the use of existing software or software knowledge to construct new software [10]. Reusable assets are assets that can be reused in the development of new products. *Reusability* is a property of a software asset that indicates its probability of reuse. In general, the goal of software reuse is to improve software quality and productivity. Software reuse at the code level is beneficial, but reuse at the architectural level can provide more benefits and leverage existing effort tremendously. Reuse of

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900

software plays an important role in the context of software product line engineering. A software product line or family is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [5]. The software architecture is considered as the key asset that is designed to meet the needs of the entire product family. In GSE, the architecture can be reused for new GSE projects thereby leveraging the productivity and reducing cost of development. The GSE can be focused on single software development or a *product line* leading to the notion of global software product line engineering. Although different processes have been proposed they share the same concepts of *domain engineering*, in which a reusable platform and product line architecture is developed, and *application engineering*, in which the results of the domain engineering process are used to develop the product members.

5. Challenges for Software Architecture in GSE

As discussed in the previous section, there are obvious benefits for adopting software architecture based development in GSE. Yet, there are also a number of challenges which require attention to design architecture in GSE.

- *Communication problems*

Though architecture provides a common medium for communication about stakeholders, this is not trivial in a GSE context. Communication appears to be one of the major problems in GSE-projects. A number of challenges in GSE have been reported and studied in connection with communication problems (e.g., [6], [9] [10], [11], [12], and [13]).The opportunities to clarify ambiguities in work items are limited because informal face-to-face meetings are missing and the communication is usually more formal than in traditional software development projects. Additionally, the communication in distributed teams is negatively affected because it happens over different time-zones and over national borders. Consequently, the first challenge architecting GSE is the reduction of the communication needs stemming from communication failures with relation to tasks or assignment of tasks between teams. Hence, it is very important that architecting processes and architectural artifacts are designed to enable different teams work independently but collaboratively in order to reduce the need of having frequent and intensive communication that may characterize collocated software development arrangement.

- *Establishing and maintaining trust*

Architecture helps to guide the development process that requires the interaction of different teams needing to exchange ideas and artefacts frequently. Herewith, establishing and maintaining trust is an integral activity of a successful GSE team. The increase of trust between sites is a challenge for architecting in GSE as it defines at technical level the dependencies between the teams.

According to [16] and [17] trust is crucial for collaboration and is fundamental to coordination and cooperation. Lack of personal contacts and impersonal communication may cause distrust. Many GSE projects are likely to have a lack of trust among team members, especially when the members do not have previous experience of working with one another [18]. Lacking trust in GSE is analyzed in [19], [20] and [21]. One reason for the mistrust can be a weak cognitive trust – this is manifested in a situation when one site not trusting in the capability of the other site to fulfill the assigned responsibilities. On the one hand, one way to increase cognitive trust is to improve and deepen communication. As distrust is caused among others by bad experiences, angst and uncertainty [22], these aspects will need to be addressed in trust building. Bad experiences in GSE are done if one site does not deliver what another site needs to fulfill its tasks or if the other site is considered as ballast for the other site. One way of supporting trust building efforts may be having a well defined and easily understandable architecture and mechanisms to share knowledge underpinning architecture design; these kinds of acts are likely to reduce the interdependencies and the need of technical communication among team members; the private communication should be enabled.

- *Coordination of architecture design process*

Software architecture may help to guide the coordination and development process. Yet, in GSE projects this might not be trivial altogether. In the GSE, the working environment is distributed in the way that two or more teams located in various parts of the globe develop software together. The geographical distance between the teams introduces barriers and complexity affecting coordination and the visibility in the project and cooperation and the communication among the team members (cp. [15]). Also issues like, work distribution across sites; software development process; knowledge management and technical issues may complicate working in a GSEproject.

- *Architectural knowledge management*

GSE projects are often adversely affected by challenges such as lack of co-domain knowledge, incomplete requirements, communication, coordination, and collaboration. However, these challenges carry heavier penalty for GSE projects if not addressed appropriately than the collocated projects [7]. According to [14], the distributed work items can take up twice as much time to complete as the similar items in a collocated setting. Failures in GSE-projects are more difficult to improve than similar failures in traditional projects because of the higher communication and coordination effort. Further, according to [23], the problems in coding are among others: “understanding the rationale behind the code”, “understanding the code someone else wrote”, “being aware of changes to code elsewhere that impact my code” and “understanding the impact of changes I make on code elsewhere”. Well-documented architectural patterns and practices for architectural knowledge management [6] may be helpful for the developer searching answers for these questions. The required information may be delivered by clear and up to date architecture documentation, whereby keeping the documentation up to date is a challenge for architecting in GSE.

Other sources of potential problem in GSE are interdependencies among work items and difficulties in task coordination. Both of them are related to each other because interdependencies lead to higher coordination effort. The interdependencies occur when the architecture does not involve independent modules. Independent modules may be developed and tested in various locations by different teams with minimal communication needs.

- *Architectural modeling of GSE*

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900

A common practice for describing the architecture according to the stakeholders' concerns is to model different architectural views. An architectural view is a representation of a set of system elements and relations associated with them to support a particular concern. Usually multiple architectural views are needed to separate the concerns and as such support the modeling, understanding, communication and analysis of the software architecture for different stakeholders. Architectural views conform to *viewpoints* that represent the conventions for constructing and using a view. In general the existing architectural frameworks tend to be general purpose and not directly focused to a particular domain. The advantage of this is that it can be applied in a broad set of domains, but on the other hand the general-purpose architectural frameworks can fall short for modeling the particular concerns of specific domains. To cope with the specific concerns of GSE, an architectural framework including architectural viewpoints for modeling GSE architecture is needed [29].

To sum up, the practice of software development is an important challenge in GSE due to its communication and collaboration-intensive nature. As such, in general the traditional software development process activities such as requirements engineering, design, implementation and testing will be different in nature. Architecture has a central role in GSE. A proper architecture does not differ from a proper architecture in traditional software development projects, but the penalty of an inadequate architecture seems higher in GSE. Suitable software architecture may support bridging challenges of the GSE by reducing the negative effects caused by geographical distance.

Designing good architecture is challenging for any software system and not only in GSE. However, for the GSE project, the importance of having a good software architecture seems to be even more important than for the traditional software engineering projects. The architecture should take into account the specific situation and limitations of GSE, e.g., the challenges caused by the geographical, cultural, temporal, and linguistic differences. Hence, the main challenge in architecting GSE is to mitigate barriers caused by the geographical, cultural, temporal, linguistic distances and to minimize the communication needs among distributed teams.

6. Roles of a GSE Architect

In general terms, "software architect" is the role in a software development team or group of teams to ensure desired qualities while achieving the required functionality. The software architect is responsible for designing the most suitable structure for a software system or systems in a way that it meets the business and stakeholder requirements to achieve the desired results under a set of constraints.

Defining the roles of a software architect is still being researched within different perspectives such as technical, business and administrative issues [1][5][20]. However, the design of a software system in a distributed way has some key differences than collaboration in a face-to-face environment such as distribution and integration of development tasks. These variations bring in particular care for the design of software infrastructure for "separation" and "composition" of architectural concerns. Thus, we have rephrased the roles of a software architect for global software engineering as follows:

- *Abstract the complexity of a system*

GSE architect decomposes a complex software system into a more manageable model that describes the essence of a system by exposing important details and significant constraints in such a way that the basic building blocks can be implemented independently and later composed seamlessly.

- *Maintain control over the architecture lifecycle*

In parallel to the project's distributed software development lifecycle, GSE architect should be visible at every stage to proactively monitor the adherence of the distributed implementation to the chosen architecture during all iterations. Here, the proposed architecture should be capable of decoupling the implementation items assigned to independent distributed teams and integrating these items through proper middleware and / or frameworks in an iterative manner.

- *Stay on course in line with the long term vision*

GSE architect should be able to manage stakeholders in such a way that different distributed teams should be aligned to produce tangible implementation results as early and consistent as possible. When project variables outside of a distributed site control change the architect must adjust the strategy given the resource available while maintaining the long term goal.

- *Progressively make critical decisions*

GSE architect should take critical decisions that define a specific direction for a system in terms of distributed implementation, collaborative operations, and coordinated maintenance across time and location boundaries. The critical decisions must be carefully made and backed up by understanding and evaluation of alternative paths among many distributed teams. These decisions usually result in tradeoffs that principally define characteristics of a system. Additionally, these decisions must be well documented in a manner understood by others residing remote sites.

- *Set quantifiable objectives*

GSE architect should put measurable objectives that encapsulate quality attributes of a system developed in a distributed way. These objectives should be defined clearly and understood by every team in a global scale. Moreover, remotely developed components should be aligned with these predefined quantifiable objects.

- *Work closely with customer and executives*

The GSE architect should work very closely with the customer and executives to reflect the quality attributes to the software system where independent distributed teams will implement decoupled parts transparently according to the strategies set by the customer and executives located at a different place. This may be done by measuring the level of component / architecture re-use between distributed teams with the help from a common governance strategy. GSE architect must be effective in order to deliver results that are meaningful to distributed teams that have an impact on the bottom line that result in greater profits.

- *Inspire, mentor, and encourage colleagues*

The GSE architect applies intelligently customized industry's best practices in distributed software development. Educating the

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900

recipients and participants of system architecture is essential to successfully selling the chosen architectural path to independent development teams. Specifically the distant stakeholders must be able to understand, evaluate, and reason about software architecture within the same perspective. If GSE architect is the only one who can read and understand documented system architecture, then he has failed to integrate his best practices into the culture of a distributed organization.

- *Fight entropy*

Managing distributed teams threatens GSE architect's structural approach to problem solving. It is a GSE architect's job to keep the inertia across distributed teams. He or she must convince all relevant geographically distributed stakeholders that the chosen common approach is sound – moreover the chosen architectural solution must be well explained and justified by all remote parties.

- *Create and distribute tailored views of software architectures*

The GSE architect should instantly distribute tailored views of the software architecture to appropriate remote stakeholders at appropriate intervals. Moreover, every stakeholder in a distributed site should only know the expected level of architectural view, not more not less.

- *Act as an agent of change*

Managing the change among distributed development teams is highly important to ensure certain quality attributes in the final system. GSE architect should make distributed teams aware of changing design decisions at early stages. Particularly, if these design decisions end up with some infrastructural changes such as the use of new frameworks or libraries, a GSE architect should inform the relevant stakeholders at remote sites.

7. Conclusion

In this paper, we have reported on the different aspects of architecting in GSE. The ideas we presented were mainly derived from the First Workshop on Architecting in Global Software Engineering (AGSE). A key conclusion of the workshop is that specifically for GSE, it is important to adopt an architecture-based development approach. The software architecture provides important benefits such as support for communication among stakeholders, guiding the development process, guiding the organization process, and early analysis of the system; next to providing a reference for codification, architecture also offers a means for organizing personalization, i.e. communicating and sharing knowledge in a geographical distant setting. The issues requiring an architecture-based development approach also represent challenges that need to be solved. We have discussed several of these challenges that require more extensive research. We also plan to organize events related to the combination of software architecture and GSE in the near future.

Acknowledgments

The organizers of the workshop would like to thank the participants and the workshops' program committee.

References

[1] L. Bass, P. Clements, R. Kazman: Software Architecture in Practice, 2nd Edition. Addison Wesley, Reading, 2003.

[2] R.D. Battin, R. Crocker, J. Kreidler, K. Subramanian. Leveraging Resources in Global Software Development. IEEE Software, 18(2), p. 70-77, Mar/Apr, 2001.

[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M.Stal, Pattern-Oriented Software Architecture Volume 1 - A System of Patterns, Wiley, 1996.

[4] V. Casey, I. Richardson. Uncovering the Reality within Virtual Software Teams. Workshop on Global Software Development for the Practitioner, Shanghai, China, p. 66-72, 2006.

[5] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Second Edition. Addison-Wesley, 2010.

[6] V. Clerc, P. Lago, & H. van Vliet. The Usefulness of Architectural Knowledge Management Practices in GSD, Proc. Fourth IEEE International Conference on Global Software Engineering ICGSE 2009. pp. 73-82. 2009.

[7] V. Clerc. Architectural Knowledge Management in Global Software Development, VU University Amsterdam, PhD Thesis, Dec., 2011.

[8] D.E.Damian & D.Zowghi. RE challenges in multi-site software development organisations. Requirements Eng 8. p. 149–160, 2003

[9] D. L. Ferrin, M. C. Bligh and J. C. Kohles. It Takes Two to Tango: An InterdependenceAnalysis of the Spiraling of Perceived Trustworthiness and Cooperation in Interpersonal andIntergroup Relationships. Organizational Behavior and Human Decision Processes, 2008.

[10] First Workshop on Architecture in Global Software Engineering Helsinki, Finland, August 15, 2011 http://www.cs.bilkent.edu.tr/AGSE-2011/. 2011.

[11] W.B. Frakes and K. Kang. Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering, Vol.31, No. 7, July 2005.

[12] R. Heeks, S. Krishna,B. Nicholson, and S. Sahay. Synching or Sinking: Global Software Outsourcing Relationships, IEEE Software, p. 54-60, Mar/Apr, 2001.

[13] J.D.Herbsleb and A. Mockus. An Empirical Study ofSpeed and Communication in Globally-Distributed SoftwareDevelopment. IEEE Transactions on Software Engineering, Vol. 29(3), 2003.

[14] J.D. Herbsleb. Global Software Engineering: The Future of Socio-technical Coordination. International Conference on Software Engineering. p. 188-198, 2007.

[15] J.D. Herbsleb and D. Moitra. Global Software Development. IEEE Software, March/April, p. 16- 20, 2001.

[16] [ISO/IEC/IEEE 42010:2011] International Organization for Standardization & International Electrotechnical Commission& IEEE. Systems and software engineering—Architecture description, March 2011.

[17] S. Jalali, C. Gencel, and D. Smite. Trust dynamics in global software engineering. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10), 2010.

[18] S. Jarvenpaa. and D. Leidner.. Communication and Trust inGlobal Virtual Teams. Journal of Computer Mediated Communication 3(4), June 1998.

[19] S. L. Jarvenpaa, K. Knoll and D. E. Leidner. Is anybody out there? Antecedents of trust inglobal virtual teams. Journal of Management Information Systems 14(4), 29–64 (1998).

[20] C. Kostenko, Architects Responsibilities. url: http://www.softwarearchitectures.com/, accessed: November 2011.

[21] S. Krishna,S. Sahay, and G. Walsham.ManagingCross-Cultural Issues in Global Software Outsourcing.Communications of the ACM. Volume 47, Number 4, 2004.

[22] P. Kruchten. The 4+1 View Model of Architecture. IEEE Software, 12(6):42–50, 1995.

[23] R. J. Lewicki, D. J. McAllister and R. J. Bies. Trust and Distrust: New Relationships and RealitiesThe Academy of Management ReviewVol. 23, No. 3, p. 438-458, 1998.

[24] A. Mockus and J. Herbsleb. Challenges of Global Software DevelopmentSoftware Metrics Symposium METRICS 2001. Proceedings. Seventh International. p. 182 – 184, 2001

[25] N. B. Moe and D.Smite. Understanding a lack of trust in Global Software Teams: a multiple-case study. Software. Process 13, 3,p. 217-231.2008.

[26] A. Piri, T. Niinimäki, and C. Lassenius. Fear and distrust in global software engineering projects. Journal of Software Maintenance and Evolution Research and Practice, 2010.

[27] B. Tekinerdogan, S. Cetin, F. Savci. Exploring Architecture Design Alternatives for Global Software Product Line Engineering, in the Proc. of the Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October 2011.

[28] G. D. Venolia, R. DeLine, and T. LaToza. Software Development at Microsoft Observed, no. MSR-TR-2005-140, October 2005.

[29] B.M. Yildiz & B. Tekinerdogan. Architectural Viewpoints for Global Software Development. in the Proc. of Global Software Engineering, First International Workshop on Architecting in Helsinki, pp. 9-16, August, 2011.

[30] B.M. Yildiz & B. Tekinerdogan. Meta-Model For Global Software Development to Support Portability and Interoperability, in the Proc. of the Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October, 2011.

[31] J. B.Walther & U. Bunz. “The rules of virtual groups: Trust, liking, and performancein computer-mediated communication”, Journal of Communication, 55, p. 828-846, 2005.

DOI: 10.1145/2088883.2088900 http://doi.acm.org/10.1145/2088883.2088900