Architectural Knowledge Management in Global Software Development: A Review

Nour Ali, Sarah Beecham
Lero-The Irish Software Engineering Research Centre
University of Limerick, Ireland
{nour.ali; sarah.beecham}@lero.ie

Ivan Mistrík
Software Systems Scientist
Heidelberg, Germany
i.j.mistrik@t-online.de

***Abstract*—Architectural Knowledge Management (AKM) aims to coordinate the knowledge produced and used during architecting a software system. Managing architectural knowledge effectively is a task that becomes even more critical and complex when operating in a distributed environment. Thus, software architectural practices, processes, and tools that work in collocated software development don't necessarily scale up in a distributed environment. In this paper, we perform a literature review that looks at AKM in a Global Software Development (GSD) context. We attempt to synthesize AKM concepts, practices, tools and challenges important in GSD. In order to provide a common understanding for the central concepts of AKM in GSD in an abstract way, we have created a metamodel which is based on our literature review. The metamodel defines a set of architecture knowledge and global software development entities and their relationships.**

Keywords-Architectural Knowledge; Software Architecting; Distributed Software Engineering; Knowledge Management

I. INTRODUCTION

Global Software Development (GSD), multi-site software development or distributed software development, is generally defined as 'software work undertaken at geographically separated locations across national boundaries in a coordinated fashion involving real time (synchronous) and asynchronous interaction' [30]. In GSD, software teams work together to accomplish project goals from different geographical locations. Globally distributed teams face challenges associated with the coordination of their work due to the location, and temporal difference [11].

Architecture Knowledge Management (AKM) involves capturing and sharing the structure of systems in architecture designs [28], the design decisions and their rationale [8] and other knowledge used in defining and using the architecture such as requirements documentation, or architectural styles. Researchers have identified that architecture knowledge such as design decisions and architecture design can serve to support and underpin the complex collaboration needs of software organisations operating in a globally distributed environment [24].

A wide range of research has been conducted in the area of AKM in GSD, which include empirical studies, tools, theories, literature reviews. In this paper we combine the practices associated with managing software architecture knowledge in a global software environment. We do this through a review of the literature on Architectural Knowledge Management (AKM) in GSD where we identify the various constructs and their relationships. We summarise these relationships in a metamodel that clearly shows how constructs inter-relate. Our aim is to make the process more transparent so that organisations working in a distributed environment can leverage the benefits of GSD by managing the architectural knowledge effectively. There have already been some efforts in defining metamodels for AKM such as the one proposed by de Boer et al. [18] or Akerman et al. [1]. However, none have focused on creating a metamodel with the critical and specific concepts of AKM in GSD.

On the other hand, in a global software environment, organizations face a real challenge in managing their knowledge [29]. AKM is by no way any different. The architectural decisions among different sites need to be taken and architecture knowledge needs to be captured and shared among the distributed teams. In this context, managing the architectural knowledge in GSD becomes a more critical task than in a collocated environment e.g. the communication and the exchange of architectural knowledge becomes more complex, and the coordination of the groups, activities and artifacts involved in the distributed architecting process becomes more difficult to manage. In this paper, we also explain the practices involved in coordinating distributed architecture knowledge. In addition, we identify the open challenges in the current state of the art in the area of AKM in a GSD context.

The paper is structured as follows: Section 2 describes the methodology that we followed to conduct the review, Section 3 presents the results of our review in three subsections: Subsection A explains the metamodel we defined for AKM in a Global Environment, which synthesizes the concepts found in our literature review. The metamodel defines a set of architecture knowledge and global software development entities and their relationships. Subsection B presents coordination strategies that have been used for AKM in GSD. Finally, Subsection C synthesizes open challenges encountered in the literature. Section 4 highlights our conclusions.

II. METHODOLOGY

This section explains briefly how, through a systematic investigation of the literature, we identified key concepts associated with AKM in a global environment and how these concepts relate to one another. Through a review of the literature, we seek to answer the following research questions:

RQ1: What are the key concepts in AKM when viewed in

978-0-7695-4122-8/10 $26.00 © 2010 IEEE
DOI 10.1109/ICGSE.2010.48

Global Software Development context and how do these concepts interact with each other?

Rationale: we need to identify the concepts involved in AKM which are important in a GSD context as these concepts need special attention to improve the distributed work .

RQ2: What are the architecture knowledge coordination practices used in GSD?

Rationale: We need to harness good practices to support industry and build on existing knowledge.

RQ3: What challenges remain in the area of AKM for GSD?

Rationale: This will indicate what work remains for research and points to where gaps in knowledge occur.

For this study, we have taken a focused yet systematic approach to identifying research publications relevant to our research questions. We select a sufficient collection of key studies to allow us to identify recurring themes in a cross section of studies. This methodology is very similar to that used in [1].

This study conformed largely to the guidelines in [2], with some modifications as discussed below.

Need for a review: Although we found a very useful survey on GSD practices in AKM [17] as validated in [15], and have included this relevant work in our review, Clerc's study does not highlight the relationships between the key concepts; hence, the need for this review which also incorporates some new studies not included in [17] [15].

Search: We used the following Boolean search string "Global Software" AND "Architect*" to ensure we captured a wide variety of papers that related to practices in AKM in global software development. We used this string to search the *IEEEXplore (http://ieeexplore.ieee.org)* bibliographic database which resulted in 30 identified papers. Using our document selection criteria (explained in next section), we were able to reject 12 of these papers leaving us with a list of 18 primary studies. We then conducted a 'secondary' search based on papers cited in these 18 accepted papers and performed separate searches on key GSD and AKM conference and workshop proceedings to include: SHARK, KNOWING, WICSA/ECSA, and ICGSE and found a further 7 papers. This process resulted in a selection 25 papers in total that we used to answer our research questions reported in section III.

Document selection: Inclusion and exclusion criteria were used to select the subset of papers from those identified by the initial search where we include texts that:

- directly answer our research questions
- present concepts or entities that capture how to manage AKM in GSD (where a concept could be a role, an action or a specific tool).
- represent empirical observations relating to our research questions
- are full research papers, peer reviewed, published in a journal or conference proceedings.

Data extraction, meta-analysis, and interpretation.

We examined each selected study to extract concepts relating to AKM in GSD; then, we synthesized the data by first identifying major categories that answer our research questions by filling in prepared forms. Two researchers completed these forms. We give each occurrence the same weight, so the frequencies merely reflect how many times a given concept or relationship is identified in different papers, not how important it might be.

III. RESULTS

In this section we present the results of our literature review. We present these results in three subsections. Subsection A presents AKM and GSD concepts and their relationships in a metamodel. Subsection B presents coordination strategies that have been used for AKM in GSD. Finally, Subsection C presents open challenges.

A. DEFINING A CORE MODEL FOR AKM in GSD

One of the objectives of this paper is to provide a common understanding of how Architectural Knowledge is managed in Global Software Development. In order to do this, we define a conceptual model (or metamodel) that takes into account the AKM and GSD concepts identified in the literature (Fig. 1). For simplicity and clarity, we have used the Unified Modeling Language (UML) class diagram to specify our metamodel.

Table 1 summarizes the meta-analysis we have performed in order to derive the metamodel entities and their relationships. Fig. 1 presents the metamodel that shows how AKM entities are related in GSD. The relationships among the entities are identified with letters indicated in Table 1. In the following, we list each entity in italics and give examples of how various studies have found them important in a GSD context. Relationships identified as I, L, and M are not encountered in Table 1 because they are not specific for GSD context and are common standards in the software architecture community.

TABLE I. META-ANALYSIS FOR DERIVING ENTIY RELATIONSHIPS FOR AKM IN GSD

	Relationships	**References**
A	An *Organization* is composed of one or many *Distributed Teams*	[7], [13], [23]
B	*Distributed Teams need Coordination Strategies*	[5], [7], [15], [17], [22], [23], [26], [31]
C, D	An *architect (or architect representative) can form part of a Distributed Team* to improve coordination	[7], [26], [27], [31]
E	*Software architecture* drives the *organization's* structure	[7], [13], [14], [23], [27], [31]
F	A *Component* is allocated to one *Distributed Team*	[7], [23], [24], [25], [26] , [31]
B, G	*Distributed Teams need Coordination Strategies* to develop *Interfaces*	[7], [23], [24]
H	*Architectural Styles/Patterns* used for identifying *Coordination Strategies* .	[7], [23], [27]
J, B	*Design Decisions* are communicated to *Distributed Teams* through *Coordination Strategies*.	[7], [12], [13], [14], [17], [24], [27]
K	*Non-functional Requirements* have to be stable and satisfied in the *Software Architecture*	[9], [12], [17], [22], [23]

Key: In table 1 entities are given in Italics

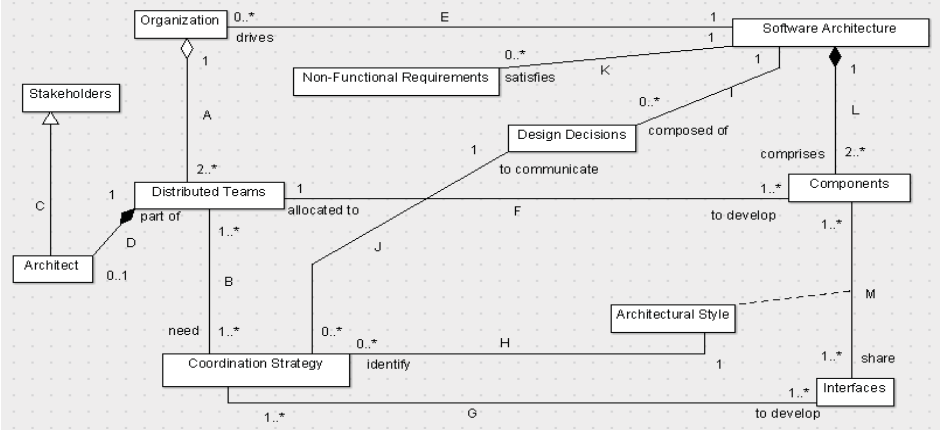

Figure 1. Metamodel for AKM in a Global Environment derived from Table 1

In GSD, an *Organization* is composed of one or many *Distributed Teams* [7], [13], [23] (*A*). *Distributed Teams* are composed of the *Stakeholders* of the software architecture (*C*). *Stakeholders* of an architecture can be integrators, designers, customers, or developers, e.g. [27]. In our literature review, distributed organizations have defined the composition of their distributed teams differently. For example, an organization can have a Distributed Team which can be the requirements/architecture team located at the headquarters and another Distributed Team to be the development team [7]. *Distributed Teams* need to have accessible *Coordination Strategies* in order to allow them to communicate in an efficient way (B). A *Stakeholder* that is recommended to exist at each *Distributed Team* is the architect (C, D), who is in charge of defining the architecture and can act as a coordinator between its local team and the distributed ones [7], [26], [27], [31] .

A *Software Architecture* design consists of *Components* and the *Interfaces* [28] which connect *Components*. A *Software Architecture* drives the structure of an *Organization* [7], [13], [14] [23], [31] (*E*). A Software Architecture has been used as a means for coordinating projects [7], [23]. The software architecture of a system is used as a coordination mechanism in distributed organizations to allocate tasks and coordinate the *Distributed Teams* [13], [24].

The development of a loosely coupled *Component* (subsystem, module or services) is allocated as a development task to *one Distributed Team* [7], [23], [26], [24], [31] (*D*). In this case the coordination among teams is not an issue. Thus, *Components* are treated as independent work-packages in order to minimize the communication between sites [25]. How these work-packages are allocated can depend on the expertise of *Distributed Teams* at given sites [26]. A *Distributed Team* can develop *many Components* but a *Component* is developed by only one team.

When *Interfaces* are developed, *Coordination Strategies* are required to coordinate the *Distributed Teams* in order to manage architectural dependencies among tasks (B, G). This means that *Coordination Strategies* are essential for allowing *Distributed Teams* to communicate and share knowledge when component *Interfaces* are developed [7], [23], [24].

In [27], it is reported that the ways components interact (*Architectural Styles or patterns*) can affect the decisions on choosing *Coordination Strategies*. Herbsleb suggests in [23] that understanding specific architecture tactics or styles, can improve shaping the task dependencies among the teams teams (H). For example in [7], a technique called Design Structural Matrix is used in order to understand the relationship between component interaction and team interaction.

Software Architecture as defined by Bosch [8] is not only the typical component connector view but it is also considered to be a composition of architecture *Design Decisions* (I). As a result of this work, software architecture is now viewed as a decision making process where decisions might be related to design. In GSD, the architecture *Design Decisions* and their rationale have to be communicated to the *Distributed Teams*' members that are architecture *Stakeholders* [27]. It has been proven that when architecture *Stakeholders'* understand properly the rationale behind a design decision they are more encouraged to comply with the architecture. Tang el at. [34] have demonstrated this empirically in collocated environments. In addition, it has been reported that this is important in distributed environments [12], [13], [14], [24]. *Design Decisions* have to be communicated to the *Distributed Teams* by adopting *Coordination Strategies* [27] (J, B) (see section B for more details and examples).

Understanding architecture *Design Decisions* and defining the organizational structure are essential for the overall software development process. Avritzer et al. [7] and Laredo et al. [26] report that *Design Decision* view of an architecture is used to understand the communication patterns in a distributed organization and the coordination of teams.

Non-Functional Requirements have to be satisfied in a *Software Architecture (K)*. Once a high level architecture is designed, the *Non-Functional Requirements* have to be stable [12], [17], [22], [31]. An architect has to collaborate in the requirements elicitation phase to understand and model the non-functional requirements. The work in [9] reports that an architect is assigned in a site near to the customer of the product in order to participate in the requirements elicitation

and include properly the *Non-Functional Requirements* in the software architecture design.

In this section, we discussed the different concepts that are important to take into account in AKM in a distributed environment. Although these concepts are important to be understood in GSD, the unique and critical entity is the coordination strategy. The different coordination strategies encountered in the literature are not reflected in the metamodel of Fig. 1. This is due to the fact that it is difficult to recommend certain strategies as well as they are no evidence to demonstrate that one is better than another. As a result, in the next section we discuss some of them.

B. ARCHITECTURE KNOWLEDGE COORDINATION STRATEGIES IN GSD

In this section, we explain the different coordination strategies that have been identified in our literature review. Clerc refers to coordination and communication strategies as the solutions provided to overcome the way individuals interact with each other in a distributed setting [17]. According to Herbsleb [23], coordination in GSD is managing dependencies among tasks over distance and includes features as communication, tools, processes, and practices.

Recalling the metamodel of Fig 1., *Coordination Strategies* are needed in a distributed environment for communicating *Design Decisions* to *Distributed Teams* and for coordinating *Distributed Teams* that develop *Interfaces* among *Components*.

There are three distinct strategies for managing architectural knowledge [19]: codification, personalization or a hybrid approach which combines the previous two. In the codification strategy, the architecture knowledge is codified and stored in a central repository. In personalization, each stakeholder stores their own knowledge, and stakeholders are encouraged to contact owners of knowledge, when this knowledge is needed. The hybrid strategy encourages having a central repository for knowledge shared across an organization and allowing stakeholders to know when, how or by whom knowledge is done in order to allow sharing of personalized knowledge. In [17], Clerk encountered that most AKM practices are supported by a personalization strategy. These are discussed in [5] and [17] . Clerk et al. [15] empirically prove that most AKM practices are perceived to be more important than centralized strategy in organizations.

Clerc et al. investigate the relationship between the number of *Distributed Teams* and the usefulness of architecture knowledge *Coordination Strategies* (practices) [15]. These practices include: having collocated face-to-face kick off meetings, providing a centralized/decentralized architecture knowledge repositories to share and manage architecture knowledge such as [22], or having a mailing list to quickly get information. Clerk concludes that planning AKM is useful.

Clerc [17] recommends that the high level architecture design should be defined through *Architects* from different sites meeting at collocated face to face meetings. A practice used for improving coordination is that each *Distributed Team* can include an architect. One of the *Architects* from the *Distributed Teams* can be chosen to be the leading architect. Usually, the *Leading Architect* can be chosen from the site near to the customers [9] or at the headquarters of the organization [26].

Many organizations evolve their communication and *Coordination Strategies* iteratively. In [26], initially the adopted communication between an architect in the headquarters (central site) and other members of the project was through emails. The architect received large volumes of emails which could cause bottlenecks. To solve this problem, in the last phases of the project, they adopted SCRUM, an agile development method [26]. Using this method, the architect met in a conference call once each week with managers and group leaders in distributed teams.

Initially in Avritzer et al. [7], a software architecture design was performed in a central team. This central team would be also responsible for eliciting and capturing the requirements, and assigning the component implementations to distributed teams. When *Interfaces* between *Components* needed to be developed, the central team would coordinate the communication between the development teams. However, this had the drawback that the central team would be overloaded with communication from the remote sites.

To solve the problem of unbalanced responsibilities, Avritzer et al., adopted a hybrid/centralized approach where *Architects* where distributed in each *Distributed Team* [7]. The central team stayed as the architecture lead. Similar to the recommended practices found in [17] , in the first stages of the project, architects met in a face to face meeting in the central team. Afterwards, once a week a teleconference was made where all project members where invited to attend. It was noticed, that the communication among teams was only performed through architects and this decreased the need for communication. The authors of [7] believe that coordination is more scalable, important tasks are distributed and trust is created between the management and the remote teams. Salger [31] reported to have adopted a similar hybrid/centralized approach similar to Avritzer et al. [7].

There are many web based tools that provide architectural knowledge management support. Ali-Babar has applied an electronic workspace paradigm for capturing and sharing architecture knowledge support for the software architecture process [1]. Other tools not only capture the knowledge but also provide extra facilities. Farenhorst et al. present a portal where documents can be searched, and a view of the project stakeholders with their expertise and discussion boards is supported [21]. The portal provides support for a hybrid AKM strategy. Solis et al. provide a wiki tool which provides task allocation, templates, brainstorming, and decision making support [33]. Capilla et al. [10] report on a web based tool for managing design decisions.

C. CURRENT CHALLENGES

Despite the many helpful coordination practices specified in the previous section, several issues relating to AKM in GSD still have to be tackled. In general, we need to conduct further empirical studies to explore the relation between

software architecture and communication requirements [7]. For example, Herbsleb [23] identified three main challenges: 1) To understand the knowledge about the relationship between software architecture dependencies and task dependencies, 2) To assess how an organization is prepared to carry out the design and implementation of an architecture, and 3) To provide tactics that adjust the organization to the architecture and vice versa. Going back to Herbsleb's earlier empirical work, we still need practices to deal with changing requirements, staff turnover and extreme schedule pressure [24] .

Clerc identified that there still remains a problem of ensuring architectural compliance in a distributed team [16]. Components need to be ready for integration on the schedule described in the plan. Workarounds need to be sought to compensate for the lack of chance discussions that is only possible in a collocated site to help developers recognize and resolve conflicts early on. Also a method is needed to pass on general information across sites such as "how things work, what issues have priority, responsibility assignments, and who was an expert at what" [24]. The problem still remains that there will be cultural differences that need to be recognised, for example, in [27] the architect and designers had cultural differences and could not communicate with each other effectively especially since physical distance between subproject participants prevented informal communication.

The literature has observed many further challenges that include the provision of a suite of tools especially for global software development [26]. Where there is a need for a tool that provides software architecture evaluation in GSD [31], which is partially satisfied in [3] but still needs to be made acceptable to all stakeholders [5]. Users need to be able to configure templates based on their individual needs and to avoid duplication of work AKM tools should be integrated with requirements management tools [6]. Architects are requesting even further integration to include existing tools such as email clients, calendars, project tools that assign people to tasks or activities using the portal [21]. However, even if these needs are met, the problem remains that tool based arrangements may not be appreciated in a distributed environment because of "lack of body language, collocated being more natural and conventional, typing problems, slow collaboration, time lag in communication, and so on " [3]. The challenge remains that we need to overcome the problems associated with the lack of personal interaction in a distributed environment shown to inhibit the transfer and acceptance of tools across sites [24].

Many of the challenges of AKM become critical in a distributed environment such as addressing non-functional requirements by giving guidelines for how to select architectural patterns/styles [12], [9]. Even the most experienced technical architects fall short when asked about metrics that need to be captured and often do not know *what* to capture [9]. The importance of having non-functional requirements stable at the time of defining the architecture becomes more essential in global software development due to the fact that coordination and communication are more complex than in a collocated environment. In addition, non-some of the recommended coordination practices such as defining the high level architecture in a collocated way suggest that the non-functional requirements should be properly specified [17]. Following the support for non-functional requirements, the need for quality attributes to be added to AKM tools together with a categorization of dependencies between decisions is also called for in [10]. There is a need to create a knowledge repository for software architectural patterns. The repository could also contain information to help find the right modularization in order to support the communication between the teams [31]. Regarding scalability, when the number of sites involved in software development projects increases the perceived usefulness of AKM practices does not necessarily increase. This is shown in [15], where the perceived usefulness of AKM practices is at its maximum in software development projects with three sites.

IV. CONCLUSIONS

Through a meta-analysis of the literature on Architecture Knowledge Management and Global Software Development, we have been able to give an overview of the key elements involved. These include the AKM concepts and the coordination strategies that are important to consider in GSD as well as the gaps that have to be filled in the further research.

We have synthesized the accepted AKM concepts and practices in globally distributed organizations through a metamodel. The metamodel includes concepts encountered in practice, as all findings are empirically founded. This metamodel will allow organizations to focus on AKM concepts that are important to consider for defining their strategies in a distributed environment. For example, architectural styles and design decisions are important inputs for defining coordination strategies in an organization. In addition, an organization has to carefully choose coordination strategies, e.g., when developing interfaces of components.

Coordination is an important entity in global software development. However, we have found that in the literature there is no common consensus concerning coordination strategies to be applied for AKM. In this paper, we have explained possible coordination strategies that have been considered in the current literature. These include structuring an organization for improving communication among architecture stakeholders such as including a leading architect; discussing practices for improving coordination such as having collocated face to face meetings; knowledge management strategies that can de implemented for sharing architectural knowledge, and several tools that support distributed stakeholders in the decision making process.

In this study we also discussed the major challenges that remain unresolved, such as the need for a suite of tools specifically aimed at managing architectural knowledge in a global setting. Other challenges might prove more difficult to combat, such as how to manage the problem of communicating design decisions across teams that are culturally and geographically dispersed.

In our near future, we plan to extend our literature review to include further studies. We also plan to validate our metamodel in industry as well as study how current tools support it.

ACKNOWLEDGMENT

This work was supported by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre.

REFERENCES

[1] Akerman, A., Tyree, J. , "Position on Ontology-Based Architecture", 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), Pittsburgh, USA, 2005, pp. 289–290.

[2] Ali-Babar M, "The application of knowledge-sharing workspace paradigm for software architecture processes", Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge (SHARK'08) at ICSE 2008, Leipzig, Germany, pp. 45-48

[3] Ali-Barbar. A Framework for Supporting the Software Architecture Evaluation Process in Fourth IEEE International Conferences on Global Software Engineering, (ICGSE 2009).

[4] Ali-Babar, M., de Boer, R.C., Dingsøyr, T., and Farenhorst, R. "Architectural Knowledge Management Strategies: Approaches in Research and Industry", *Second ICSE Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent 2007 (SHARK-ADI'07)*. Minneapolis, MN, USA,

[5] Ali-Babar, M., B. Kitchenham, and R. Jeffery, *Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment.* Empirical Software Engineering (ESE), 2008. 13(1): p. 39-62.

[6] Ali-Babar, M, Northway, A, Gorton, I, Heuer, P and Nguyen, T. "Introducing Tool Support for Managing Architectural Knowledge: An Experience Report", 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.

[7] Avritzer A, Paulish D, Yuanfang C, "Coordination implications of software architecture in a global software development project", Seventh working IEEE/IFIP Conference on Software Architecture (WICSA 2008), 2008, pp. 107-116

[8] Bosch, J. , "Software architecture: the next step", European Workshop on Software Architecture (EWSA 2004), pp. 194-199

[9] Caprihan, G. (2006) Managing software performance in the globally distributed software paradigm in global software engineering. International Conference on Global Software Engineering (ICGSE 2006), pp. 83-91

[10] Capilla, R., Nava, F., Perez, S., and Duenas, J.D. "A web-based Tool for Managing Architectural Design Decisions", 1st Workshop on SHARing and Reusing architectural Knowledge Sharing and Reusing Architectural Knowledge.

[11] Carmel, E., "Global Software Teams: Collaborating Across Borders and Time Zones". Prentice Hall, Upper Saddle River, 2009

[12] Clerc V., "Do architectural knowledge product measures make a difference in GSD?", Fourth IEEE International Conference on Global Software Engineering (ICGSE/KNOWING 2009), pp. 382-387

[13] Clerc, V., P. Lago, and H. Van Vliet. "Assessing a Multi-Site Development Organization for Architectural Compliance", Sixth Working IEEE/IFIP Conference on Software Architecture. 2007: IEEE Computer Society.

[14] V. Clerc, P. Lago, and H. van Vliet, "Global Software Development: Are Architectural Rules the Answer?" Second IEEE International Conference on Global Software Engineering, 2007. ICGSE 2007.

[15] Clerc, V., Lago P, van Vliet H, "The usefulness of architectural knowledge management practices in GSD", International Conference on Global Software Engineering (ICGSE 2009), pp. 73-82

[16] Clerc, V., P. Lago, and H. Van Vliet., "Assessing a Multi-Site Development Organization for Architectural Compliance", Sixth Working IEEE/IFIP Conference on Software Architecture. 2007: IEEE Computer Society.

[17] Clerc V, "Towards architectural knowledge management practices for global software development", 3rd International Workshop on Sharing and Reusing Architectural Knowledge (SHARK'08) at ICSE 2008, Leipzig, Germany, pp. 23-28

[18] de Boer, R. C., Farenhorst, R., Lago, P., Vliet, H. van, Clerc, V. , Jansen, A, Architectural Knowledge: Getting to the Core. In *Third International Conference on Quality of Software-Architectures (QoSA)*, volume 4880 of *LNCS*, pp 197–214.

[19] Desouza, K.C., Awazu, Y., Baloh, P., *Managing Knowledge in Global Software Development Efforts: Issues and Practices*, IEEE Software, vol. 23, no. 5, pp. 30-37, Sep./Oct. 2006, doi:10.1109/MS.2006.135

[20] Farenhorst R, de Boer RC., "Knowledge management in software architecture: state-of-the-art", Software Architecture Knowledge Management. Ali-Babar M, Dingsoyr T, Lago P, van Vliet H (editors), Springer-Verlag, 2009, pp. 21-38

[21] Farenhorst R, Izaks R, Lago P, van Vliet H. "A just-in-time architectural knowledge sharing portal", Software Architecture 2008 (WICSA 2008), pp. 125-134

[22] Faria, H.R.d. and G. Adler. *Architecture-Centric Global Software Processes*. in *Global Software Engineering, 2006. ICGSE '06. International Conference on*. 2006.

[23] Herbsleb JD (2007) Global software engineering: the future of socio-technical coordination. Future of Software Engineering (FOSE 2007), pp. 188-198

[24] Herbsleb, J.D. and R.E. Grinter, *Architectures, Coordination, and Distance: Conway's Law and Beyond.* IEEE Software, 1999. 16(5): p. 63-70.

[25] Lamersdorf, A., Munch J, Rombach D., "A Survey on the state of the practice in distributed software development: criteria for task allocation", Fourth IEEE International Conference on Global Software Engineering(ICGSE 2009), pp. 41-50

[26] Laredo JA, Ranjan R (2008). Continuous improvement through iterative development in a multi-geography. Third IEEE International Conference on Global Software Engineering 2008, pp. 232-236

[27] Ovaska, P., Rossi, M., Marttiin, P., *Architecture as a coordination tool in multi-site software development*, Software Process: Improvement and Practice, 2003; **8**: 233–247

[28] Perry DE, Wolf AL (1992) Foundations for the study of software architecture. SIGSOFT Software Engineering Notes, 17(4): 40-52, 1992

[29] Richardson I, Casey V, O'Riordan M, Meehan B, Mistrík I (2009) Knowledge management in the global software engineering environment. Fourth IEEE International Conference in on Global Software Engineering (ICGSE/KNOWING 2009), pp. 367-369

[30] Sahay, S., Nicholson, B., Krishna S.(2003) Global IT Outsourcing : Software Development Across Borders, Cambridge Univerity Press, Cambridge.

[31] Salger F (2009) Software architecture evaluation in global software development projects. Meersman R, Herrero P, Dillon T (editors): OTM 2009 Workshops, LNCS 5872, Springer, pp. 391-400

[32] Shaw M, Clements P (2006) The golden age of software architecture. IEEE Software, 23(2): 31-39, 2006

[33] Solis C, Ali N, Ali-Babar M (2009) A spatial hypertext wiki for architectural knowledge management. ICSE Workshop on Wikis for Software Engineering (WIKIS4SE 2009), pp. 36-46

[34] Tang, A., Tran, M. H., Han, J., van Vliet, H., "Design Reasoning Improves Software Design Quality", QoSA 2008: 28-4