Towards Architectural Knowledge Management Practices for Global Software Development

Viktor Clerc
Department of Computer Science
VU University Amsterdam
the Netherlands
viktor@cs.vu.nl

ABSTRACT

In the past few years, an increasing interest in architectural knowledge is recognized in the software architecture community. Architectural knowledge is generally regarded as important to guide the development of software systems. With the trend of Global Software Development (GSD), the management of architectural knowledge becomes even more important due to the geographical, temporal, and socio-cultural distance innate to GSD. In this paper we build on the *requirements engineering* discipline to identify practices that can aid in overcoming GSD challenges and assess their applicability for management of architectural knowledge in a GSD setting. We provide a light-weight pattern language that we use to describe architectural knowledge management practices and provide a first validation of these practices from an ongoing case study.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Design

Keywords

Architectural Knowledge, Requirements Engineering, Global Software Development

1. INTRODUCTION

In the past few years, an increasing interest in architectural knowledge management is recognized in the software architecture community [28, 29, 2, 33]. Architectural knowledge is generally regarded as important to guide the development and evolution of software systems [29].

With the trend of Global Software Development (GSD), architectural knowledge management becomes even more important due to challenges that arise as a result of the geographical, temporal, and socio-cultural distance innate to GSD [25]. Overviews of the challenges in GSD have been widely reported [1, 6, 8, 9, 25] and include lack of informal contact, language differences, and coordination complexity [1].

Solutions to overcome GSD challenges generally deal with the way individuals interact with each other in a distributed setting. These solutions are provided in terms of communication and coordination strategies [22, 21, 30]. In these strategies, recording decisions about the architecture plays a pivotal role [8, 22]. However, we are currently lacking detailed insight into architectural knowledge management practices that can effectively be applied in a GSD setting, following observations from the literature on challenges in knowledge sharing in a GSD situation [15, 4].

To address this lack of insight, we build on the discipline of requirements engineering. The requirements engineering discipline is a well-discussed example of a discipline that becomes challenging in GSD [11, 26]. We observe a close resemblance between a set of requirements for a software system and a set of architectural decisions taken for that software system: what one person regards as requirements for a software system, another person may regard as architectural decisions [32]. Furthermore, we conjecture that the same challenges that the requirements engineering discipline faces with GSD hold for architectural knowledge management practices in GSD as well: as with requirements, architectural decisions too need to be communicated across different sites in order to maintain a shared vision of the software system that is designed.

We have constructed a set of architectural knowledge management practices based on a study of relevant literature on the requirements engineering discipline related to GSD. We describe these practices using a light-weight pattern language. The elements of this pattern language and its application on describing architectural knowledge management practices may help organizations in making a well-supported choice between architectural knowledge managements practices for application in GSD projects. We provide an initial validation of the usefulness and effectiveness of these practices through semi-structured interviews on these practices within a large IT service provider.

This paper is structured as follows. Section 2 lists the approach we applied for this research. Next, in Sect. 3 we povide an overview of the resulting architectural knowledge management practices in global software development. We then provide an initial validation in Sect. 4 and list our conclusions and ongoing research in Sect. 5.

SHARK'08, May 13, 2008, Leipzig, Germany.

Table 1: Pattern Language

Practice Name	An unique name describing the practice
Intent	The goal for which this practice can be applied
Motivation	An example in which or discussion why the given practice proves to be useful
Prerequisites	Prerequisites or limitations that exist when applying the practice
Benefits	The benefits, side effects and trade-offs
Drawbacks	The drawbacks or negative effects of the practice
Strategy	The architectural knowledge management strategy which is supported best by the practice (e.g. codification or personalization [20])

2. RESEARCH APPROACH

In this section, we describe our approach to select relevant literature for this research in Sect. 2.1 and describe a light-weight pattern language for the description of the architectural knowledge management practices in Sect. 2.2.

2.1 Literature Research

We built a representative subset of relevant literature on the topic of requirements engineering in relation to Global Software Development. First, we identified important conferences on this subject, such as the *ICSE* GSD workshops, the *ICGSE* '06 and '07 conferences, and the *RE* conferences. Subsequently, we collected the proceedings of these conferences. In addition, we selected special issues from relevant journals (*Communications of the ACM*, *IEEE Software*).

In our search to collect GSD requirements engineering practices that can help in overcoming GSD challenges, we scanned the abstracts, introduction, and conclusion of all contributions. Contributions which explicitly reported on validated practices were studied in full detail.

After having collected the GSD requirements engineering practices, we translated these practices, if needed, to the discipline of architectural knowledge management. When the requirements engineering practices mentioned "requirements", we translated this into "architectural decisions"; when the practices mentioned the "requirements engineering discipline", we translated this into the "architecting phase" or "architecture development phase". In this way, we ensure that we stay fully in line with the practices as they were initially reported in the requirements engineering literature; we list the references to the requirements engineering literature throughout the description of the practices.

2.2 A Pattern Language for Describing Architectural Knowledge Management Practices

In parallel with the literature study, we devised a light-weight pattern language to structure and describe the potential GSD architectural knowledge management solutions posed by the literature. We used the work of [5, 31], based on [18] as a basis to define a light-weight pattern language. We added relevant insights from the field of architectural knowledge management, such as the architectural knowledge management strategy organizations may adhere to [2].

Table 1 lists the results. The elements of this pattern language and its application on describing architectural knowledge management practices may help organizations in making a well-supported choice between architectural knowledge managements practices for application in GSD projects.

3. ARCHITECTURAL KNOWLEDGE MANAGEMENT PRACTICES FOR GSD

This section lists the practices for architectural knowledge management to overcome GSD challenges, distilled from the literature on the requirements engineering discipline.

When a practice does not refer to any requirements-specific terminology or process, we have not applied any modifications. For a practices that does, we translate the given requirements engineering-specific element to an appropriate counterpart in the realm of architectural knowledge management (see Sect. 2.1), following the insights from e.g. [32]. We provide an overview of the changes we made.

3.1 Frequent Interaction across Sites

Practice Name – Frequent interaction across sites [13].

Intent – This practice intends to let practitioners from different development site interact frequently with each other.

Motivation – In knowledge-intensive tasks such as requirements engineering and architecting, the geographical distance and possible time difference burden effective sharing of architectural knowledge. This may result in language and terminology problems. More frequent interaction across sites help to address these problems. In addition, more frequent interaction help in building trust across sites [9]. Furthermore, it helps in gaining awareness of the local working context of architects, developers, and other representatives which eases understanding across sites.

Frequent interaction can be implemented in a variety of ways: organizations may use on-site management visits [5] in which project status, schedules, and planning issues are discussed and technical interchanges are held. In addition, organizations may utilize cross-site delegation [5] by sending practitioners from the remote development site to the main development site and vice versa.

Prerequisites – Interactions should be planned for and performed regularly. Preference is given to on-site visits. If this appears not to be cost-effective, collaborative technologies such as video-conferencing [13, 9] or wikis [17] could be employed. Furthermore, it is advised to facilitate and improve the decision-making meetings by using a trained human facilitator [13].

Benefits – Frequent interaction helps in maintaining a shared sense of urgency [9], which can speed up development time significantly.

Drawbacks – No significant drawbacks are reported in the literature. Although setting up and maintaining the infrastructure poses some costs, this is of no major influence once the infrastructure is used widely across development sites.

Strategy – This practice supports a personalization strategy, since frequent interaction helps in identifying "who knows what" among practitioners and interaction does not focus on extracting and capturing architectural knowledge in a repository, but on sharing architectural knowledge among practitioners instead.

3.2 Cross-site Delegation

Practice Name – Cross-site delegation [5].

Intent – This practice intends to obtain better integration between teams from different development sites by delegation of team members from a local site to a remote site and vice versa [5].

Motivation – When establishing a remote site, more effort is needed to align objectives and regard each other as peers. Applying cross-site delegation helps in achieving a shared understanding of the architectural problem that needs to be solved [9]. After the delegation period, the delegate may become a liaison [9, 5].

Prerequisites – When applying this practice, it is important that each development site is able to appoint this role to suitable candidates. Consequently, it requires that similar or equal roles (e.g. architects or designers) are present at all development sites.

Benefits – Applying this practice results in an increased speed in which architectural knowledge is disseminated to all development sites. Furthermore, applying this practice establishes a means for more effective future interaction and management of architectural knowledge across sites, since practitioners know whom to contact; as such, the problems to initiate contact are alleviated [24].

Drawbacks – Structural traveling, which is implied by this practice, results in traveling, housing, and facilities costs. Organizations can use a specialized department to gain efficiency benefits. Furthermore, organizations need to carefully select the roles that travel and the frequency by which traveling occurs (possibly relative to the development phase).

Strategy – This practice supports a personalization strategy.

3.3 Face-to-Face Project Kick-Off Meetings

Practice Name – Face-to-face project kick-off meetings [13].

Intent – This practice intends to establish initial relationships across sites.

Motivation – It is important to align expectations of practitioners from all development sites involved as early as possible in a software development project to prevent delay because of uncertainties later on.

Prerequisites – A group kick-off meeting with practitioners from various sites could be difficult to plan. Although relevant literature [13, 23] reports on the importance of having face-to-face contact, remarkable results are provided as well, e.g. in a large empirical study [27]: in this study, only 4 out of 55 studied best practices concerns face-to-face communication.

Benefits – Obtaining a shared understanding of the project's context and goals may result in sharing architectural knowledge during a kick-off meeting. This helps in speeding up the initial phase of the project and finding the right people across sites.

Drawbacks – For large software development projects, a lot of development sites may be involved in determining the architecture for the system to be built. This could lead to planning problems in getting *all* relevant stakeholders together.

Strategy – This practice supports a personalization strategy.

3.4 Urgent Request

Practice Name – Urgent request [5].

Intent – This practice intends to quickly collect information on a given topic of interest.

Motivation – During the architecting phase, it can be necessary to, at a given moment in time, collect information on a specific topic in order to proceed with developing the architecture. An example could be the availability of a certain infrastructure component which could be reused across projects. Obtaining a quick response helps in keeping the architecting activities up to speed.

Prerequisites – This practice requires that a distribution mechanism is implemented. The mechanism should be implemented to be separate from general knowledge *sharing* mechanisms, since a large number of unrelevant requests, or requests for different expertise areas, lowers the sense of urgency and the motivation of the volunteers. In addition, this network of volunteers with expertise on a variety of technical subjects should be created and maintained. Management support and urge for selecting volunteers is important to achieve this. This practice furthermore requires willingness to share information, and thus a sense of openness throughout the organization.

Benefits – Quick responses to urgent and/or ad hoc questions greatly improves the speed by which architectural knowledge is shared across development sites and, consequently, an architecture for a system can be developed.

Currently, low-threshold mechanisms to implement this practice exist, e.g. *rss feeds* [17].

Drawbacks – It is necessary to establish a basic level of trust between the key individuals from different development sites to allow this practice to be successful - one needs to know *who is who* before one will respond to an urgent request.

Strategy – This practice supports a personalization strategy since it reveals potential sources of architectural knowledge, which then can be accessed through e-mail interaction.

3.5 Collocated High-Level Architecture Phase

Practice Name – Collocated high-level architecture phase [5].

Intent – This practice intends to create a sound high-level architecture in an efficient way.

Motivation – The architecting phase consists of knowledge-intenstive tasks and is important for outlining the solution structure for the future development phase. In addition, identifying a high-level architecture generally is done in limited amount of time – certainly compared to detailed design and development activities.

This practice may be implemented by arranging for a single location on which architects from the development site

meet. In addition, knowledge management tooling should be catered for to capture the design rationale and major architectural decisions during the phase.

Prerequisites – The set of most important quality requirements should be fairly stable. Requirements engineers and domain experts should be available and participate in this phase.

Benefits – When organizations apply a collocated high-level architecture phase, all relevant issues can be expected to be tackled as early as possible.

Drawbacks – As already described in Sect. 3.3, This practice does not support a specific

Strategy – This practice does not support a specific architectural knowledge management strategy.

The practice *collocated initial architecture phase* has been adapted from the requirements engineering domain, where it was termed *collocated analysis phase* (see [5]). The practice describes how functional specifications are produced as a result of this collocated analysis phase with a high-level architecture as input. Based on the similarities between requirements and architecture [32, 19], we conjecture that a similar practice is applicable for architectural knowledge management in a GSD setting. We are aware that this practice removes the global aspect in GSD by introducing collocatedness. However, this practice does pose additional issues, such as effective sharing of the architectural knowledge of this single location to all development sites, which may be problematic [8].

3.6 A Clear Organization Structure with Communicating Responsibilities

Practice Name – A clear organization structure with communicating responsibilities [11].

Intent – This practice intends to maintain open communication lines between well-defined stakeholder roles [15].

Motivation – Delay can occur when practitioners spend time on finding and reaching the correct person in a given role. Lack of information often causes burden as well – it is not that information is not available, it is just not shared with the right persons. In earlier research [9], we observed this as well: an organization which applied top-down definition and communication of architecture principles and an organization with more focus on collaborative architecting practices both followed architectural rules; in the former organization, however, architectural decisions did not sink in properly.

This practice may be implemented by creating roles with clear responsibilities and by assigning these roles to the distributed stakeholders. Furthermore, it should be indicated which roles need to communicate with each other in structured meetings.

Other implementation-specific suggestions as mentioned by [27] are: communicating more often, communicating immediately as a question arises (see Sect. 3.4), communicating according to formal rules (see [9]), using a tool such as a wiki, or using common terminology.

Prerequisites – It is important to involve the right people in the meetings that are conducted [14].

Benefits – Cross-functional teams associated with developing a specific part of the architecture represent social networks whose members benefit from clear identification of roles and responsibilities in the architectural knowledge management discipline [12]. It is important to identify these possible social networks in addition to managing the communication responsibilities.

Drawbacks – No significant drawbacks are reported in the literature.

Strategy – This practice supports a personalization strategy since it helps in defining the structures by which individuals can communicate with eachother. This practice does not focus on codification of the knowledge entities that are discussed and exchanged.

3.7 Establish a Repository for Architecture Artifacts

Practice Name – Establish a repository for architecture artifacts [13].

Intent – This practice intends to build a repository to store architectural decisions including the rationale of these decisions.

Motivation – Having a shared repository with architectural knowledge helps in obtaining common understanding of the architecture and architectural decisions across development sites and in obtaining awareness of the local working context [13].

Prerequisites – In order to alleviate the distance in accessing relevant information from a shared repository, it is imperative that the development sites utilize a shared infrastructure. Furthermore, it is important to structure the information in the repository, following e.g. templates as defined by [31].

Benefits – When a shared repository is in place, all development sites easily can query and access architectural knowledge.

Drawbacks – This practice poses a challenge for the codification strategy: design decisions should be captured (codified) in a way which makes them searchable and reusable in similar situations. Consequently, a starting point for the codification strategey would be to identify appropriate use cases of this architectural knowledge [10].

Strategy – This practice supports a codification strategy.

4. VALIDATION

We have selected a number of global software development projects from one domain at a large IT service provider to identify to what extent the practices for architectural knowledge management in a GSD setting are applied in practice. We included five projects from different sizes (ranging from 10 until 75 developers) and a different number of development sites (2 until 4 sites, within two countries at most). We performed an initial validation by studying the characteristics of the projects. Following that, we conducted semi-structured interviews with project managers and used the list of practices to collect feedback on the perceived usefulness of the practices. Although further validation is still ongoing, we provide initial results:

- Large introduction programmes for new-hires from remote development sites help in obtaining understanding in the context of the organization and specific techniques applied. An *off-shore desk* supports the new-hires in a variety of logistical details to allow for smooth accommodation to the organization's standards.

- A common infrastructural platform which makes use of different project-specific and generic *environments* in which members of projects can interact using collaboration-intensive tools.

- At several levels within the organization, e-mail lists exist which are characterized by frequent discussions and questions on a specific topic related to architecture. These topics are not further structured, but allow for fellow practitioners to share experiences and respond to questions.

- Several projects reported on traveling plans to allow employees from a remote site to come to the local site.

5. CONCLUSIONS AND FUTURE WORK

Our study aims to identify practices that can overcome GSD challenges in the realm of architectural knowledge management by studing the requirements engineering discipline. Our study so far shows that the practices that have been reported in the requirements management literature can be easily translated to be applicable in the field of architectural knowledge management – we do not see any reason why certain practices would not be applicable. Consequently, we feel that both disciplines can share experiences to learn in understanding each other's challenges and practices. Furthermore, we conclude that the majority of practices focus on a *personalization* strategy for architectural knowledge management since the practices support fostering interaction among knowledge workers [2]. The literature reports on several tools that can help in capturing architectural knowledge and support a *codification* strategy [7, 3, 16]. We support the arguments from [15, 20] that a hybrid approach could combine the best of both worlds.

Initial validation of the architectural knowledge management practices for global software development shows that most of the practices seem applicable in an industrial setting. Our future work is aimed at a further structured validation of the list of architectural knowledge management practices for GSD by collecting detailed feedback on when practices have been applied and on the perceived success of their application. We plan to augment the list, which is currently primarily based on the literature, with additional practices by focusing on the different types of distance (as e.g. identified in [25]) to build a balanced set of practices for architectural knowledge management in GSD.

Our study so far shows that some of these practices may be considered as being alternatives or could be applied in parallel: having a collocated high-level architecture phase might reduce the need for frequent interaction (since the majority of the architectural knowledge is created and shared at a single site), and might more easily be combined with a face-to-face project kick-off meeting. We intend to study possible relationships and mutual influences between these practices more thoroughly in future research.

Acknowledgments

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge.

6. REFERENCES

[1] P. J. Ågerfalk, B. Fitzgerald, H. Holmström, B. Lings, B. Lundell, and E. O. Conchúir. A Framework for Considering Opportunities and Threats in Distributed Software Development. In *International Workshop on Distributed Software Development*, pages 47–61, Paris, 2005. Austrian Computer Society.

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

[3] M. A. Babar and I. Gorton. A Tool for Managing Software Architecture Knowledge. In *Second ICSE Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent 2007 (SHARK-ADI'07)*, Minneapolis, MN, USA, 2007. IEEE Computer Society.

[4] S. Balaji and M. K. Ahuja. Critical Team-Level Success Factors of Offshore Outsourced Projects: A Knowledge Integration Perspective. In *38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1*, volume 01, page 52b. IEEE Computer Society, 2005.

[5] M. Bass, J. D. Herbsleb, and C. Lescher. Collaboration in Global Software Projects at Siemens: An Experience Report. In *The Second IEEE International Conference on Global Software Engineering (ICGSE'07)*, pages 203–212, Munich, Germany, 2007. IEEE Computer Society.

[6] R. D. Battin, R. Crocker, J. Kreidler, and K. Subramanian. Leveraging Resources in Global Software Development. *IEEE Software*, 18(2):70–77, 2001.

[7] R. Capilla, F. Nava, and J. C. Dueñas. Modeling and Documenting the Evolution of Architectural Design Decisions. In *Second ICSE Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent 2007 (SHARK-ADI'07)*, Minneapolis, MN, USA, 2007. IEEE Computer Society.

[8] V. Clerc, P. Lago, and H. Van Vliet. Assessing a Multi-Site Development Organization for Architectural Compliance. In *6th Working IEEE/IFIP Conference on Software Architecture (WICSA 2007)*, Mumbai, India, 2007. IEEE Computer Society.

[9] V. Clerc, P. Lago, and H. Van Vliet. Global Software Development: Are Architectural Rules the Answer? In *The Second IEEE International Conference on Global Software Engineering (ICGSE'07)*, München, Germany, 2007. IEEE Computer Society.

[10] V. Clerc, P. Lago, and H. Van Vliet. The Architect's Mindset. In *Third International Conference on the*

Quality of Software Architectures (QoSA 2007), volume 4880 of *Lecture Notes in Computer Science*, pages 231–249, Boston, USA, 2007. Springer Berlin / Heidelberg.

[11] D. Damian. Stakeholders in Global Requirements Engineering: Lessons Learned from Practice. *IEEE Software*, 24(2):21–27, 2007.

[12] D. Damian, S. Marczak, and I. Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks. In *RE'07: International Conference on Requirements Engineering*, pages 59–68, Delhi, India, 2007. IEEE Computer Society.

[13] D. Damian and D. Zowghi. The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In *IEEE Joint International Conference on Requirements Engineering (RE'02)*. IEEE Computer Society, 2002.

[14] D. Damian and D. Zowghi. Requirements Engineering Challenges in Multi-Site Software Development Organizations. *Requirements Engineering Journal*, 8:149–160, 2003.

[15] K. C. Desouza, Y. Awazu, and P. Baloh. Managing Knowledge in Global Software Development Efforts: Issues and Practices. *IEEE Software*, 3(5):30–37, 2006.

[16] R. Farenhorst, P. Lago, and H. van Vliet. EAGLE: Effective Tool Support for Sharing Architectural Knowledge. *International Journal of Cooperative Information Systems (IJCIS)*, 16(3/4):413–437, 2007.

[17] R. Farenhorst, P. Lago, and H. van Vliet. Effective Tool Support for Architectural Knowledge Sharing. In F. Oquendo, editor, *1st European Conference on Software Architecture (ECSA)*, pages 225–234, Aranjuez (Madrid), Spain, 2007. Springer.

[18] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley Professional, 1994.

[19] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti. Relating Software Requirements and Architectures using Problem Frames. In *RE'02: International Conference on Requirements Engineering*, pages 137–144, Essen, Germany, 2002. IEEE Computer Society.

[20] M. T. Hansen, N. Nohria, and T. Tierney. What's Your Strategy for Managing Knowledge? *Harvard Business Review*, 77(2):106–116, 1999.

[21] J. D. Herbsleb. Global Software Engineering: The Future of Socio-technical Coordination, 2007.

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

[23] J. D. Herbsleb, D. J. Paulish, and M. Bass. Global Software Development at Siemens: Experience from Nine Projects. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 524–533, St. Louis, Missouri, USA, 2005. ACM Press.

[24] G. Hofstede. *Culture's Consequences: International Differences in Work-Related Values, second edition.* Sage Publications Inc., 2001.

[25] H. Holmström, E. O. Conchúir, P. J. Ågerfalk, and B. Fitzgerald. Global Software Development Challenges: A Case Study on Temporal, Geographical, and Socio-Cultural Distance. In *The IEEE International Conference on Global Software Engineering (ICGSE'06)*, pages 3–11, Florianopolis, Brazil, 2006. IEEE Computer Society.

[26] Y. Hsieh. Culture and Shared Understanding in Distributed Requirements Engineering. In *The IEEE International Conference on Global Software Engineering (ICGSE'06)*, pages 101–108, Florianopolis, Brazil, 2006. IEEE Computer Society.

[27] T. Illes-Seifert, A. Herrmann, M. Geisser, and T. Hildenbrand. The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey. In *The First International Global Requirements Engineering Workshop (GREW'07)*, pages 55–66, Munich, Germany, 2007.

[28] A. Jansen and J. Bosch. Software Architecture as a Set of Architectural Design Decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pages 109–120, Pittsburgh, Pennsylvania, 2005.

[29] P. Kruchten, P. Lago, and H. Van Vliet. Building up and Reasoning about Architectural Knowledge. In *Second International Conference on the Quality of Software Architectures (QoSA 2006)*, volume 4214 of *Lecture Notes in Computer Science*, pages 43–58, Västerås, Sweden, 2006. Springer Berlin / Heidelberg.

[30] F. Lanubile, D. Damian, and H. L. Oppenheimer. Global software development: Technical, Organizational, and Social Challenges. *SIGSOFT Softw. Eng. Notes*, 28(6), 2003.

[31] J. Tyree and A. Akerman. Architecture Decisions: Demystifying Architecture. *IEEE Software*, 22(2):19–27, 2005.

[32] H. Van Vliet. Software Architecture Knowledge Management. In *ASWEC'08: 19th Australian Conference on Software Engineering*, Perth, Australia, 2007. IEEE Computer Society.

[33] J. S. v. d. Ven, A. G. J. Jansen, J. A. G. Nijhuis, and J. Bosch. Design decisions: The Bridge between Rationale and Architecture. In A. H. Dutoit, R. McCall, I. Mistrik, and B. Paech, editors, *Rationale Management in Software Engineering*, pages 329–346. Springer-Verlag, 2006.