Collaboration tooling focuses on supporting collaboration throughout the software engineering lifecycle. Version control systems (e.g., Concurrent Versions System, Subversion) allow for over-the-web management of evolution of source code and artifacts. Issue tracking systems allow for e.g., distribution of (corrective maintenance) tasks to different sites. Other tools allow for collaborative modeling (and not just sharing the results of collocated modeling) using UML or other formal or semiformal languages. Finally, communication tools such as email, mailing lists, on-line meeting facilities, groupware tools and more recently Web 2.0 solutions such as (micro-) blogs and wikis have proven their value to software developers at multiple sites, see e.g., (Damian et al., 2009).

Certain tools support only a (set of) software engineering life-cycle activities: project management, requirements engineering, architecture and design (Capilla et al., 2007; Cataldo et al., 2009), and testing. Yet, no current tool supports all activities necessary for GSD. Lanubile et al. (2010) conclude by stating that “users must (. . . ) prioritize their collaboration needs and the tools to support them” instead of the other way around.

1.3 Research Context

The research presented in this thesis has been performed in participation with two Dutch research projects: GRIFFIN and Stephenson.

The GRIFFIN (a GRId For inFormation about architectural knowledge) project was a four year multi-partner research project. The goal of GRIFFIN was to identify tools, methods, and techniques for managing architectural knowledge. The project involved two universities (VU University Amsterdam and the University of Groningen) and several industrial organizations. The organizations involved range from SMEs to multinationals, and from scientific institutes to IT service providers. In the GRIFFIN research, we had the opportunity to collaborate with two of these partners.

The Stephenson project is a joint project between VU University Amsterdam and an industrial organization. In the project, research in the area of sharing architecture knowledge in a multi-site context was performed within the context of software product line development.

The GRIFFIN project was sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406. The Stephenson project was sponsored by the Dutch “Regeling Kenniswerkers”, project KWR09164.

1.4 Problem Statement

As discussed (1.1.3) the management of knowledge plays an increasingly important role in the discipline of software engineering. One of the most important types of knowledge is knowledge pertaining to the architecture of the system being built. By sharing this architectural knowledge, challenges such as design erosion, high maintenance costs, and lack of information or documentation can be further reduced (Jansen, 2008).
1. Introduction

When compared with collocated software development, global software development poses additional challenges. These challenges are related to temporal, geographical, and socio-cultural distance, as set forth by (Ågerfalk et al., 2005; Holmström et al., 2006). Communication and coordination between various development sites is hampered because of the time difference (limited overlap in time available to synchronize work), socio-cultural distance, and geographical distance. We have elaborated on these challenges in §1.2.2.

The combination of aforementioned two recent developments (the management of architectural knowledge and GSD) may appear to be fruitful; the challenges and issues involved in GSD may be addressed using architectural knowledge management and, conversely, the expected benefits of GSD may be further leveraged using architectural knowledge management techniques. On the other hand, specific architectural knowledge management techniques may be necessary in sharing architectural knowledge across time zones and geographical borders. Likewise, the form in which architectural knowledge is shared should be chosen deliberately to overcome cultural barriers, such as differences in language. In conclusion, we currently lack the insight into how architectural knowledge can be managed in GSD.

1.5 Research Questions

The problem statement as put forward in the previous section provides the motivation for this research: understanding how architectural knowledge management and global software development can co-exist, and how architectural knowledge management can further support the benefits from GSD and address challenges and issues involved with GSD. Hence, we formulate our central research question as follows:

**RQ How can architectural knowledge be managed in a global software development environment?**

Based on the definition of architectural knowledge as provided in §1.1.3, we have further elaborated the use of architectural knowledge within the GRIFFIN project. As a first step towards understanding how architectural knowledge is managed, we pose the research question below. This research question is answered in Chapter 2 of this thesis.

**RQ-1 How is architectural knowledge used?**

For addressing the central research question, we are interested in identifying practices for the management of architectural knowledge in GSD. This directly leads to our second research question.

**RQ-2 What are practices for managing architectural knowledge in a global software development environment?**

The second research question serves as the main contribution of the work presented in this thesis. In our research, we have applied a breakdown of RQ-2 by identifying several distinct areas of research. In performing this research, we collected the results with which we are able to answer RQ-2. We describe each of these research areas in
1.5. Research Questions

turn. As such, the breakdown as provided below provides us with step by step results needed to answer RQ.2.

- First, we identified what practices a typical software development organization involved in GSD uses. Next, we compared these typical practices with the practices used by another organization to further obtain confidence in the necessity or usefulness of these practices. This first study provided us with the insight that primarily architectural knowledge management practices related to the development process were used in the organizations studied. This part of RQ.2 is answered in Chapters 3 and 4 of this thesis.

- While our understanding of the management of architectural knowledge matured, we used the insight obtained to choose an alternative perspective towards the management of architectural knowledge in GSD. This alternative perspective focuses on managing architectural knowledge in GSD by capturing the architectural knowledge in the software product. With software product, we refer to the software architecture, the software itself, and its documentation, following (ISO/IEC, 2000). The software product can be a (possible intermediate) tailor-made solution, since we do not focus on commercial-off-the-shelf components in our research. Hence, we posed an additional question to identify whether or not the quality of software products developed using GSD, can be improved by architectural knowledge stored in the products. This question is answered in Chapter 5.

- As we have learned during the course of our research, the requirements engineering discipline is a well-discussed example of a discipline that becomes challenging in GSD (Damian, 2007; Hsieh, 2006). Furthermore, several solutions have been proposed and validated in the requirements engineering discipline. The similarity between the requirements and architecture and, more specifically, requirements and (architectural) design decisions has been pointed out by (van Vliet, 2008; de Boer and van Vliet, 2009). We have chosen the requirements perspective to identify whether architectural knowledge management can leverage the solutions that are available from the requirements engineering domain to address the challenges in GSD. As such, we identify proposed solutions using this alternative requirements perspective. We validated the usefulness of these proposed solutions at GSD projects that ran an industrial partner in our research and specifically looked into the number of sites involved in these GSD projects. The proposed solutions are described in Chapters 6 and 7.

Practitioners involved in managing architectural knowledge in GSD are only motivated to do so if they are supported adequately. As described in 1.2.6 several supporting solutions are proposed but not yet aimed at managing architectural knowledge. Hence, we identify possible supporting instruments (tools) for the practices that identified in RQ.2. The answer to this question is provided in Chapter 8.

RQ-3 How can architectural knowledge management (practices) in global software development be supported (by tools)?
1. Introduction

To fully understand the applicability of the practices for architectural knowledge management in GSD, we investigate which of our defined practices (see RQ-2) are actually used in GSD practice. The answer to this question is provided in Chapter 9.

**RQ-4 Which practices for managing architectural knowledge are used in a global software development environment?**

Fig. 1.2 illustrates the research questions in context.

![Research Questions Diagram](image)

**Figure 1.2: Research questions for this research**

1.6 Research Methods and Studies

This research was conducted at several industrial partners within the GRIFFIN and Stephenson projects.

All industrial partners where we performed studies are involved in global software development. Furthermore, the organizations were interested in obtaining research results on (how to) better utilize architectural knowledge in their GSD activities.

Although the GSD involvement and the interests of the industrial partners provides for the ability to use results obtained in a certain case study in subsequent studies, we do acknowledge that differences exist between the industrial partners in e.g., industry domain and size of the development organizations. To overcome these differences, we explicitly decided to not make an *a priori* assumption on the development process or