How Software Architects Collaborate: Insights from Collaborative Software Design in Practice

Jae Young Bang, Ivo Krka, and Nenad Medvidovic
University of Southern California
941 Bloom Walk, Los Angeles, California, USA
{jaeyounb, krka, neno}@usc.edu

Naveen Kulkarni and Srinivas Padmanabhuni
Infosys Limited
#44, Electronics City, Hosur Road, Bangalore, India
{Naveen_Kulkarni, Srinivas_P}@infosys.com

***Abstract*—The increasingly complex software systems are developed by globally distributed engineering teams consisting of a number of members who collaborate to gather the requirements, as well as design, implement, and test the system. Unlike other development activities, collaborative software design has not yet been studied extensively, and thus it is not fully understood how it is conducted in practice. We have commenced a series of studies to address this. As the first step, we have interviewed architects at a global software solutions provider to observe how collaborative software design works in practice. In this paper, we report the observations and insights we gained from the interviews related to (1) the various roles of software architects in collaborative software design, (2) the project-specific networks of software architects, (3) the impacts of geographic distribution, and (4) the collaboration cost drivers. We also discuss how we are using these insights to shape up our subsequent research.**

I. Introduction

The complexity and size of modern software projects dictates involvement of multiple software architects during a system's design. These software architects collaborate to make design decisions and produce design artifacts such as architectural documentation. To effectively support these collaborative design practices, it is necessary to understand them intimately.

We are by no means the first to recognize the need to understand collaborative software design (e.g. [1]). However, the issues from the collaboration persist even today while its sophistication has grown significantly.

Researchers have studied a number of topics related to collaborative software design, covering (1) general issues in collaborative software engineering [2]–[9], (2) design model merging and inconsistency checking [10]–[12], (3) design model version control [13]–[16], (4) workspace awareness [17]–[19], and (5) recently introduced collaborative modeling tools [20]–[22] including our own [23].

While potentially useful, it is not clear whether the existing approaches are readily applicable to collaborative design in industrial practice. This is due to the current lack of documented information about how collaborative software design is conducted in practice. Therefore, it is difficult to assess the existing, frequently general-purpose, approaches and their underlying assumptions. The current literature does not fully consider the following attributes of collaborative software design: (1) the different types of stakeholders who are involved in the process of making design decisions, (2) how those stakeholders are distributed and how they are connected to one another, (3) the types of artifacts that are created and exchanged, and (4) how collaborative software design differs from the other collaborative software development activities (e.g., implementation). Recent research efforts shed some light on these issues. Unphon et al. reported how software architects manage large software architecture and exchange architectural changes [24], and Figueiredo et al. further explored different roles of software architects [25]. Nevertheless, the collaborative aspects of software design has not been fully discovered.

To have deeper understanding of collaborative software design practices, we have conducted a set of 18 interviews with software architects from a large global software solutions provider. This, in turn, enabled us to explore and clearly define research problems that are specific to the domain of collaborative software design. Furthermore, we have conducted two additional *cross-validation* interviews with architects at different software companies for the purpose of obtaining preliminary indication whether our findings from the 18 interviews were observable from other organizations as well.

In this paper, we describe the insights we gained from the interviews. The main insights were related to:

1) *Different roles and collaboration patterns of software architects and other stakeholders*. For example, there are roles of software architects that are not properly differentiated and thus are supported inadequately by existing approaches.
2) *Distinct topologies of software architects*. For example, we discovered that architects responsible for detailed design may need to collaborate more than the others.
3) *Geographic distribution of the software architects*. For example, the interviewees suggested that geographic distribution has a limited impact on collaboration.
4) *Factors that drive costs in collaborative software design*. For example, the architects were aware that the way design tasks are allocated impacts the subsequently required collaboration.

This paper focuses on the first of a series of planned studies targeting collaborative software design. We report on the preliminary interviews targeted at understanding the current practices. In turn, this work will serve as the foundation for (1) a pilot survey to evaluate the research questions obtained in the process, and (2) a large-scale survey for the full evaluation of the research questions. Although based on a preliminary set of interviews, we posit that our insights and their illustration through the examples add to the current body of knowledge, and should direct future research efforts.

The rest of the paper is structured in the following manner. In Section II, we describe the research setup. In Section III, we describe and summarize the interviews as well as the insights we gained. We describe the cross-validation interviews we conducted in Section IV. In Section V, we discuss the threats to validity. We conclude the paper in Section VI.

II. Research Setup

A. Interview Questions

The ultimate goal of this research is to improve the understanding of collaborative software design in industrial practice. We had interviews to identify the stakeholders involved in collaborative design, to analyze how collaboration among those stakeholders occurs, and to identify the sources and extent of the costs incurred by collaborative design. The interviews were hour-long and free-form with topics including but not limited to:

- What kind of project did you participate most recently?
- What is the structure of your team (authority, size, location, etc.)? Who are the stakeholders involved in collaborative design? Draw a diagram of the network of the stakeholders.
- What is your role in the team? What are the roles of the other stakeholders?
- How is each design task formed and assigned? What are the criteria used for design task assignment?
- What artifacts are created and how are they exchanged?
- In which situations do you need to collaborate?
- What communication media do you use to exchange design decisions or knowledge, especially with people in remote locations?
- What are the main factors that affect collaborative design cost?
- What is the overall collaborative software design process in your current project?

Based on the answers to these general preliminary questions, we followed up by further exploring the most important or unclear aspects of the interviewees' responses.

B. Target Organization

We conducted the interviews at a very large, global software solutions company with long history of developing services for clients world-wide. They provide services in various domains such as finance, energy, telecommunication, and aerospace. Their software projects are of different types such as systems integration, product development, and maintenance projects, and of different sizes ranging from small ones for a few months to large ones that take multiple years by a large group of people. This company is mature (CMMI level 5, 7 billion USD revenue, 155K employees in 2012).

C. Interview Execution

The first and second authors (PhD students) spent 10 weeks at the primary target to interview 18 highly experienced software architects who have participated in commercial software development projects for over 12 years on average and more than 9 years at the target organization. The fourth author (a researcher at the target organization) also participated in the interview sessions. We randomly selected software architects from different projects conducted in various domains as the interviewees. By selecting interviewees from diverse backgrounds, we tried to ensure that we gain general rather than domain-specific insights. For similar reasons, we aimed to select interviewees of different backgrounds in terms of the types of projects on which they have worked (e.g., greenfield vs. brownfield, or system integration vs. product development vs. customization vs. maintenance). About two thirds of the interviews were done face-to-face, and the rest were done over conference calls. All interview responses were noted, and 12 of those were also voice-recorded. 15 interviewees were senior architects while 3 had junior architect roles. 14 interviewees were involved in outsourcing projects while 4 were involved in in-house projects.

III. Observations and Insights

In this section, we summarize the most important insights supported by the interview responses. We also discuss the practical impact of our findings. The names, titles, and examples have been modified from those used at the target organization to be suitable for public disclosure.

It is worth noting that this work is the preliminary step of a larger study and thus the purpose of this work is to form research questions rather than to prove them. We discuss the limitations of our observations and insights that are drawn from the interviews as part of the threats to validity of this work (Section V).

We outline the different architects' roles in Section III-A. In Section III-B, we explain the common software architect topologies and how different types of software projects alter these topologies. We discuss the perceived impact of geographic distribution in the context of collaborative software design in Section III-C. In Section III-D, we list the drivers of collaboration costs as elicited from the interviewees.

A. Various Roles of Software Architects

Lane et al., in their previous work [6], identified four roles of global collaborative software engineering: Business Analyst, Designer, Developer, and Development Support.

Of these roles, the stakeholders playing the general Designer role (which we refer to as "software architect" in our work) are supposed to collaboratively create a system's design. One observation from our interviews was that multiple software architects participating in collaborative design play different roles despite having similar job titles. Furthermore, in certain situations the development personnel can assume some design responsibilities. This observation aligns with a recent report by Figueiredo et al. that software development organizations adapt roles to their reality [25].

The job titles of the software architects at the target organization were dependent on their experience. In this paper, we will call an architect with relatively more experience a *senior architect* and the less experienced architect a *junior architect*.

The senior architects are primarily responsible for making high-level design decisions. Each senior architect in a software design project makes decisions related to determining the modules, devising a data-flow architecture, deciding on COTS usage, system integration, product delivery, and so on. In a collaborative effort with multiple senior architects, each architect becomes responsible for only a subset of these decisions. Hence, each senior architect has a relatively tightly scoped role as opposed to general high-level decision making.

The junior architects are primarily responsible for detailed design and modeling. In the interviews, we observed that each junior architect's role is further refined: a junior architect is assigned to projects and design tasks based on her/his technical background (e.g., Java, databases, web technologies).

Our interviews also suggested that, depending on the project specifics such as the size of the project and the structure of the design team, a *senior developer* who leads a team of developers can assume a subset of a junior architect's role and participate in making detailed-level design decisions. The junior architects and the senior developers do not assume the responsibilities of senior architects.

"We would have the senior developers on the call, but more on a listening mode, so that they are getting the context and starting to see the big picture rather than asking questions and doubts."

In other words, the senior developers only observe the collaborative high-level design sessions in order to get familiar with the system and its requirements.

Impact. Identifying the different roles of software architects may open new research avenues in the context of collaborative design. Specifically, particular roles could have more or less prominent collaboration issues. Focusing on roles that have greater collaboration challenges is more valuable [6]. For example, an architect responsible for devising a data-flow architecture may need to communicate with other architects more frequently than an architect who is deciding on an appropriate COTS.

B. Software Architect Topologies

In this subsection, we identify the common ways multiple architects work together during a software system's design. Our descriptions are based on the four topologies of collaboration paths between software architects and other stakeholders that are directly drawn by the interviewees. These topologies are depicted in Figures 1–4.

This topology-based representation allows us to clearly capture with whom the different software architects directly collaborate in their respective responsibilities. The distinctions between the topologies also provide insights into how collaborative design differs for different types of projects as well as how the different stakeholders are geographically distributed. Though common, these topologies were not standardized and varied greatly depending on the available resources and project specifics.

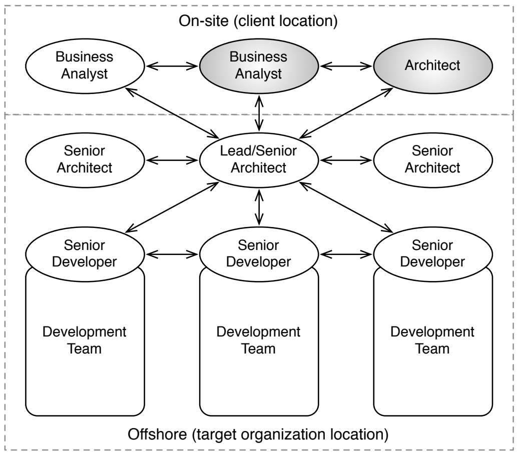

Fig. 1. Topology for Mid-Sized Projects

To help understand the topologies, we will first define some terminology. We refer to the entire delivered software product as the *system*, which is then decomposed into *subsystems*, and, in turn, into one or more *modules*. The system architecture consists of the high-level design decisions. For example, if a web portal is being designed, the high-level design decisions would include the definition of the services provided by the portal, the definition of elements shared by these services (e.g., a back-end database), and how those components are supposed to interconnect. Each service and shared component would then be a subsystem of the web portal system. The module architecture consists of detailed design decisions such as creating a class diagram defining implementation-level objects that will be used to implement a module in a subsystem.

Figure 1 depicts the most common topology of software architects at the target organization that is used for mid-sized projects. Gray ovals represent stakeholders from third-party organizations that contracted the system, while white ovals represent the stakeholders who belong to the development organization. The arrows represent the channels of direct collaboration. In this topology, the lead senior architect initially receives the set of requirements from the business analysts. The lead architect then proceeds to come up with the high-level system design together with other senior architects. When the system's high-level design is complete, the identified modules are assigned to the senior developers for detailed design. The conventional software engineering wisdom suggests that the senior architects are supposed to be actively involved in other phases of a project after the design is completed. This view was reinforced by some of the interviewees.

"Solution architect (senior architect) initially puts more effort in the initial stage until the solution architecture is complete, and plays the reviewer role in later stages."

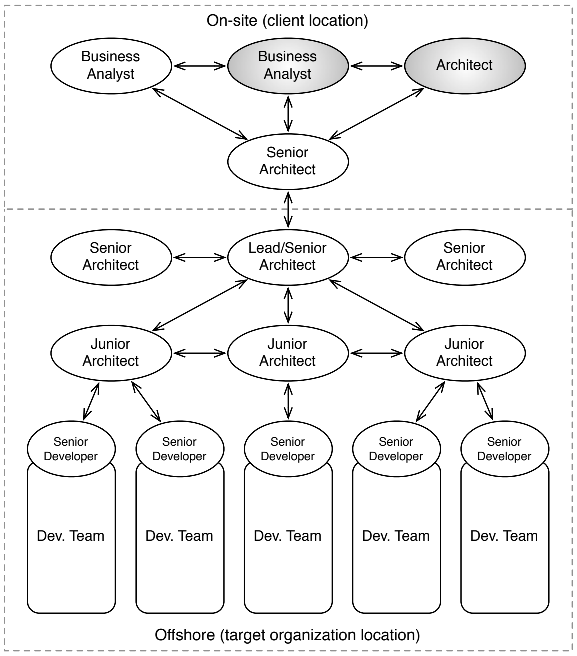

Fig. 2. Topology for Large-Scale Projects

This statement may suggests reduced but sizable role in later phases. Somewhat surprisingly, however, we found that senior architects, who have the most end-to-end knowledge about the system requirements and the overall design, typically have very limited participation in the detailed design. At this stage, their involvement in the project amounts to propagating requirement changes and helping to resolve major inconsistencies or conflicts between modules. In further contrast to conventional wisdom, the role of software architects after the design is complete was vividly described in another response:

> *"Architects make sure the development is following the architecture looking at the final code, (but) in the ideal world."*

The interview responses suggested that larger-scale development projects at the target organization frequently utilize the topology depicted in Figure 2. In this topology, there is an additional senior architect between the lead architect and the outside stakeholders. The role of this architect is to explicitly manage the collaboration between the design team and the customer-side stakeholders. Moreover, there is an additional layer between the lead architect and the development teams compared to the simpler topology shown in Figure 1. In the lower layers that are responsible for detailed design, the junior architects manage the design of the subsystems, while the senior developers design the modules of the subsystems and manage the subsequent development.

In case the systems are designed in explicit collaboration with the client-side architects, the topologies change. Figure 3 depicts the topology for a software project where the lead architect is from the client-side. The on-site senior architect in this topology is acting as a liaison.

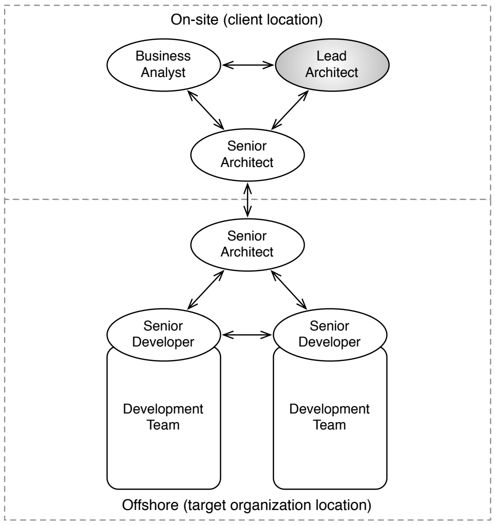

Fig. 3. Topology with an External Lead Architect

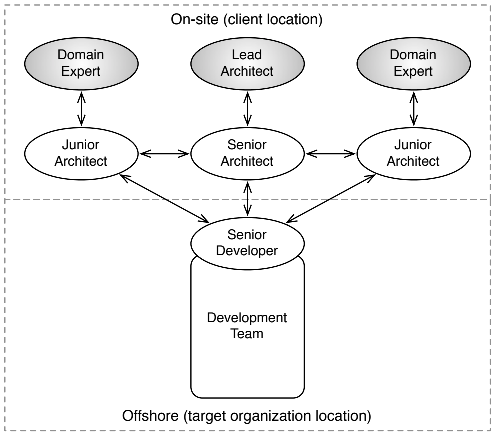

Fig. 4. Topology for High Criticality Projects

Figure 4 depicts the topology for a software project with high criticality so that all architects are relocated to the client location. The distinction of the topology in Figure 4 is that it allows junior architects to directly communicate with the client-side domain experts who have knowledge about the system requirements and the domain.

While the above topologies have notable differences and are employed on different types of projects, they share some common traits. First, senior architects, other than the lead architect, have specific duties such as creating data-flow architecture and project-delivery schedule (recall Section III-A). Second, junior architects and senior developers communicate upstream only via the lead architect, with the exception of the topology in Figure 4. For example, if a senior developer had a query regarding the project delivery schedule, s/he would first raise an issue with the lead architect who would in turn communicate with the senior architect responsible for product delivery. Lastly, the geographic distribution of the architects can vary. Lead architects can change locations across the border between on-site and offshore. Furthermore, during preliminary design, the senior architects typically gather at the same location for intensive collaborative design; we elaborate on the impact of geographic distribution in Section III-C.

We also found that different types of software projects may change the way collaboration occurs. The above topologies were for IT service projects. In case of a product development project at the target organization, the topology would be largely different. One interviewee who was a senior architect of a product development team said that the way his team members collaborate was more ad-hoc, and the overall development process of choice was primarily agile.

Impact. Improved understanding of architect topologies may affect the agenda of collaborative design research. According to conventional software engineering wisdom, we see the critical importance of collaboration between senior architects due to the great impact of the high-level design decisions made by them. However, we hypothesize that major opportunities for research on collaborative design may actually reside at the lower levels of the topologies. The reason for this is that collaboration at the higher level is often immediate, frequently in-person, and involves experienced stakeholders who work on a manageable number of high-level design decisions. By contrast, the junior architects and senior developers are less experienced, have a larger number of tasks, and need to handle a plethora of low-level details and dependencies. This leads to a larger number of potential errors, conflicts, and misunderstandings that may require more voluminous collaboration. New and better methods, techniques, and tools are likely to be needed to facilitate that collaboration.

C. Geographic Distribution and Its Impacts

Geographic distribution has been seen as one of the most important challenges in global software engineering [26]. Our interview responses corroborated that knowledge sharing and context transfer between distributed software architects were non-trivial. However, surprisingly, the responses also suggested that geographic distribution is not as prominent of a challenge as expected in case of software design.

The reasons for this are the two remedies the target organization applied: (1) relocation of some architects to minimize the negative impacts of the geographic distribution and (2) extensive effort in high-level design up front that reduces the amount of collaboration required for detailed design.

In the context of the four architect topologies from Section III-B, the architect relocation primarily happens at the upper levels of the topologies. By moving senior architects to the client location early in the software project, it is possible to more actively gather and clarify requirements as well as come up with an appropriate high-level architecture, which can be immediately explained to the client.

> *"Communication and collaboration have to happen frequently especially heavily in the initial stage of the project. The first 1/3 of the time is put into the frequent meetings."*

In the scenario of Figure 1, the lead architect could be relocated to the client location in the early phase of software design to facilitate the intensive collaboration, and then moved back to the target organization location to be collocated with the senior developers at a later time.

Moreover, several architects may be relocated if the criticality of the project is higher, as depicted in Figure 4. The feasibility of relocating the architects stems from their relatively low number in comparison to developers, while providing additional benefits in terms of direct interaction with the clients.

> *"Architects sometimes need to travel to be collocated when the complexity of the current task is very high."*

Contrary to the senior architects, junior architects and the senior developers who lead the development teams are likelier to be geographically distributed. The interviewees suggested that the collaboration cost created by the distribution between these stakeholders was perceived as not as high due to the modularized design tasks that require limited synchronization.

> *"Spending more effort on architecture documentation to the very much detailed story lines and to the level of a single transaction would prevent the need for large amount of effort in later stages, and this is what is happening today."*

We hypothesize that this perception might be partly due to the fact that the stakeholders are distributed across only a few time zones (client and target organization's location). Most of the target organization's personnel are stationed in the same country that spans only one time zone. Hence, the junior architects are able to "just make a phone call" whenever necessary.

Impact. Most of the senior architect interviewees did not see geographic distribution as problematic, which does not align with the conventional wisdom of software engineering. Although further study is required, we suspect their responses might be influenced by their collocation, which eliminated the distribution in the first place. In addition, the interviewees with junior architect roles were typically situated in different cities within a single time zone. In that sense, while the organization is indeed a global provider of software solutions, our data indicates that its software development strategy is much more localized. For these reasons, we posit that future research should be conducted to determine the full effects of wider distribution of software designers, both geographic and temporal.

D. Collaboration Cost Drivers

One of the objectives of our interviews was to discover a list of potential drivers of collaboration costs in software design. While our expectation was that these drivers would be largely implicit, the software architects we interviewed were keenly aware of a number of drivers. They in fact suggested a list of drivers that were perceived as having a significant impact on collaboration. In this section, we list the drivers along with the rationales suggested by the interviewees; we see this list as a starting point for future exploration of costs incurred by collaboration during software design. We posit that some of the drivers may be correlated with each other.

- **External dependency**: It is generally accepted that having a dependency on resources (e.g. modules, data, personnel) that are owned and deployed outside the development organization increases the collaboration costs. Even for cases when the client has full ownership of such a resource, we learned from our interviews that costs related to utilizing such external resources are often significantly higher than expected. For example, the client may require a complicated procedure or on-site presence to access the resource for privacy reasons, which leads to long wait times, although the same client required this system integration in the first place. As a result, some software architects saw external dependencies to already existing external modules owned by the client as potentially costlier, in terms of required collaboration, than dependencies to modules that are developed from scratch.
- **Internal Module dependency**: The interviewees corroborated the conventional software engineering wisdom regarding high coupling: an increasing number of module dependencies requires more collaboration. The way modules depend on each other may also affect the task allocation.
- **Task allocation**: Task allocation is concerned with defining and mapping design tasks to specific junior architects or senior developers.
  The interviewees considered that an improved task allocation results in less collaboration, which is aligned with the previous research [8], [27]. The combined criteria software architects use during task allocation include the properties of the modules (i.e. size, complexity), the dependencies between modules, and the technical background and geographic distribution of the junior architects. Several interviewees, however, noted that the selection criteria can be more direct in practice.
  > *"Assigning senior developers on module design and development is primarily based on the skillset the senior developers have."*
- **Advanced collaboration tools**: The quality of software development in general and software design in particular is often dependent on the availability and quality of appropriate tool support. Expectedly, the interviewees saw benefits in using advanced technologies during collaborative design. However, the interviewees noted several major obstacles to adopting new collaboration tools, including high training costs and support for only limited types of collaboration. Further, some interviewees did not find appropriate collaboration tools for their needs. In several cases, however, they came up with creative ways to use widely available general-purpose technologies. For example, one used a "doodle" for inter-module interface design, and another one developed communication media based on online tools such as wiki or special interest group (SIG) forums. This use of various on-line tools has also been observed at other organizations [24].
- **Collaboration experience of software architect**: As intuitively expected, the interviewees considered that prior collaboration experience of a software architect decreases the collaboration costs.
  Collaboration experience of the lead architects can affect the overall collaboration on the project as well as other drivers such as task allocation:
  > *"One of the responsibilities of a senior architect is to facilitate communication between the junior architects to establish common perspective with minimal necessity of integration points."*
- **Technical expertise of software architect**: The interviewees believed that lack of technical and domain expertise tends to increase the collaboration costs. Software architects without enough expertise may spend more time on collaborating with other software architects than those with more extensive backgrounds. According to our interviews, having such challenges is considered as a bottleneck of collaborative software design. One interviewee suggested that having a technical consulting team inside the organization can reduce this bottleneck.
- **Mature development process**: The collected interview responses suggested that applying a mature software development process reduces the required collaboration. For example, a mature software development process clearly defines the developers' roles and communication patterns. Thus, the process guides and sometimes "forces" the stakeholders to produce relevant artifacts at the right moments to prevent unnecessary future collaboration.
- **Criticality of project**: The interviewees consider that projects with high criticality tend to require more collaboration. During the design of those projects, more frequent meetings and discussions would happen to prevent any misinterpretation of design artifacts or miscommunication between architects, and to identify and resolve inconsistencies and conflicts early.
- **Time pressure**: The interviewees mentioned that, in cases of high time pressure, early rigorous planning is given lower priority due to the urgency to complete the current tasks. At the same time, they also agreed that this might leave higher risk of resulting in problems during integration.

Impact. We consider that the above list of cost drivers provides a useful way of classifying and possibly directing future research efforts — the new collaboration-support techniques should target particular drivers. It is important to note that the current list was suggested by the architects themselves. Hence, there may exist additional implicit drivers; discovering these requires further research including shadowing architects during collaborative design and analyzing data from completed projects. Furthermore, the exact impact of these drivers is yet to be validated and quantified. In addition, the discussions of the drivers with the interviewees revealed creative solutions employed by software architects to reduce collaboration costs.

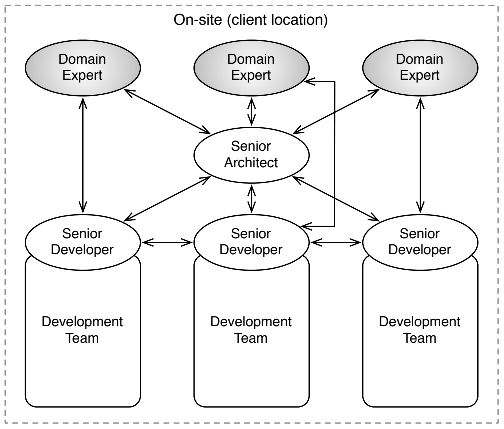

Fig. 5. Topology of Validation Interview 1

We believe that analyzing and documenting those solutions may further inform and advance collaborative design research.

IV. Cross-Validation Interviews

Once we completed the 18 interviews and collected the insights, in order to assess our findings, we conducted preliminary *cross-validation interviews* with software architects outside the target organization. We selected two interviewees with an average of seven years of industry experience at two different companies, both of which focus on IT services as well as software product development. Those companies have their headquarters in a different country and are of similar size and maturity (CMMI level 5, over 1 billion USD revenue, over 10 thousand employees) as the original target organization (recall Section II-B). The interview questions and overall process were unchanged.

The aim of these additional interviews was not to get a sample that would be large enough to draw statistically significant conclusions about the collaborative design practices in the two companies. Instead, we wanted to obtain preliminary indication whether the experience of the architects at the two additional companies resembled the experience of the 18 original interviewees.

Although the two companies had different attributes (location, market) as compared to our primary target organization, the collected responses suggested that all three companies shared many commonalities in terms of the collaboration characteristics and issues during collaborative software design. For example, the additional interviews corroborated the following:

- **Architect roles.** The roles of the software architects differ and can be highly specialized.
- **Topologies.** The architect topologies reported in the two cross-validation interviews resembled the topologies from Figure 2 and Figure 4. For example, one of the interviewees drew the topology depicted in Figure 5. In this topology, the junior architects can communicate directly with the client-side domain experts in order to facilitate requirements understanding, analysis, and revision.
- **Distribution.** The cross-validation interviews confirmed that, in practice, potential collaboration issues stemming from distribution are often mitigated by avoiding distribution in the first place. For example, in case of the topology from Figure 5, all of the architects as well as the development teams were moved to the client location for the duration of the project.
- **Cost driver awareness.** We found that the two interviewees were keenly aware of several cost drivers (recall Section III-D). In particular, the interviewees suggested that (1) a lack of *domain knowledge* is a prominent cause of collaboration issues and (2) unprecedented projects tend to have higher collaboration cost; note that both of these relate to the *technical expertise* cost driver from Section III-D. Similarly, the architects agreed that internal dependencies increase collaboration costs, while suggesting that dealing with those dependencies should not be deferred to later stages of a project's life cycle. The following quotes were obtained from the two interviews (translated from Korean):

  "Teams tend to develop their own components for a set of functionalities that are shared by different jobs rather than coordinate to develop and share them together (due to high collaboration cost)."

  "We often face inconsistencies between components developed by different engineers in later stages. Half of the cases lead to full-scale reverting to earlier stages, and local patches are made for the other half."

V. Threats to Validity

In this section, we explain the limitations of our insights with respect to the fact that they stem from a preliminary set of exploratory interviews.

A. Representativeness

The results presented in this paper are based on a set of collected responses and anecdotes from the interviews rather than a set of statistically significant findings from a systematically designed survey. Therefore, our insights may not be representative of the practices at the target organization as a whole, and more generally, of the current industrial practices. However, this was not the intended purpose of our interviews: with these exploratory free-form interviews we aimed to obtain insights that would help us, as well as other researchers, to form informed hypotheses for future empirical research. In addition, while a set of 18 interviews could be sufficiently large for a study of a few variables, we acknowledge that additional variables with impact on collaboration may yet to be identified due to the limited number of samples. As a mitigating factor to the above limitations, we are currently conducting a large-scale survey to verify the insights and to further explore the practices in collaborative software design.

B. Generality

As discussed above, the 18 interviews were conducted at a single organization, which potentially limits the generality of the outcomes in the context of broader industrial practices.

However, there are three mitigating factors that strengthen the general value of our conclusions:

- **Organization expertise**: Our target organization is a very large software company that is considered as a mature software development organization. We thus believe it applies practices that are at least somewhat overlapping with those used by the competitors. Furthermore, considering the target organization's maturity (e.g., CMMI level 5 organization), its collaboration process can be considered the state-of-the-practice. Hence, we expect that the less-capable software organizations face collaboration issues that are equal or even more prominent than those found at our target organization.
- **Project coverage**: The interviews we conducted covered software architects whose experience differed in terms of their project types and project domains. In particular, we selected architects who worked on various types of software development projects, and also covered various application domains. This variation serves the purposes of obtaining findings that apply more generality regardless of the project type and application domain.
- **Cross-validation interviews**: By having the cross-validation interviews outside the target organization, we found that our insights were not limited to the target organization but something that could be observable from other organizations.

VI. Conclusions

In this paper, we present the results of a set of interviews we conducted with experienced software architects who are working at a large global software solutions provider. The goal of the interviews was to observe how collaborative software design occurs in practice. We presented the insights we gained from the interviews; these insights point to necessity of further research and suggest several potentially fruitful topics in this area. First, future research should target the specific architect-roles that require greater collaboration. Second, the bottlenecks of software design, extractable from the identified topologies, can suggest areas which would maximize the reduction of collaboration costs. Third, the remedies employed at the target organization to reduce the impact of geographic distribution should be verified, while further studies are required to reveal additional opportunities. Finally, the collaboration cost drivers identified by the software architects suggest promising categories of future research efforts to reduce collaboration costs.

As we look to the future, although software design is an activity that highly relies on "soft skills" that is not readily documentable nor formalized as a general structure, we aim to achieve a clearly defined collaborative software design process. Such a process will aid software architects to work better in concert and optimize their tasks, leading to reduced costs and higher quality of the resulting software systems. We also believe that such a process will guide the less experienced software architects to quickly grasp important concepts and rapidly improve their expertise.

Acknowledgments

This work has been supported by Infosys Limited.

References

[1] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.

[2] A. Begel, N. Nagappan, C. Poile, and L. Layman, "Coordination in Large-Scale Software Teams," *Proceedings of CHASE*, pp. 1–7, 2009.

[3] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software Dependencies, Work dependencies, and Their Impact on Failures," *IEEE TSE*, vol. 35, no. 6, pp. 864–878, 2009.

[4] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, Dependencies, and Delay in a Global Collaboration," *Proceedings of CSCW*, 2000.

[5] A. Lamersdorf, J. Münch, A. F. V. Torre, C. R. Sánchez, and D. Rombach, "Estimating the Effort Overhead in Global Software Development," *Proceedings of ICGSE*, pp. 267–276, 2010.

[6] M. T. Lane and P. J. Ågerfalk, "On the Suitability of Particular Software Development Roles to Global Software Development," *ICGSE*, 2008.

[7] A. Mockus and J. D. Herbsleb, "Expertise Browser: A Quantitative Approach to Identifying Expertise," *Proceedings of ICSE*, 2002.

[8] S. Mohan and J. Fernandez, "Distributed Software Development Projects: Work Breakdown Approaches to Overcome Key Coordination Challenges," *Proceedings of ISEC*, 2010.

[9] A. Sarma, D. Redmiles, and A. van der Hoek, "Categorizing the Spectrum of Coordination Technology," *IEEE Computer*, 2010.

[10] X. Blanc, I. Mounier, A. Mougenot, and T. Mens, "Detecting Model Inconsistency through Operation-Based Model Construction," *Proceedings of ICSE*, pp. 511–520, 2009.

[11] A. Egyed, "Automatically Detecting and Tracking Inconsistencies in Software Design Models," *IEEE TSE*, vol. 37, no. 2, pp. 188–204, 2010.

[12] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, "xlinkit: A Consistency Checking and Smart Link Generation Service," *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 151–185, 2002.

[13] K. Altmanninger, M. Seidl, and M. Wimmer, "A Survey on Model Versioning Approaches," *International Journal of Web Information Systems*, vol. 5, no. 3, pp. 271–304, 2009.

[14] M. Koegel, M. Herrmannsdoerfer, J. Helming, and Y. Li, "State-based vs. Operation-based Change Tracking," *Proceedings of MODELS*, 2009.

[15] S. Maoz, J. O. Ringert, and B. Rumpe, "ADDiff: Semantic Differencing for Activity Diagrams," *Proceedings of ESEC/FSE*, 2011.

[16] T. N. Nguyen, E. V. Munson, and J. T. Boyland, "An Infrastructure for Development of Object-Oriented, Multi-Level Configuration Management Services," *Proceedings of ICSE*, 2005.

[17] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, "FASTDash: A Visual Dashboard for Fostering Awareness in Software Teams," *Proceedings of CHI*, pp. 1313–1322, 2007.

[18] Y. Brun, R. Holmes, M. D. Ernst, and D. Notkin, "Proactive Detection of Collaboration Conflicts," *Proceedings of ESEC/FSE*, 2011.

[19] A. Sarma, D. F. Redmiles, and A. Van Der Hoek, "Palantír: Early Detection of Development Conflicts Arising from Parallel Code Changes," *IEEE TSE*, vol. 38, no. 4, pp. 889–908, 2012.

[20] M. Cataldo, C. Shelton, Y. Choi, Y. Y. Huang, V. Ramesh, D. Saini, and L. Y. Wang, "Camel: A Tool for Collaborative Distributed Software Design," *Proceedings of ICGSE*, pp. 83–92, 2009.

[21] H. K. Dam and A. Ghose, "An Agent-based Framework for Distributed Collaborative Model Evolution," *Proceedings of IWPSE-EVOL*, 2011.

[22] N. Mangano and A. Van Der Hoek, "The Design and Evaluation of a Tool to Support Software Designers at the Whiteboard," *Automated Software Engineering*, pp. 1–41, 2012.

[23] J. Bang, D. Popescu, G. Edwards, N. Medvidović, N. Kulkarni, G. M. Rama, and S. Padmanabhuni, "CoDesign – A Highly Extensible Collaborative Software Modeling Framework," *Proceedings of ICSE*, 2010.

[24] H. Unphon and Y. Dittrich, "Software architecture awareness in long-term software product evolution," *Journal of Systems and Software*, vol. 83, no. 11, pp. 2211–2226, 2010.

[25] M. Figueiredo, C. de Souza, M. Pereira, J. Nicolas Audy, and R. Prikladnicki, "On the Role of Information Technology Systems Architects," *Proceedings of AMCIS*, 2012.

[26] N. Ramasubbu, M. Cataldo, R. K. Balan, and J. D. Herbsleb, "Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits," *Proceedings of ICSE*, 2011.

[27] A. Lamersdorf and J. Münch, "TAMRI: A Tool for Supporting Task Distribution in Global Software Development Projects," *Proceedings of ICGSE*, pp. 322–327, 2009.