Using Social Network Analysis to Investigate the Collaboration Between Architects and Agile Teams: A Case Study of a Large-Scale Agile Development Program in a German Consumer Electronics Company

Ömer Uludağ(✉), Martin Kleehaus, Soner Erçelik, and Florian Matthes

Technische Universität München (TUM),
85748 Garching bei München, Germany
{oemer.uludag,martin.kleehaus,soner.ercelik,matthes}@tum.de

Abstract. Over the past two decades, agile methods have transformed and brought unique changes to software development practice by strongly emphasizing team collaboration, customer involvement, and change tolerance. The success of agile methods for small, co-located teams has inspired organizations to increasingly use them on a larger scale to build complex software systems. The scaling of agile methods poses new challenges such as inter-team coordination, dependencies to other existing environments or distribution of work without a defined architecture. The latter is also the reason why large-scale agile development has been subject to criticism since it neglects detailed assistance on software architecting. Although there is a growing body of literature on large-scale agile development, literature documenting the collaboration between architects and agile teams in such development efforts is still scarce. As little research has been conducted on this issue, this paper aims to fill this gap by providing a case study of a German consumer electronics retailer's large-scale agile development program. Based on social network analysis, this study describes the collaboration between architects and agile teams in terms of architecture sharing.

Keywords: Large-scale agile development · Social network analysis · Agile architecture

1 Introduction

Emerging in the 1990s, agile methods have transformed and brought unprecedented changes to software development practice by strongly emphasizing change tolerance, continuous delivery, and customer involvement [1,2]. With these agile methods, self-organizing teams work closely with business customers in a single-project context, maximizing customer value and quality of delivered software

P. Kruchten et al. (Eds.): XP 2019, LNBIP 355, pp. 137–153, 2019.
https://doi.org/10.1007/978-3-030-19034-7_9

product through rapid iterations and frequent feedback loops [1]. The success of agile methods for small, co-located teams has inspired enterprises to increasingly apply agile practices to large-scale endeavors [2,3]. Since the initial application of agile methods was originally intended for small, co-located teams, many organizations are uncertain how to introduce them at scale and therefore face new challenges such as inter-team coordination, dependencies to other existing environments or distribution of work without a defined architecture [1,4,5]. The latter is also the reason why large-scale agile development has been subject to criticism since it neglects detailed assistance on software architecting [2,6]. Agile methods assume that architecture should evolve incrementally rather than being imposed by some direct structuring force (emergent architecture) [7]. However, the practice of this design is effective at team level but insufficient at large-scale. It causes excessive redesign efforts, architectural divergence, and functional redundancy increasing a system's complexity [7,8]. Therefore, an intentional architecture is required, which embraces architectural guidelines that specify inter-team design and implementation synchronization [7,9]. The effective evolution of a system's architecture requires the right balance of emergent and intentional architecture and a close collaboration between architects and agile teams [7,9,10].

Literature describing the collaboration between architects and agile teams in large-scale agile development is still scarce. This paper aims to fill this gap by providing a case study of a German consumer electronics retailer's large-scale agile development program. Based on this objective, our research question is:

How does the collaboration take place between architects and agile teams in a large-scale agile development program?

The remainder of this paper is structured as follows. In Sect. 2, we provide an overview of foundations and related works. In Sect. 3, we present the research approach of this paper. Section 4 describes the case study on the collaboration between architects and agile teams in the large-scale agile development program. We discuss our lessons learned in Sect. 5 before concluding the paper with a summary of our results and remarks on future research in Sect. 6.

2 Background and Related Work

In the following, the Scaled Agile Framework and Spotify Model are introduced, as the observed program has adopted these two scaling frameworks. Thereafter, the concept of communication networks is presented, which is essential for interpreting the results of the social network analysis in Sect. 4.

2.1 Scaled Agile Framework

The Scaled Agile Framework (SAFe), a widely used scaling framework [11], was first published by Dean Leffingwell in 2011. SAFe builds on existing lean and agile principles that are combined into a method for large-scale agile projects.

It provides a soft introduction to the agile world as it specifies many structured patterns. This introduction is needed for organizations moving from traditional to agile development environment [7]. The latest SAFe 4.6 version supports four out-of-the-box configurations: *Essential SAFe*, *Large Solution SAFe*, *Portfolio SAFe*, and *Full SAFe*. As the observed program uses Essential SAFe, we will subsequently focus on this. Essential SAFe is the simplest entry point for implementing SAFe and consists of team and program levels [7]. At team level, the techniques outlined are those used in Scrum. Each team consists of five to nine members, one scrum master (SM), and one product owner (PO). All teams are part of an agile release train (ART), a team of agile teams that delivers a continuous flow of incremental releases. Each team is responsible for defining, building, and testing stories from its team backlog in a series of two-week iterations using common iteration cadences [7]. At program level, the product management (PM) serves as the content authority for the ART and is accountable for identifying program backlog priorities. The PM works with POs to optimize feature delivery and direct their work at team level. A release train engineer (RTE) facilitates program execution, escalates impediments, manages risk, and helps to drive continuous improvement [7]. The system architect has the technical responsibility for the overall architectural design of the system and aligns the ART with the common technical and architectural vision [7].

2.2 Spotify Model

In 2012, Kniberg and Ivarsson [12] published Spotify's approach to scale agile methods over 30 teams across three cities. The Spotify Model emphasizes the importance of "aligned autonomy", i.e. the autonomy of agile teams with simultaneous collaboration and coordination to achieve the same goals. The basic unit of development is called a *Squad*, which is similar to an agile team in SAFe. Squads are self-organizing and autonomous teams that have all the skills to design, develop, test, and release for production. A *Tribe* is designed as a collection of squads working in related areas (correspondents to an ART in SAFe). Squads within a tribe are co-located. People with similar skills in the same competency area within the same tribe form a *Chapter*. A *Guild* is a community of people that share same interests and often includes all chapters working in this area (complies with a community of practice in SAFe) [12].

2.3 Communication Networks

According to Guo and Sanchez [13], communication is understood as the creation or exchange of thoughts, ideas, and emotions between senders and receivers. Communication can be decomposed into two types: inter-team and intra-team communication. The former stands for communication between several teams, the latter for communication within a team [14]. The flow of communication connecting senders and receivers are called communication networks [15]. Figure 1 depicts five common communication networks. The *wheel network* is the most centralized network pattern. In this network, each member communicates with

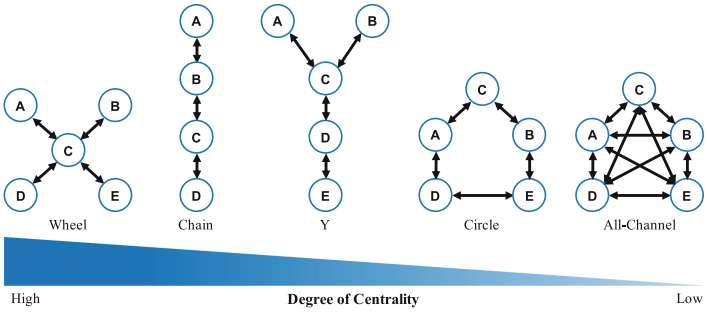


Fig. 1. Common communication networks [15]

only one other person. The superintendent C receives all the information from his subordinates A, B, D, and E and sends back information, usually in the form of decisions. The *chain network* is the second highest in centralization. Only two people communicate with each other, and they have only one other person to communicate with. The *Y network* is similar to the chain network except that two members are out of the chain. In the Y network, members A and B can send information to C but they cannot receive information from anyone else. Members C and D can exchange information. Member E can exchange information with member D. The *circle network* stands for horizontal and decentralized communication, which offers equal communication possibilities for every member. Each can communicate with one other to his right and left. Members have identical restrictions but the circle is a less restricted condition than the wheel, chain, or Y network. The *all-channel network* is an extension of the circle network and connects everyone in the circle network, as it permits each member to communicate freely with all other persons [15].

2.4 Agile Architecture

Angelov et al. [16] describe the role of architects and challenges they face in Scrum such as insufficient collaboration, lack of understanding of the value of architecture, and poor communication between team architects [16]. Bachmann et al. [17] and Nord et al. [18] present four tactics to achieve agility at scale by aligning the system architecture, organization structures and product infrastructures. These include vertical and horizontal system decomposition, matrix and augmented team structures, architecture and infrastructure runway, and deployability tactics and can be used in different phases in a system's life cycles. Uludağ et al. [10] describes how the adoption of domain-driven design supported a large-scale agile development program with three agile teams at a large insurance company. Uludağ et al. [10] report that agile teams and project managers involved in the program conceived that without any form of architectural guidance, large-scale agile development programs can hardly be successful. Dingsøyr et al. [19] investigated a large-scale development program with an extensive use

of Scrum and a focus on customer involvement, inter-team coordination, and software architecture. Two key findings related to software architecture are the tension between up-front and emergent architecture and the demanding role of architects in large-scale agile development.

3 Case Study Design

A case study is a suitable research methodology for this paper, since it helps to study contemporary phenomena in a real life context [20]. We followed the guidelines described by Runeson and Höst [20].

Case Study Design: The main objective of this paper is to investigate the collaboration between architects and agile teams in large-scale agile development in terms of architecture sharing. Based on this objective, we defined one research question (see Sect. 1). The study is a an exploratory single case study, since this paper looks into an unexplored phenomenon and aims to seek new insights and generate ideas for future research [20]. The case was purposefully selected, because the studied company has successfully adopted SAFe for building complex software for the last one and a half years. The unit of analysis is the consumer electronics retailer's large-scale agile development program.

Data Collection: We used a mixed methods approach with three levels of data collection techniques [21]. As direct methods, we observed two Program Increment (PI) Planning events [7] with low degree of interaction by the researcher and low awareness of being observed [20]. These observations provided a deep understanding of the overall structure. With the help of seven semi-structured interviews, roles and practices related to architecture were identified and documented. Quantitative data was collected by the online-survey tool Questback for building the social networks and revealing the collaboration between architects and agile teams (see Sect. 4). Therein, we asked respondents how often they exchange architectural advice and decisions with their colleagues, how often they see their colleagues, and if they have suggestions on how to improve the exchange among team members (using a Likert scale). A total of 32 out of 62 available people from eight teams took part in the survey. Three persons were removed from the analysis because no clear assignment to these persons could be made. The response rate for the remaining 29 program members from eight teams is 47% with 758 connections for architecture sharing.

Data Analysis: Interviews and observation protocols were coded using a deductive approach as proposed by Cruzes and Dybå [22]. Qualitative data collected in interviews form the theoretical foundation for interpreting social relations between architects and agile teams. After initial coding, codes were refined and consolidated by merging related ones and removing duplicates. Quantitative data was analyzed through the use of social network analysis, which comprises a set of methodological techniques that aim to describe and explore patterns in relationships that individuals and groups form with each other [23].

4 Results

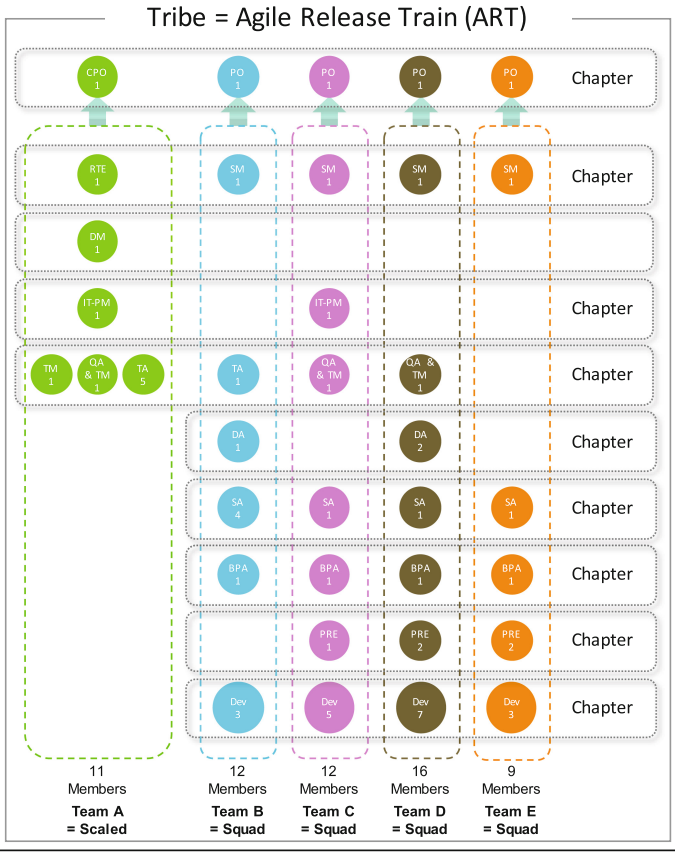
4.1 Case Description

In 2016, the case organization decided to relaunch a failed CRM project using agile methods. Due to the complexity of the project, the management decided to relaunch it with the help of a scaling framework. During early stages of research, the reasons for using Essential SAFe (from now on SAFe) became more apparent and convincing to the management. One reason for choosing SAFe was that it has proven itself in large organizations and offers comprehensive documentation. The adoption was initiated with a pilot project, which was geographically distributed. At the beginning, the pilot project faced a lot of problems. Thus, all involved employees were trained upon agile methods and SAFe by external agile coaches. After a few PIs, the responsible management team perceived that SAFe did not provide sufficient guidance on the coordination of their agile teams. Thus, the organization decided to combine SAFe with the Spotify Model. Within the transformation process, program members were divided into tribes, chapters, squads, and guilds. Figure 2 shows the current organizational structure of the observed program. Figure 2 also shows all 62 members forming a tribe. This tribe consists of a "scaled" team (Team A), which does not play a hierarchical superior, but a more coordinating role without personnel management, and four squads (Team B, Team C, Team D, and Team E). Team F, Team G, and Team H, which are not shown in Fig. 2 constitute representatives of three suppliers that provide external support for their third-party systems. The tribe is divided horizontally into nine chapters for: (1) the chief product owner (CPO) and POs, (2) RTE and SMs, (3) IT project managers (IT-PMs), (4) quality analysts and test managers (QAs & TMs), (5) data analysts (DAs), (6) solution architects (SAs)[1], (7) business process architects (BPAs), (8) product reliability engineers (PRE), and (9) developers (Devs). Each SA is assigned to a squad and takes care of the overall system architecture with its subsystems and interfaces. The team concentrates on the cross-system data flows and processes related to the integration of the architecture. These data flows and processes are used to define minimum interface requirements that all teams must meet. In contrast to SAs, who represent technical architects, BPAs are functional architects that are also dedicated to squads. The responsibilities of BPAs are not really known yet, as their role has been added to the program just recently. However, both architect roles should play a dual role within their squads by making architectural decisions and guiding them to fulfill the required architectural standards. Due to ongoing transformation, guilds have not yet been established but will be organized soon. In the following two sections, the inter- and intra-team exchange of architecture-related information of the observed program will be presented.

[1] The role of the SA in the case organization correspondents to the role of the system architect as described by SAFe [7]. For reasons of consistency, we use the same terminology as the case organization.

4.2 Inter-Team Architecture Sharing

Figure 3 provides an overview of how architecture-related information is shared across all teams. An interesting finding here is that the scaled team is located in the center of the graph. This indicates continuous communication and coordination between the scaled team and the four squads on architectural topics. Figure 3 also shows a close collaboration between Team B and Team E and between Team B and Team D, which is due to architectural dependencies between the systems on which they work. Figure 3 also provides an overview of roles that are inten-



Abbreviations	
BPA - Business Process Architect	PRE - Product Reliability Engineer
CPO - Chief Product Owner	QA & TM - Quality Analyst & Test Manager
DA - Data Analyst	RTE - Release Train Engineer
Dev - Developer	SA - Solution Architect
DM - Delivery Manager	SM - Scrum Master
IT-PM - IT Project Manager	TA - Test Analyst
PO - Product Owner	TM - Test Manager

Fig. 2. Organizational structure of the observed large-scale agile development program

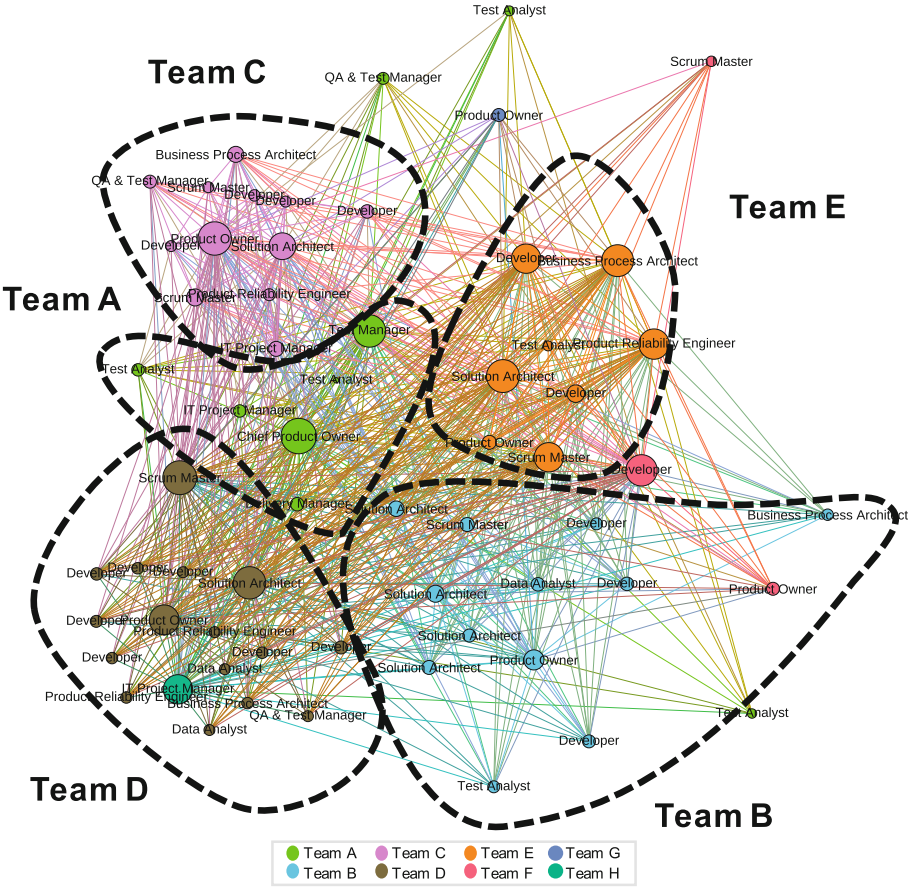


Fig. 3. Social network of eight teams including salient roles that are intensively involved in inter- and intra-team architecture-sharing

sively involved (large nodes) in architecture sharing. First, it shows that the CPO of Team A (CPO_A) is the most outstanding node in the inter- and intra-team exchange of architecture-related information. Second, SAs also form relatively large nodes compared to other roles. This observation confirms the importance of SAs for the exchange of inter- and intra-team architectural information. Figure 3 also shows that the TM_A also plays an important role in architecture sharing. Table 1 presents top 10 stakeholders involved in inter-team sharing based on the normalized degree centrality[2] measure. Table 1 shows that the CPO_A has a normalized degree centrality value of 1,0, which indicates that he/she is sharing information with all stakeholders involved in the observed program. The SA_E

[2] The normalized degree centrality is defined as the number of links of an stakeholder divided by the maximal possible number.

Table 1. Top 10 stakeholders involved in inter-team architecture sharing based on normalized degree centrality

Rank	1	2	3	4	5	6	7	8	9	10
Role	CPO_A	SM_D	PO_C	SA_E	SA_D	TM_A	BPA_E	Dev_F	PRE_E	PO_D
Value	1,0	0,95	0,93	0,92	0,90	0,90	0,89	0,87	0,84	0,82

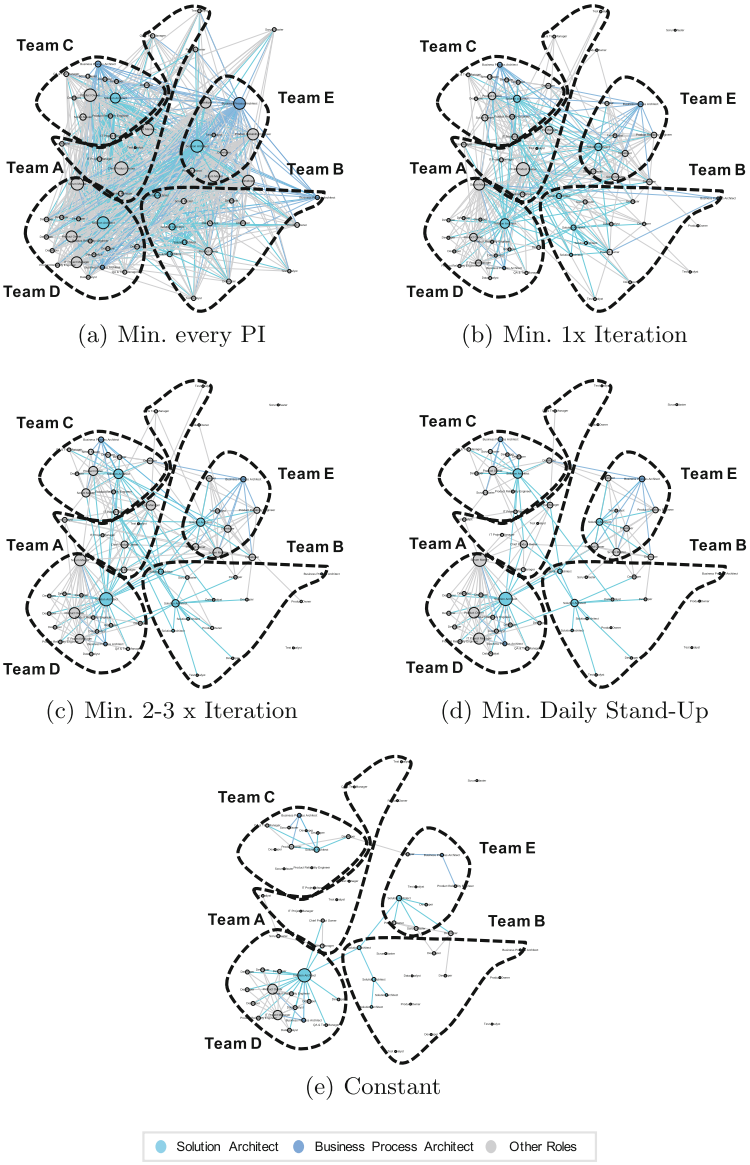
and SA_D have normalized degree centrality values of 0,92 and 0,90 indicating high involvement in inter-team sharing.

The PI planning event of SAFe is a face-to-face event [7] that aims to align all agile teams within the ART to share the common mission and vision by creating iteration plans and team objectives for the upcoming PI. It is conducted every two and a half months and offers a platform for the exchange of general and architectural information across teams, since all members of the ART are present in one location. Figure 4(a) shows that SAs and BPAs have a very strong sharing with other teams during the PI planning. Figure 4(d) reveals a chain communication between the SA_B, SA_C, SA_D, and SA_E on a daily basis. In particular, the chain is composed as follows: SA_E exchanges information with SA_B, who exchanges information with SA_D, who shares information with SA_C. This communication pattern characterizes a centralized communication between SAs. The chain communication pattern can also be observed with SA_B, SA_D, and SA_E. Figure 4(e) shows that SA_B, SA_D, and SA_E constantly[3] exchange information and that the SA_C is no longer involved in an exchange with other SAs. Figure 4 shows that SAs form a decentralized all-channel communication pattern. This means that each SA speaks with all other SAs. The overall comparison also shows that the three external SA of Team B are less participating in the inter-team exchange than the rest of internal SAs involved in the program. Other roles such as SM, TM, PO, and CPO are also heavily involved in exchange of information within the PI planning. The shorter the observed time intervals become, the more dominant the SA becomes with regards to the inter-team sharing.

4.3 Intra-Team Architecture Sharing

The exchange of architectural information in Team B shows a central wheel communication pattern between SAs, since external SAs are guided by the internal SA, who represents the intra-team lead architect (see Fig. 5(a)). Figure 5(a) also shows that SAs form the core of the team. Moreover, Fig. 5(a) shows that BPA_B only exchanges information with another role. A decentralized all-channel communication pattern can be observed in Team C (see Fig. 5(b)). This means that other non-architectural roles exchange information without necessarily involving SA_C. Nevertheless, SA_C plays the most central role, since the SA frequently communicates with all team members. Compared to BPA_B, BPA_C plays a more central role, as he/she shows a close collaboration and communication with his/her

[3] Constant exchange means that it takes place more than once a day.



(a) Min. every PI (b) Min. 1x Iteration

(c) Min. 2-3 x Iteration (d) Min. Daily Stand-Up

(e) Constant

Fig. 4. Social networks focusing on SAs and BPAs with regards to the frequency of inter- and intra-team architecture sharing

squad (see Fig. 5(a) and (b)). The comparison of the two figures also shows that SA_C and BPA_C exchange information more frequently than SA_B and BPA_B. Figure 5(b) shows a decentralized all-channel communication pattern between architects and other team members of Team D. Similar to BPA_C, BPA_D often

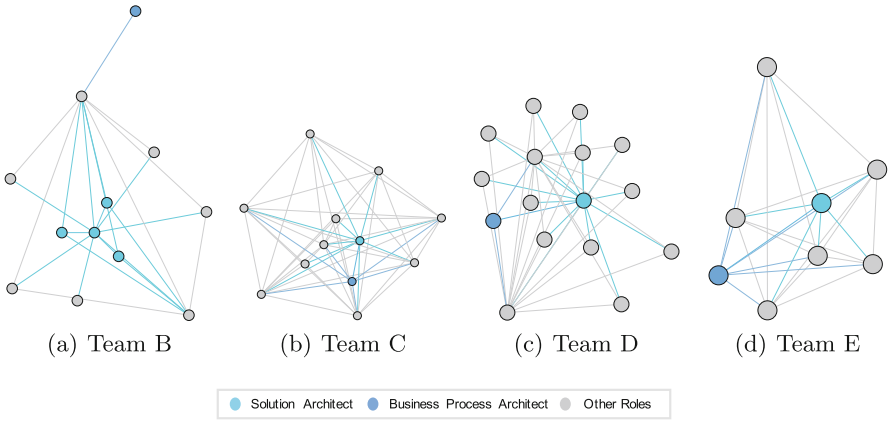


Fig. 5. Social network of the four squads focusing on SAs and BPAs involved in intra-team architecture-sharing

Table 2. Normalized degree centralities of architects in intra-team architecture sharing

	Team B	Team C	Team D	Team E
SA	0,91	1,0	1,0	1,0
BPA	0,09	0,64	0,2	1,0

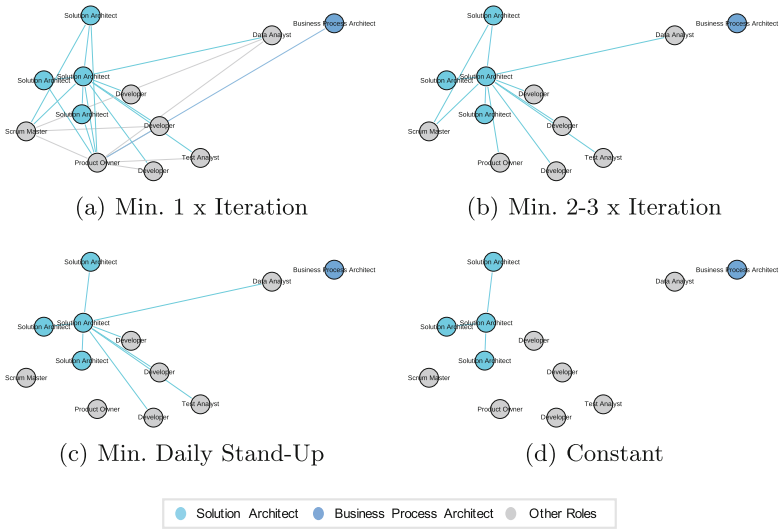
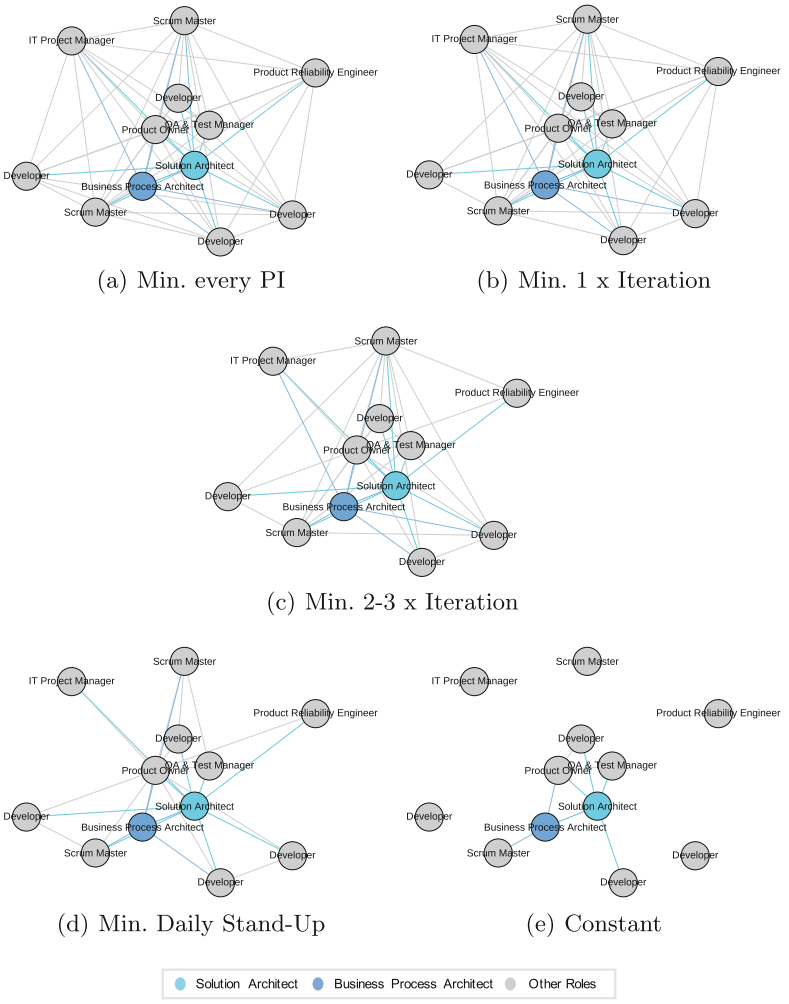


Fig. 6. Social network of Team B focusing on SAs and BPAs with regards to the frequency of intra-team architecture sharing



(a) Min. every PI (b) Min. 1 x Iteration

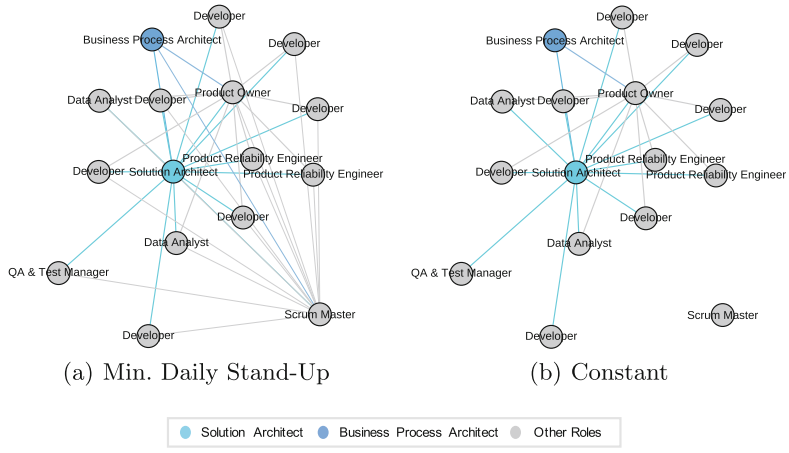
(c) Min. 2-3 x Iteration

(d) Min. Daily Stand-Up (e) Constant

Fig. 7. Social network of Team C focusing on SAs and BPAs with regards to the frequency of intra-team architecture sharing

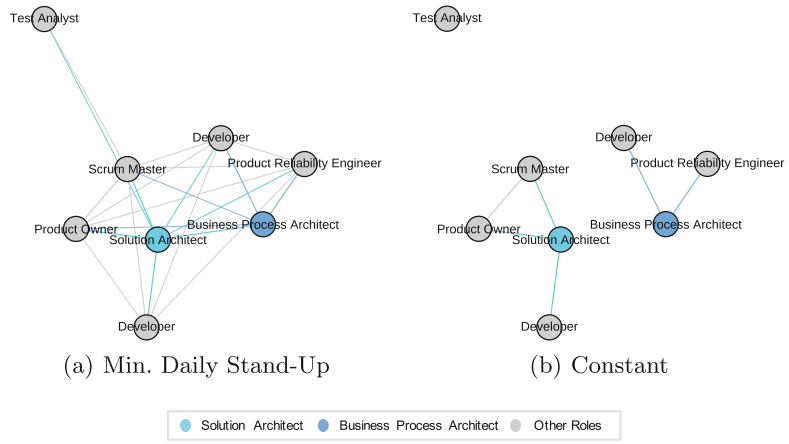
exchanges architecture information with team members. Table 2 shows the normalized degree centrality values of SAs and BPAs involved in intra-team architecture sharing. 75% of the SAs possess a normalized degree centrality value of 1,0 indicating that they share information with all squad members. Comparing SAs with BPAs, Table 2 shows that SAs have a stronger exchange of information with their squad members than BPAs (except Team E).

Figure 6 shows how Team B's intra-team sharing changes at four distinct time intervals. For instance, Fig. 6(a) shows that BPA_B only exchanges information with one Dev_B once per iteration. Figure 6(b) shows that the exchange of information between SAs and non-architectural roles mostly takes place two to three



(a) Min. Daily Stand-Up (b) Constant

Fig. 8. Social network of Team D focusing on SAs and BPAs with regards to the frequency of intra-team architecture sharing



(a) Min. Daily Stand-Up (b) Constant

Fig. 9. Social network of Team E focusing on SAs and BPAs with regards to the frequency of intra-team architecture sharing

times per iteration, while the sharing between SAs takes place constantly (see Fig. 6(d)). Similar to Figs. 6, 7 shows Team C's intra-team architecture sharing. The exchange in the team usually takes place two to three times per iteration (see Fig. 7(c)). Sharing between architects and non-architectural roles takes place on a daily basis (see Fig. 7(d)). In contrast to Team B, Fig. 7(e) shows that SA_C and BPA_C constantly communicate together. Figure 8(a) shows that the exchange between architects and non-architectural roles as well as among architects mainly takes place on a daily basis. SA_D and BPA_D constantly exchange architectural information (see Fig. 8(b)). Figure 8(b) also shows that other mem-

bers such as DA_D, QA & TM_D, PRE_D, and PO_D constantly exchange architectural information. The intra-team exchange of Team E takes place mainly on a daily basis (see Fig. 9(a)). SA_E and BPA_E communicate on a daily basis (see Fig. 9(b)). Architecture sharing between architects and non-architectural roles takes place on a daily basis. Figure 9(b) shows that two groups are formed during the constant exchange of information. The first group includes SA_E, SM_E, Dev_E, and PO_E, while the second group constitutes Dev_E, BPA_E, and PRE_E. Table 3 provides a summary of the social network analysis with identified communication patterns and frequencies.

5 Discussion

5.1 Key Findings

Both architectural roles, i.e. SAs and BPAs, and other roles, e.g. TMs, SMs, and POs, are involved in inter- and intra-team architecture sharing. In particular, the CPO plays one of the most salient roles. An all-channel communication network can be observed in each squad. SAs enable a decentralized exchange so that other team members can exchange architecture-relevant information without necessarily involving SAs. This observation coincides with the values and principles of agile software development. Both SAs and BPAs prefer face-to-face communication with their team members and do not exchange information by including bridging roles. Each squad is accompanied by at least one SA and BPA. Both architects play a dual role in their squads. On the one hand, they make architectural decisions and iteratively create architecture models. On the other hand, they provide guidance and support their squad in meeting architectural standards. With this setup, the observed program aims to increase development speed by balancing emergent and intentional architecture. In all social networks, SAs form central nodes in inter- and intra-team sharing.

Table 3. Summary of the social network analysis

	Team A	Team B	Team C	Team D	Team E
Overview (communication network)	decentralized; -	decentralized; all-channel	decentralized; all-channel	decentralized; all-channel	decentralized; all-channel
Architects with other roles (communication network)	not applicable	decentralized; all-channel	decentralized; all-channel	decentralized; all-channel	decentralized; all-channel
Architects with other roles (frequency of sharing)	not applicable	mostly 2-3x pro iteration (except BPA)	mostly daily basis	mostly daily basis	mostly daily basis
Between architects (Communication Network)	not applicable	centralized; wheel (between SAs)	not applicable	not applicable	not applicable
Between architects (frequency of sharing)	not applicable	constant (except BPA)	constant	constant	mostly daily basis

5.2 Threats to Validity

We discuss potential threats to validity along with an assessment scheme as recommended by Runeson and Höst [20].

Construct Validity: This aspect reflects to what extent operational measures that are studied really represent what the researcher has in mind [20]. Two countermeasures were taken for construct validity. First, interview protocols were coded by the author of this paper and reviewed by a second researcher. Second, a key informant of the organization has reviewed the analyses of this paper.

Internal Validity is irrelevant, as this study was neither explanatory nor causal.

External Validity: This aspect of validity concerns to what extent the findings can be generalized, and to what extent the findings are of interest to other persons outside the case under investigation [20]. This paper focuses on analytical generalization [20] by providing a detailed description of the case. It provides empirical insights that allow for a profound understanding on the collaboration between architects and agile teams. The shown findings should be viewed as valuable insights for other organizations that adopted Essential SAFe.

Reliability: This validity is concerned with to what extent the data and the analysis are dependent on the specific researcher [20]. To mitigate this threat, two countermeasures were taken. First, the case study has been designed so that the large number of interviewees and multiple interviewers allowed data and observer triangulation. Second, a case study database was created, which includes case study documents such as audio recordings protocols, and field notes of observations.

6 Conclusion and Future Work

In this paper, we described the collaboration between architects and agile teams in a large-scale agile development program of a German consumer electronics retailer. Due to the complexity and extent of the CRM product, each squad is guided and supported by at least one SA and BPA. Each SA is responsible for the architecture of a subsystem and ensures that the respective squad complies with defined architectural requirements. The observed program also introduced the new role of the BPA that is responsible for developing the functional architecture of the subsystem. To understand the role of SAs and BPAs and their collaboration with squads, we investigated social networks of one scaled team and four squads. We learned that intra-team architecture sharing is usually facilitated by SAs. Comparing the social networks with common communication networks, we discovered that SAs and BPAs prefer direct communication. For the most part, architects share information on a daily basis with their teams. The intra-team sharing between architects and their teams is characterized by an all-channel communication network.

As future work, we will continue to study the large-scale agile development program of the German consumer electronics retailer. First, we will research how

the current state of architecture sharing is perceived by the stakeholders and how it could be improved by the use of various coordination mechanisms such as ad hoc meetings, co-location or communities of practices. Second, as the squads in the large-scale agile development program become more mature and evolve towards feature teams, we will investigate the architectural decision-making process of squads. We hope to gain a better understanding of the collaboration between architects and squads regarding the distribution of their responsibilities for architectural issues.

References

1. Kettunen, P.: Extending software project agility with new product development enterprise agility. Softw. Process: Improv. Pract. **12**(6), 541–548 (2007)
2. Dingsøyr, T., Moe, N.B.: Towards principles of large-scale agile development. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (eds.) XP 2014. LNBIP, vol. 199, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_1
3. Alqudah, M., Razali, R.: A review of scaling agile methods in large software development. Int. J. Adv. Sci. Eng. Inf. Technol. **6**(6), 28–35 (2016)
4. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016)
5. Uludag, Ö., Kleehaus, M., Caprano, C., Matthes, F.: Identifying and structuring challenges in large-scale agile development based on a structured literature review. In: IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC) 2018, pp. 191–197. IEEE (2018)
6. Rost, D., Weitzel, B., Naab, M., Lenhart, T., Schmitt, H.: Distilling best practices for agile development from architecture methodology. In: Weyns, D., Mirandola, R., Crnkovic, I. (eds.) ECSA 2015. LNCS, vol. 9278, pp. 259–267. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23727-5_21
7. Leffingwell, D.: SAFe® 4.5 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering. Addison-Wesley Professional, Boston (2018)
8. Mocker, M.: What is complex about 273 applications? untangling application architecture complexity in a case of european investment banking. In: 2009 42nd Hawaii International Conference on System Sciences 2009, HICSS, pp. 1–14. IEEE (2009)
9. Waterman, M.: Reconciling agility and architecture: a theory of agile architecture, Ph.D. thesis, Victoria University of Wellington (2014)
10. Uludağ, Ö., Hauder, M., Kleehaus, M., Schimpfle, C., Matthes, F.: Supporting large-scale agile development with domain-driven design. In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNBIP, vol. 314, pp. 232–247. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91602-6_16
11. Uludağ, Ö., Kleehaus, M., Xu, X., Matthes, F.: Investigating the role of architects in scaling agile frameworks. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC), pp. 123–132. IEEE (2017)
12. Kniberg, H., Ivarsson, A.: Scaling agile @ spotify (2012)
13. Guo, L.C., Sanchez, Y.: Workplace communication. Organizational behavior in health care, pp. 77–110 (2005)

14. Presbitero, A., Roxas, B., Chadee, D.: Effects of intra- and inter-team dynamics on organisational learning: role of knowledge-sharing capability. Knowl. Manag. Res. Pract. **15**(1), 146–154 (2017)
15. Lunenburg, F.: Network patterns and analysis: underused sources to improve communication effectiveness. Nat. Forum Educ. Adm. Super. J. **28**(4), 1–7 (2011)
16. Angelov, S., Meesters, M., Galster, M.: Architects in scrum: what challenges do they face? In: Tekinerdogan, B., Zdun, U., Babar, A. (eds.) ECSA 2016. LNCS, vol. 9839, pp. 229–237. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48992-6_17
17. Bachmann, F., Nord, R.L., Ozakaya, I.: Architectural tactics to support rapid and agile stability. Technical report, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute (2012)
18. Nord, R.L., Ozkaya, I., Kruchten, P.: Agile in distress: architecture to the rescue. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (eds.) XP 2014. LNBIP, vol. 199, pp. 43–57. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_5
19. Dingsøyr, T., Moe, N.B., Fægri, T.E., Seim, E.A.: Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empirical Softw. Eng. **23**(1), 490–520 (2018)
20. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical softw. Eng. **14**(2), 131 (2009)
21. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: data collection techniques for software field studies. Empirical softw. Eng. **10**(3), 311–341 (2005)
22. Cruzes, D.S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 275–284. IEEE (2011)
23. Scott, J.: Social Network Analysis. Sage, Thousand Oaks (2017)