Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems

Citation for the original published paper (version of record):

Figalist, I., Elsner, C., Bosch, J. et al (2019). Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems. Lecture Notes in Business Information Processing, 355: 189-206. http://dx.doi.org/10.1007/978-3-030-19034-7_12

Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems

Iris Figalist[1(✉)], Christoph Elsner[1], Jan Bosch[2], and Helena Holmström Olsson[3]

[1] Corporate Technology, Siemens AG, 81739 Munich, Germany
{iris.figalist,christoph.elsner}@siemens.com
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Hörselgången 11, 412 96 Göteborg, Sweden
jan.bosch@chalmers.se
[3] Department of Computer Science and Media Technology, Malmö University, Nordenskiöldsgatan, 211 19 Malmö, Sweden
helena.holmstrom.olsson@mau.se

Abstract. The shift from sequential to agile software development originates from relatively small and co-located teams but soon gained prominence in larger organizations. How to apply and scale agile practices to fit the needs of larger projects has been studied to quite an extent in previous research. However, scaling agile beyond organizational boundaries, for instance in a software ecosystem context, raises additional challenges that existing studies and approaches do not yet investigate or address in great detail. For that reason, we conducted a case study in two software ecosystems that comprise several agile actors from different organizations and, thereby, scale development across organizational boundaries, in order to elaborate and understand their coordination challenges. Our results indicate that most of the identified challenges are caused by long communication paths and a lack of established processes to facilitate these paths. As a result, the participants in our study, among others, experience insufficient responsivity, insufficient communication of prioritizations and deliverables, and alterations or loss of information. As a consequence, agile practices need to be extended to fit the identified needs.

Keywords: Large-scale agile software development · Inter-team coordination · Software ecosystems

1 Introduction

Agile practices in software development have been around for quite some time and originally emerged due to the need to adapt faster to changing customer

P. Kruchten et al. (Eds.): XP 2019, LNBIP 355, pp. 189–206, 2019.
https://doi.org/10.1007/978-3-030-19034-7_12

requirements [1]. Developing in iterations provides the required flexibility and enables agile projects to "identify and respond to changes more quickly than a project using a traditional approach" [1]. With the focus being on "individuals and interactions over processes and tools" [2], preferably through face-to-face communication, agile practices initially aimed at small, local teams. However, as agile development became more popular, larger organizations adapted certain practices as well [3]. As a result, large-scale agile frameworks, such as SAFe [4] and LeSS [5], were developed to provide guidance to large organizations. The application of scaled agile methods has already been investigated to a great extent, e.g. in [3,6–10], and [11]. The focus, however, lies mostly on large-scale projects or multiteams within one organization. This lead us to the question: How to scale agile even further, beyond organizational boundaries?

Software ecosystems can be defined as "the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. [...]" [12]. Opening up a platform to external developers enables platform operators to expand their offerings and provide further functionalities that they would not have been able to develop themselves, thereby providing more value to the customer but at the same time requiring additional coordination efforts [13,14].

In large-scale distributed, agile teams, inter-team coordination has previously been identified as a major challenge [8]. As large-scale software ecosystems differ from traditional organizations in various ways, these challenges cannot directly be adopted. For one, each actor within the ecosystem has its own way of working that can hardly be standardized [13]. This results in many different practices applied across the ecosystem and, therefore, many different degrees of agility which can be difficult to align. Moreover, the actors do not belong to a single but instead to many different organizations which share different, possibly competing, relationships. For this reason, the actors do often neither share the same goals nor communicate in a fully open way. This makes the inter-team coordination even more difficult.

To our knowledge, the challenges of inter-team coordination beyond organizational boundaries are not yet sufficiently investigated in existing literature. For that reason, we raise the following research questions:

(a) How do agile teams within software ecosystems coordinate their efforts?
(b) Which inter-team coordination challenges do agile teams within software ecosystems face?

In order to achieve this, we conducted a case study within two large, industrial software ecosystems to investigate their processes and inter-team coordination. We elaborate the results along three dimensions that constitute the framing for our findings: (a) maturity of the ecosystem (b) phases within the agile lifecycle (c) openness/closedness of the ecosystem. We use this framing to map the conflicting interests between actors as well as the resulting challenges and implications to the dimensions. Therefore, the contribution of this paper is to unfold why certain conflicts arise in a particular situation or setting in order to increase awareness of other actors' mindsets, and to help practitioners understand certain challenges

and possible trade-offs they, thereby, might be facing. Moreover, we were able to tie most of the challenges back to long communication paths and a lack of established processes, raising the need to extend existing agile practices.

The remainder of this paper is organized as follows: First, we explain the characteristics of software ecosystems in Sect. 2, followed by our case study design in Sect. 3. In Sect. 4 we present the results of our study, before providing an overview of related work in Sect. 5, and summing up and concluding our work in Sect. 6.

2 Characteristics of Software Ecosystems

One of the major differences between distributed teams in traditional organizations and software ecosystems is the fact that the teams or actors are not within the same company or organization but instead spread across several organizations, whereby each actor contributes different elements to the system or product. This entails various types of relationships between actors within an ecosystem. For instance, the actors can be competitors or share mutual benefits [12]. The complexity of relationships and dependencies increases with the number of parties involved in the ecosystem [13]. Moreover, the number of actors and their possibly competing relationships results in a lack of sharing data which has previously been observed in large organizations [15].

Even though iterative requirements engineering processes provide an increased flexibility and are already widely applied in software ecosystems, further challenges arise due to the ecosystem's various actors, the physical distance between them, and the complexity of dependencies within the ecosystem, which impede a common understanding among and alignment between actors [16–18]. For one, the interpretation and prioritization of requirements can differ highly between ecosystem actors because different stakeholders value different attributes when dealing with a requirement. Every partner contributes requirements to the ecosystem which might result in a requirement overload, causing complications in the prioritization process [19]. Additionally, the negotiation process of requirements is highly influenced by the amount of power and dependencies between actors [18]. An adequate understanding of the other actors' goals and business models is required in the requirements engineering process in order to satisfy existing stakeholders and attract new partners.

In this paper we analyze how the described characteristics and challenges manifest in the inter-team coordination of agile teams within software ecosystems.

3 Case Study Design

Case studies are a well-known research methodology to investigate and understand contemporary phenomena in their real-world context with no or little control by the researcher [20,21]. As our research questions aim at answering exploratory questions, we believe that this is the right methodology for our study, following the guidelines by Runeson and Höst [20].

Case Study Design. The research objective of our study was to investigate the inter-team coordination and its accompanying challenges across organization boundaries. Specifically, our study focuses on how teams in distributed organizations communicate with each other, what kind of information, feedback or data they share, how it is shared, and how they are making and communicating decisions that affect any of the other teams. To achieve this, we performed this case study in two large software ecosystems, *Ecosystem A* and *Ecosystem B*, which are established in the industrial and the healthcare domain, respectively. Each ecosystem originated in a large, industrial company and offers several services, mostly in terms of applications, to the companies' customers. In order to expand their offerings, they opened up their platform to internal as well as external partners developing applications on the platforms. Figure 1 gives an overview of the ecosystems' structures and actors. We chose the respective ecosystems for our study since the keystone as well as the partners work in agile teams and experience difficulties in their coordination.

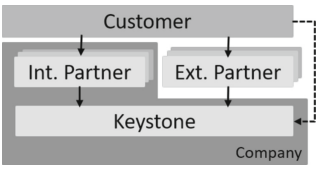

Fig. 1. Actors in ecosystems

Table 1. Description of ecosystems

Ecosystem	A	B		
# of platform devs	500–750	50–100		
# of internal partners	20–50	5–10		
# of external partners	100–200	5–10		
# of apps	20–50	10–20		
Interviewees keystone	DM	PO	PM	
Interviewees partner	SA		PO I	PO II	PO III

Data Collection and Analysis. We conducted semi-structured interviews with key stakeholders within the two ecosystems. Four product owners (PO), one product manager (PM), one demand manager (DM), and one software architect (SA) participated in the study. The demand manager is responsible for collecting and structuring requests from partners and customers before forwarding them to appropriate platform teams. Each of our interviewees belongs to a different, individual agile team within the respective ecosystem and was chosen as a key stakeholder to represents the views of their entire team. Moreover, all interviewees belonged to either the keystone who develops the platform (PM, DM, and one PO) or to a complementing player (three POs, SA) of an ecosystem, therefore providing views from both angles. One product owner, the demand manager, and the software architect belonged to *Ecosystem A* and the product manager and three of the product owners belonged to *Ecosystem B* (see Table 1 for a structured overview of the ecosystems and our participants).

All interviews included the following topics: communication structures, exchange of data and feedback with partners and customers, and decision making processes; though the interview guides were slightly adjusted to the specific roles. At the beginning of each interview the participants were given a brief

introduction into the study and the structure of the respective ecosystem was shortly discussed in order to create a common understanding. Following this, the interviewees were asked for their permission to audio tape the interview, before the interviews were conducted. Each interview lasted between 45 min and one hour, and was transcribed and summarized afterwards.

Additionally to the interviews we derived further knowledge from two experts in this field. Both of them have been working in and with multiple ecosystems, including *Ecosystem A* and *Ecosystem B*, for several years, and shared their experiences in a couple of unstructured interview sessions. They provided additional insights on how the respective ecosystems are structured from a bird's-eye perspective in contrast to the team perspectives. Moreover, we discussed the findings of our interviews with them in order to support the validity of our study.

As a result, we achieve triangulation by **(a)** investigating multiple software ecosystems **(b)** interviewing multiple roles within the ecosystems **(c)** adding expert knowledge.

4 Results

One major difference between distributed teams in large organizations and teams within software ecosystems is that the actors within an ecosystem do not belong to the same company or organization and, therefore, do not necessarily share a common (business) goal. Since we wanted to understand why certain challenges occurred, we decided to first investigate the respective (differing) interests on the keystone's as well as the partners' side in order to locate potential sources of conflicts, before we focused on the analysis of the challenges. Hence, this section is structured as follows: First, we describe the dimensions used in our framework, followed by the conflicting interests, and concluding with the identified challenges.

4.1 Influencing Factors

During the analysis of our data we observed that our findings were highly dependent on three different factors: the phases of an *agile lifecycle* comprise different tasks that require different communication and coordination processes, and the *maturity* as well as the *openness* of an ecosystem influence the relationships between actors and, therefore, also the communication. These factors constitute the main dimensions of the models that include the results of our study, and are explained in detail in the following sections.

Agile Lifecycle. The common agile lifecycle includes an initial requirements definition and planning phase, followed by a development phase including integration and tests, a review and feedback phase, frequent releases, and a reprioritization phase before the next iteration begins [22]. Since all our interviewees work in agile teams, they all go through a similar agile lifecycle, experiencing

different challenges in different phases. As the focus of our study is on coordination challenges, we neglected the technical phases (development and release) but rather focused on the planning, prioritization and feedback phase. We asked each interviewee about the phases they go through and, based on the literature as well as their answers, we propose the agile lifecycle in Fig. 2 that constitutes one dimension in our study.

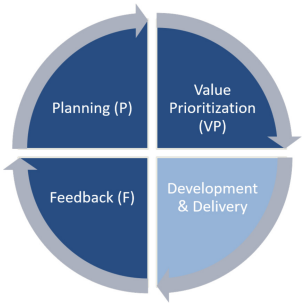

Fig. 2. Agile lifecycle

Maturity. Based on our interviews, we noticed that the maturity of the respective ecosystems plays a rather important role. Especially the communication between keystone and partners changes drastically over different phases of maturity. For instance, in the early phases of opening up a platform it is important to attract new and please existing partners, therefore the focus on communication is much higher than in later phases when the ecosystem is mature enough to attract partners automatically because of its success and the benefits for the partners.

One way to describe the evolution of a technology or an innovation is the s-curve. It describes the performance of a technology during different maturity stages from "pregnancy, birth, childhood, adolescence, maturity, and decline" [23]. Both ecosystems already have products on the market but regarding their maturity we would classify *Ecosystem A* as still being in the "birth" phase and *Ecosystem B* as being in the "childhood" phase. Specifically, *Ecosystem A* is still in the process of opening up their development to external partners while *Ecosystem B* is already established but still accelerating. For this reason, we define the second dimension as the maturity from "opening up" to "acceleration".

Openness vs. Closedness. Hartman et al. identified two different types of ecosystems, open and closed [24]. While closed ecosystems are still tightly coupled to and somewhat controlled by the keystone, open ecosystems are easily accessible for partners and can be characterized by their interchangeability of components and parties. However, ecosystems are not necessarily one or the other, they can also be in a hybrid stage [24]. This is relevant for our study since

the communication and trust across ecosystem partners appears to be different for the two types. For instance, the actors in closed ecosystems are more interconnected than the actors of open ecosystems and, therefore, communicate more openly and share a higher level of trust. By tendency, *Ecosystem A* belongs to the category "open ecosystem" while *Ecosystem B* incorporates closed as well as open aspects. This is directly reflected in the characteristics of relationships that the keystone shares with different types of partners (both external & internal).

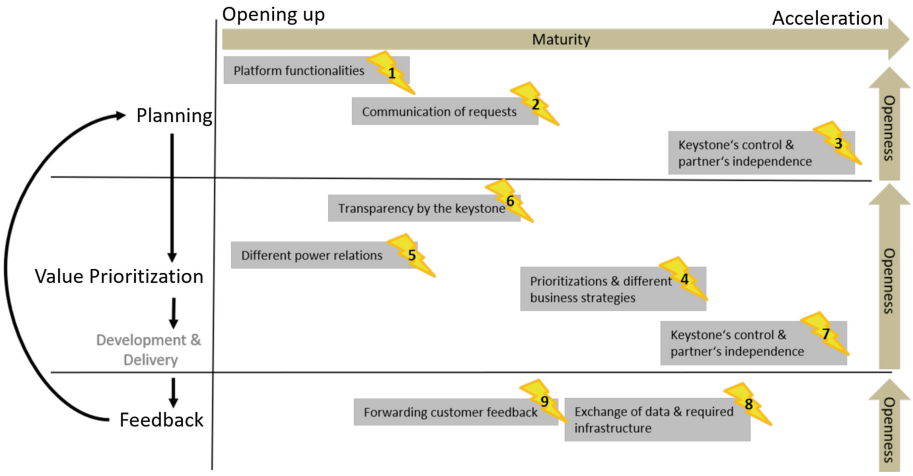

Fig. 3. Conflicting interests between different ecosystem actors

4.2 Conflicting Interests

Each ecosystem actor usually follows its own business strategy which can easily result in different interests that are hard to align and, therefore, cause conflicts between the actors. In order to analyze which opposing interests might lead to conflicts, and based on that even challenges, we extracted the interests of the partners as well as the keystone out of the interviews. Next, we mapped contrary interests that share a common link from both sides to each other which, therefore, constitute main drivers and crucial influencing factors for certain conflicts. Since not all interests and conflicts apply to all ecosystems or to all phases of agile development, we mapped the conflicting interests to different maturity levels, phases within an agile lifecycle, and the degree of openness (see Fig. 3). A more detailed description of the conflicts, interests of the platform and partners, and other influencing factors can be found in Table 2. All of the described results were derived out of the interview sessions of our case study. Overall, we identified nine conflicts that were caused by opposing interests on the partners' and the keystone's side. Three of the conflicts originated in the planning phase, four in the value prioritization phase, and two in the feedback phase.

Table 2. Description of conflicting interests between keystone and partners and the influencing factors in different phases

#	Conflict	Partners' Interests	Keystone's Interests	Ph	Ma	IF
1	Platform functionalities	Request specific partner functionalities	Provides the functionality that brings most value to the ecosystem	P	OP	O
2	Communication of requests	Expect fast & easy communication processes	Asks for well described requests	P	OP& A	O
3	Keystone's control & partners' independence (planning)	Become more independent	Keep control over the partners' customer interaction	P	A	C
4	Prioritizations & different business strategies	Follow own business strategy	Ensure the ecosystem's future	VP	OP& A	-
5	Different power relations	Want to be recognized by keystone	Wants to bind "important" partners to ecosystem	VP	OP	PR
6	Transparency by the keystone	Expect transparency (e.g. delivery timelines, commitments)	Wants to stay flexible/be able to reprioritize	VP	OP	O
7	Keystone's control & partners' independence (prioritization)	Obtain broad picture over all customers	Keep control over the partners' customer interaction	VP	A	C
8	Exchange of data & required infrastructure	Share customer feedback/data with collaborative partners	Low priority to provide infrastructure	F	OP& A	O
9	Forwarding customer feedback	Only benefit from forwarding feedback if it is directly connected to the partner's app/service	Needs the partners to forward requirements (if related to the keystone) of their customers	F	OP& A	O

Phases (Ph): Planning (P), Value Prioritization (VP), Feedback (F)
Maturities (Ma): Opening up (OP), Acceleration (A)
Influencing factors (IF): Openness (O), Closedness (C), Power Relations (PR)

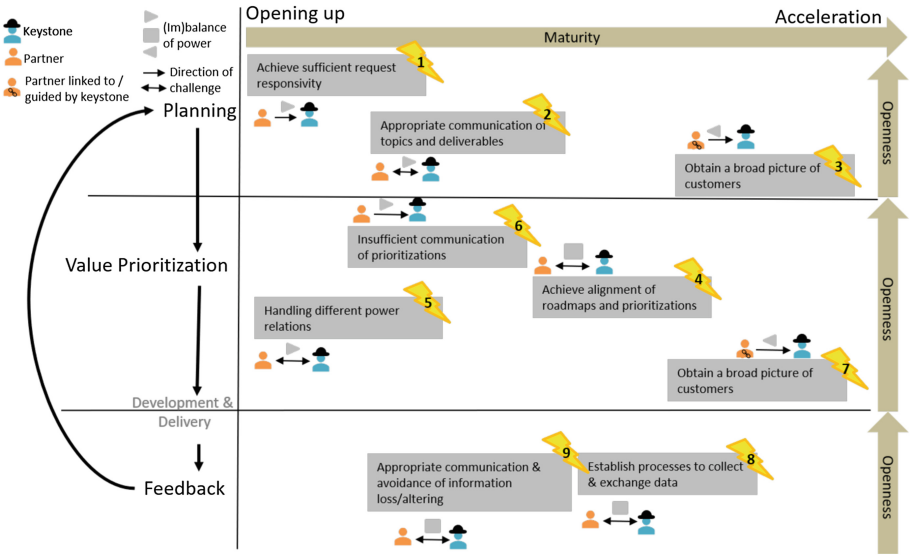

Fig. 4. Mapping of challenges to different settings

Planning Phase. Conflict #1 concerns the ***platform functionalities*** provided by the keystone and required by the partners. On the one side, the partners require the keystone to provide specific functionalities for their minimum viable product (MVP) while the keystone receives so many requests that it is difficult to take all partners into consideration so "[they] need to come to a point where [they] say what platform value creates the most value to the ecosystem and that's not easy because the ecosystem is so broad". For this reason, the keystone concentrates its planning efforts on the partners that bring the most value to the ecosystem.

The next issue (#2) is related to the ***communication of requests*** and the inconsistency of processes. Partners want to communicate their requests in an easy and fast way, and expect the keystone to respond to their requests, while the keystone "receive[s] quite small or tiny, tiny described requests" but "would like to receive more well described requests" from their partners. As there are no consistent processes available this often leads to misunderstandings of what the requirement actually is and, therefore, causes displeasure on both sides.

Moreover, the ***keystone's control and the partners' independence*** (#3) lead to conflicts, especially in closed ecosystems. While the partners want to be independent of the keystone in order to being able to optimize individual business interests, the keystone wants to keep control over the end-customers and the partners' interactions with these customers in order to ensure a coherent, overall business offering.

Table 3. Description of coordination challenges caused by divergent interests of actors within the ecosystem

#	Challenge	Description	Ph	Ma	IF	PR
1	Achieve sufficient request responsivity	P: Keystone not sufficiently reactive K: Keystone receives too many requests over a lot of different channels Why: many different communication channels, processes not well established yet	P	OP	O	P>K
2	Appropriate communication of topics & deliverables	P: Lack of transparency & lack of support -> leads to lack of trust K: Apps formulate high level user stories -> Keystone PM refines it -> potential misunderstandings Why: long communication paths, not all PMs have expertise in all areas -> easy to misunderstand	P	OP&A	O	P>K
3	Obtain a broad picture of customers (planning)	P: Partners get restrictions from keystone concerning customer interaction -> no broad overview on customer's needs K: Keystone wants to ensure an appropriate representation of the entire ecosystem in front of the customer Why: partners & keystone closely coupled, keystone does not want the customer to see the product in a non-ready state	P	A	C	K>P
4	Achieve alignment of roadmaps and prioritizations	P: Every ecosystem partner has own roadmap and prioritizations do often not match K: Challenging to consider all partners & decide what brings most value to ecosystem Why: no common business interests in software ecosystems	VP	OP&A	-	-
5	Handling different power relations	P: Keystone gives preference to certain stakeholders -> neglect "less important" partners Why: Not all partners can be treated the same K: Partners create pressure in order get their requests preferred -> How to decide which partner/customer is more important? Why: Keystone relies on "powerful" partners in opening up phase	VP	OP	PR	P>K
6	Insufficient communication of prioritizations	P: Prioritizations are not well communicated K: Challenge to handle trade-off between pleasing partners and maintaining flexibility Why: Keystone wants to stay flexible	VP	OP	O	P>K
7	Obtain a broad picture of customers (prioritization)	P: Limited communication between customers and partners -> not all customers included in prioritization process K: See challenge #3 Why: as a result of #3	VP	A	C	K>P
8	Establish processes to collect & exchange data	Partner & Keystone: No exchange of customer feedback within ecosystem, no direct Feedback channel / centralized way to store & access feedback -> information loss, "one-sided" feedback Why: off-the-shelf infrastructure can usually not be used, do both partners/keystone benefit by sharing?	F	OP&A	O	-
9	Appropriate communication & avoidance of information loss/altering	P: Insufficient communication of malfunctions by the keystone Why: communication channels for incident reporting must be established, more challenging in ecosystem K: Limited amount of information & information loss when communicating across multiple ecosystem partners Why: Multiple alterations due to long communication paths, feedback from end-customers communicated via partners	F	OP&A	O	-

Partner challenges (P), Keystone Challenges (K)
Phases (Ph): Planning (P), Value Prioritization (VP), Feedback (F)
Maturities (Ma): Opening up (OP), Acceleration (A)
Influencing factors (IF): Openness (O), Closedness (C), Power Relations (PR)
Power relations (PR): Partner > Keystone (P>K), Keystone > Partner (K>P)

Value Prioritization Phase. Conflict #4 concerns ***prioritizations and different business strategies***. As each partner follows its own business strategy, the keystone wants to ensure the ecosystem's future which leads to conflicts in the prioritization process since the different strategies can be difficult to align. One interviewee explains that they need to decide "from a platform point of view, what makes most sense, what is scalable, what is beneficial for a lot of customers [...] that's a challenge".

It is quite natural that some ecosystem partners are more important business partners to the keystone than others. However, this leads to ***different power relations*** (#5) as the important partners can create more pressure on the keystone than the others. Ultimately, the partners expect the keystone to be aware of them and their needs and to be treated (at least) equally to other partners,

while the keystone wants to bind its important partners (e.g. defined by the number of customers, revenue etc.) to the platform.

Moreover, especially during that opening up phase, the ecosystem partners expect transparency of the keystone, e.g. concerning the prioritization of next steps, delivery timelines, or commitments. One of the interviewees states that he "would like to have more transparency how and on what basis decisions are made [...] because at the moment it's very non-transparent how [the keystone] decides what constitutes the biggest value for the overall project". However, the keystone avoids giving too detailed commitments and detailed timelines in order to stay flexible and being able to reprioritize. This leads to conflicting interests concerning the amount of ***transparency by the keystone*** (#6).

Analogously to and building upon conflict #3, the ***keystone's control and the partners independence*** (#7) create a conflict in the prioritization phase of closed software ecosystems. The partners would like to base their prioritization on a broad picture of all their customers while the keystone wants to keep control over the interaction with customers.

Feedback Phase. The partners would like to receive as much information on their customers as possible even if the data is collected by another partner. However, they are disinclined to share their data with competitors within the ecosystem but would be willing to do so with collaborative partners. On the other hand, the keystone perceives it as low priority to provide an infrastructure for sharing data across the ecosystem. This leads to conflicts concerning the ***exchange of data and the required infrastructure*** (#8) to do so.

Lastly, and related to the previous conflict, the handling and ***forwarding of customer feedback*** (#9) concerning other partners or the keystone also constitutes certain challenges since the partners only benefit from forwarding feedback if it is directly connected to the partners' apps or services. However, the keystone relies on the partners to forward platform-related requirements of their customers to them. Additionally, one interviewee explains that "feedback from customer visits are a tricky thing because the POs are going to the visits and we have a process described on how to feed back the feedback to the organization and also to me but that is still a little bit... some use it, some don't and some you have to chase to get their feedback for their customer" which is why they "need to turn that into a more automated, also tool-automated, process flow".

4.3 Challenges

Based on the previously extracted conflicting interests, all ecosystem related challenges faced by either the keystone or the partners were extracted out of the interviews. For each challenge we identified the following properties: The ecosystem's maturity, the phase within the agile lifecycle, other influencing factors, and causes for the respective challenge. Table 3 shows an overview of the detected challenges. In a next step, we mapped the challenges into a multi-dimensional model (see Fig. 4). The two main dimensions are the phases within the agile lifecycle (planning, value prioritization, and feedback) and the degree of maturity

of the ecosystems. We added an extra dimension, the openness of the ecosystem, to each of the phases of the agile lifecycle because we identified challenges within these phases that were also strongly influenced by this factor. We identified two types of partners: partners that are closely coupled to or guided by the keystone and partners that are only loosely coupled to the keystone. Challenges between actors may be effective only in one or in both directions (arrows with one vs. two heads in Fig. 4). The power balance can be even or be dominated by one actor (indicated by a square or by triangles respectively).

Planning Phase. The first challenge results out of conflict #1 concerning the development of basis or new platform functionalities. The partners sometimes feel like the keystone is not sufficiently responsive and takes too long to deliver needed functionalities while the keystone receives too many requests over a lot of different channels which makes it difficult to respond to or handle the requests in a decent amount of time, as one interviewee explains "we keep on getting requests from everywhere [...] not everything can be taken up at the same time". The challenge is to ***achieve the right request responsivity*** (#1). Among others, this is caused by missing or not well established processes to handle such requests which, again, leads to many different communication channels.

Furthermore, both cases in our study perceived an ***appropriate communication of topics and deliverables*** (#2) between platform and partners as quite difficult as a result to conflict #2. The partners reveal that they would appreciate more transparency on the keystone's side in order to get a clear picture of what is possible to achieve. If this is not well communicated, this lack of transparency easily leads to a lack of trust. On the other hand, the keystone explains that they mostly receive high level user stories from their partners which have a high potential for being misinterpreted by the product manager who has to refine the user stories. Possible causes for this challenge are long communication paths across multiple stakeholders often implying information loss, and the fact that it is impossible for product managers to have expertise in all of the partners' areas which easily leads to misunderstandings, especially since the partner offering are "operating in a very specific domain which makes it difficult for most people to understand the topics".

Another challenge, closely coupled to closed ecosystems and related to conflict #3, is to ***obtain a broad picture of the end-customers*** (#3) due to keystone guidelines. In case partners and the keystone are very tightly coupled, partners who talk to customers are perceived as representatives of the entire ecosystem. For this reason, the platform wants to ensure an congruent representation of the ecosystem in front of the customers. In the particular case, the keystone has collaboration agreements with certain customers and the partners get instructions which partners they should talk to. However, this keeps the partners from getting a broad overview over all customers.

Value Prioritization Phase. As a result to conflict #4, it is challenging to ***achieve alignment of roadmaps and prioritizations*** (#4) as the actors

within an ecosystem simply do not share a common business interest. This makes it difficult for the keystone to consider all partners and decide what brings the most value to the ecosystem. One of the partners states that "it is challenging because the keystone has its own roadmap, its own prioritizations and this can cause conflicts if the priorities or values do not match".

Quite the contrary to the previous challenge and related to conflict #5, this challenge addresses the difficulty of ***handling different power relations*** (#5) within the ecosystem. The partners feel like the keystone gives preferences to certain "important" customers and neglects the "less important" customers. One interviewee explains that he feels like "it depends on which partner has the greatest business potential". However, the keystone is simply not able to treat all partners with the same amount of attention because "[the partners] are always creating pressure" in order to get their requests preferred which naturally leads to the questions how to decide which partners are more important.

Conflict #6 concerning the amount of transparency by the keystone leads to an ***insufficient communication of prioritizations*** (#6), e.g. concerning the keystone's next steps, which causes displeasure on the partners' side. At the same time, the keystone faces the challenge how to handle the trade-off between pleasing partners and maintaining its flexibility.

As a result of challenge #3 and related to conflict #7, we observed that – due to the divergent viewpoints concerning the partners independence and the keystone's control – the partners face the challenge of ***obtaining a broad picture over all their customers*** (#7). The reason is that they are unable to include all their customers in their prioritization process due to the keystone's limitations concerning the customer communication. One of the interviewees even states that "It would be much better to run more statistics because I don't feel like this is a comprehensive picture".

Feedback Phase. Some of the interviewees reported on their interest in collecting and sharing customer data with certain collaborative stakeholders (conflict #8), however, so far there exist no established processes for software ecosystems to do so. The interviewees explain that they "would rather have direct feedback channels" or "centralized ways to store and access feedback" because off-the-shelf infrastructure can usually not be used for such kind of data sharing since fine-grained access to the data is not easy to control and legal or privacy issues need to be addressed. As a result, this leads to information loss and one-sided feedback. Therefore, the challenge is to ***establish processes to collect and exchange data*** (#8).

Lastly, both of our cases perceive the communication across stakeholders as insufficient and are under the impression that a lot of information gets altered or lost due to the (mis-)communication across multiple partners. This results in the challenge of an ***appropriate communication and avoidance of information loss and altering*** (#9). One of the interviewees revealed that the keystone does not immediately communicate malfunctioning platform features that the partners' features rely on to them because high priority communica-

tion channels for incident reporting in software ecosystems would need to be established first. On the other hand, the keystone suffers from limited amount of customer feedback and information loss since the chances for alterations are very high due to the long communication paths. One interviewee reports that "the more people you involve in between the more information gets lost". Moreover, the feedback from end-customers concerning platform features are often communicated via the partners. Especially in open ecosystems the keystone often does not have direct customer contact. This makes it challenging for the keystone to receive information on and to understand the real customer's needs.

5 Related Work

Previous research suggests that the application of agile practices is more difficult in large projects or organizations than in small teams [25]. As an organization grows it becomes challenging to keep an overview of all projects and groups within one organization [26]. Additionally, if the activities between them are not well communicated, it is hard to keep track of the existing dependencies. These factors often result in coordination challenges and additional coordination efforts [9,27]. For instance, an overarching figure or role, as well as appropriate methods, are required to coordinate the teams and address team-crossing challenges [11]. However, inter-team coordination in software ecosystems rises additional complications as the teams are distributed over several organizations who rarely share common goals or strategies nor a centralized control figure who coordinates them.

Moreover, it has been observed that an increased autonomy enabled by agile practices causes individual teams within a mutual organization to prioritize their own goals over the larger context [27]. Knowledge regarding the system is spread across the distributed teams and processes to share that knowledge need to be established [28,29]. Additionally, a global distribution of teams leads, among other challenges, to "reduced feelings of proximity when telecommunication is necessary, and difficulty in arranging frequent meetings due to time zone differences" [27].

Previous studies by Dingsøyr et al. [6] and Stettina et al. [30] indicate that most issues identified in agile large-scale software projects are related to processes as well as the people and their relationships. We observed quite similar results in our case study. Nevertheless, our results differ from the challenges of distributed agile teams in the way that the interactions between teams of a single organization are quite different to the interactions across organizations. In the latter case, single parties do not necessarily share a mutual larger context or even a common business interest which also impedes the sharing of knowledge. Moreover, the individual teams do neither apply or utilize unified processes nor can they be forced to do so since a central control figure does not exist in this context.

In order to improve the lack of visibility in large-scale projects, methods and solutions such as agile portfolio management, reporting or inter-team retrospectives have been introduced to connect the business strategy and the respective

teams, to get an overview of all initiatives within a portfolio, and to address inter-team coordination challenges [6,10]. However, it has not been investigated yet how such practices could be applied across organizational boundaries. For instance, some actors can be reluctant to share their reports with certain other actors. Therefore, practices or guidelines would need to be established to coordinate the distribution of reports or to enable inter-organization retrospectives.

The complexity of (inter-)team coordination tends to increase with the size of the project and the number of teams involved (e.g. in multiteam systems). A shared mental model (e.g. concerning the work process, tasks, or awareness of who knows what), closed-loop communication, and trust are considered mechanisms that facilitate the coordination of multiple teams. Bjørnson et al. [7] investigated the practices that can be applied in order to implement these mechanisms. They identified, among others, formal as well as informal communication channels, specialized roles that rotate between teams, stand-up meetings, mini demos, and discussions in an open workspace as helpful tools to implement the mechanisms. In their case study, all teams are located in the same office and many of the proposed practices rely on the co-location of the teams, or at least on a shared common business interest and willingness to exchange information. These characteristics can usually not be observed in software ecosystems which makes it challenging to adapt these practices and mechanisms in this context.

Scheerer et al. [31] investigated different types of coordination strategies for multiteam systems. Each of their strategy types comprises three coordination types – mechanic (e.g. plans, rules), organic (e.g. mutual adjustment, feedback), and cognitive (e.g. cognitive similarity configurations) – that are applied to different kinds of extents, e.g. low, medium, or high. This work specifically focuses on multiteams that work on the same software product and while each team works toward an individual goal, they still share, at least to some extent, a mutual collective goal [31]. Rolling out a unified coordination strategy ecosystem-wide is very difficult to enforce since it would affect teams across several organizations, each of them pursuing their own goals and applying their own practices, without sharing a central control figure.

6 Conclusion

The research objective of our study was to elaborate the arising coordination challenges of agile teams within software ecosystems. Our findings indicate that many of the identified coordination challenges are either directly or indirectly related to long communication paths and a lack of well established communication processes, especially if information needs to be shared with other actors across organization boundaries. In contrast to distributed teams within one company, this is additionally challenging because of the varying, sometimes even competitive, relationships that influence the communication and the way data is forwarded or shared. For one, our participants perceived the responsivity as very slow and insufficient. Moreover, the deficient communication structures cause a lack of awareness and understanding of topics, deliverables and timelines between the keystone and its partners. The keystone is rather cautious when it comes to

revealing its prioritizations and plans for the future which causes frustration on the partners' side. In addition to that, our results imply that on many occasions information gets lost or altered due to the multiple hops it has to pass. Our research provides evidence that there is a need to adapt or develop agile processes to facilitate and enable across-organization communication, coordination, and exchange of data.

Therefore, future work could be dedicated to solving the identified challenges and to investigate how agile practices would need to be adapted in order to fit across-organizational needs.

References

1. Cohen, D., Lindvall, M., Costa, P.: An introduction to agile methods. Adv. Comput. **62**(03), 1–66 (2004)
2. Alliance, A.: Agile manifesto, vol. 6, no. 1 (2001). http://www.agilemanifesto.org
3. Jørgensen, M.: Do agile methods work for large software projects? In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNBIP, vol. 314, pp. 179–190. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91602-6_12
4. Leffingwell, D.: SAFe 4.0 Reference Guide: Scaled Agile Framework for Lean Software And Systems Engineering. Addison-Wesley Professional, Boston (2016)
5. Larman, C., Vodde, B.: Large-scale Scrum: More With Less. Addison-Wesley Professional, Boston (2016)
6. Dingsøyr, T., Mikalsen, M., Solem, A., Vestues, K.: Learning in the large - an exploratory study of retrospectives in large-scale agile development. In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNBIP, vol. 314, pp. 191–198. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91602-6_13
7. Bjørnson, F.O., Wijnmaalen, J., Stettina, C.J., Dingsøyr, T.: Inter-team coordination in large-scale agile development: a case study of three enabling mechanisms. In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNBIP, vol. 314, pp. 216–231. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91602-6_15
8. Begel, A., Nagappan, N., Poile, C., Layman, L.: Coordination in large-scale software teams. In: Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, pp. 1–7. IEEE Computer Society (2009)
9. Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A.: Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans. Softw. Eng. **44**(10), 932–950 (2018)
10. Rautiainen, K., von Schantz, J., Vahaniitty, J.: Supporting scaling agile with portfolio management: case paf. com. In: 2011 44th Hawaii International Conference on System Sciences (HICSS), pp. 1–10. IEEE (2011)
11. 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
12. Manikas, K., Hansen, K.M.: Software ecosystems - a systematic literature review. J. Syst. Softw. **86**(5), 1294–1306 (2013)
13. Bosch, J.: From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference, SPLC 2009, pp. 111–119. Carnegie Mellon University, Pittsburgh (2009)

14. Bosch, J., Bosch-Sijtsema, P.: From integration to composition: on the impact of software product lines, global development and ecosystems. J. Syst. Softw. **83**(1), 67–76 (2010)
15. Fabijan, A., Olsson, H.H., Bosch, J.: The lack of sharing of customer data in large software organizations: challenges and implications. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 39–52. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33515-5_4
16. Fricker, S.: Specification and analysis of requirements negotiation strategy in software ecosystems. In: CEUR Workshop Proceedings, vol. 505, pp. 19–33 (2009)
17. Knauss, E., Damian, D., Knauss, A., Borici, A.: Openness and requirements: opportunities and tradeoffs in software ecosystems. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 213–222 (2014)
18. Valença, G., Alves, C., Heimann, V., Jansen, S., Brinkkemper, S.: Competition and collaboration in requirements engineering: a case study of an emerging software ecosystem. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 384–393 (2014)
19. Karlsson, L., Dahlstedt, Å., Natt och Dag, J., Regnell, B., Persson, A.: Challenges in market-driven requirements engineering-an industrial interview study. In: Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (2002)
20. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**(2), 131 (2009)
21. Yin, R.K.: Case Study Research: Design and Methods, 5th edn. SAGE Publications, Thousand Oaks (2014)
22. Burger, R.: The ultimate guide to agile software development (2016). https://blog.capterra.com/the-ultimate-guide-to-agile-software-development/. Accessed 8 Jan 2019
23. Slocum, M.S.: Technology maturity using s-curve descriptors. TRIZ J. (1999)
24. Hartmann, H., Trew, T., Bosch, J.: The changing industry structure of software development for consumer electronics and its consequences for software architectures. J. Syst. Softw. **85**(1), 178–192 (2012)
25. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9–10), 833–859 (2008)
26. Stettina, C.J., Schoemaker, L.: Reporting in agile portfolio management: routines, metrics and artefacts to maintain an effective oversight. In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNBIP, vol. 314, pp. 199–215. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91602-6_14
27. 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)
28. Rolland, K.H.: Scaling across knowledge boundaries: a case study of a large-scale agile software development project. In: Proceedings of the Scientific Workshop Proceedings of XP2016, p. 5. ACM (2016)
29. Moe, N.B., Olsson, H.H., Dingsøyr, T.: Trends in large-scale agile development: a summary of the 4th workshop at XP2016. In: Proceedings of the Scientific Workshop Proceedings of XP2016, p. 1. ACM (2016)
30. Stettina, C.J., Hörz, J.: Agile portfolio management: an empirical perspective on the practice in use. Int. J. Proj. Manag. **33**(1), 140–152 (2015)
31. Scheerer, A., Hildenbrand, T., Kude, T.: Coordination in large-scale agile software development: a multiteam systems perspective. In: 2014 47th Hawaii International Conference on System Sciences, pp. 4780–4788. IEEE (2014)