Explaining the Emergence of Team Agility: A Complex Adaptive Systems Perspective

Karl Werder (karl.werder@paluno.uni-due.de) - Paluno – the Ruhr Institute for Software Technology, University of Duisburg-Essen

Alexander Maedche (alexander.maedche@kit.edu) - Institute of Information Systems and Marketing, Karlsruhe Institute of Technology

Abstract

Purpose: Agile software development helps software producing organizations to respond to manifold challenges. While prior research focused on agility as a project or process phenomenon, we suggest that agility is an emergent phenomenon on the team level.

Research approach: Using the theory of complex adaptive systems (CASs), we capture the multiple influencing levels of software development teams (SDTs) and their interplay with self-organization and emergence. We investigate three agile SDTs in different contextual environments that participate with four or more different roles each.

Findings: The results suggest self-organization as a central process when understanding team agility. While contextual factors often provide restriction on self-organization, they can help the team to enhance its autonomy.

Research implications: Our theoretical contributions result from the development and test of theory-grounded propositions and the investigation of mature agile development teams.

Practical implications: Our findings help practitioners to improve the cost-effectiveness ratio of their team's operations.

Originality: The study provides empirical evidence for the emergence of team agility in agile SDTs. Using the lens of CAS; the study suggests the importance of the team's autonomy.

Keywords: Information systems development, global software development, case study, agile computing, system dynamics.

1. Introduction

Software development organizations need to respond to manifold challenges that include new customer requirements, market dynamics, mergers, and technological innovation (Börjesson and Mathiassen, 2005). Hence, software development organizations need to improve their reaction to changes, such as changing customer requirements. As a result, many organizations are increasingly adopting agile software development (ASD) as their development methodology (Serrador and Pinto, 2015; West and Grant, 2010). However, simply adopting agile methods, such as scrum or extreme programming, will not automatically lead to an agile organization (Conboy, 2009; Gregory et al., 2016). A corresponding example is the development organization for Visual Studio Online in Microsoft, which took four years to move from a waterfall-oriented organization to becoming a truly agile organization that releases new features into the cloud-based product in a three-week cycle[1]. Thus, there are different states of agility, but as of yet we lack a proper understanding of how they emerge. We shed light on this dilemma by identifying different dynamics that explain different emergent states of agility within software development teams.

While previous research on agility often compared ASD with traditional or waterfall methods (Dybå and Dingsøyr, 2008), more recently there has been a stronger investigation into more mature agile teams (Dingsøyr et al., 2012; Silva et al., 2015). Many teams have already adopted ASD focusing on the method, and institutionalized methods such as scrum help organizations to get a start. However, the adoption of a specific method (e.g., scrum) cannot explain the difference between mature and immature agile teams (Gill et al., 2016). Rather, prior work suggests concepts independent of the method and scholars have called for more research related to human or social factors (Campanelli and Parreiras, 2015; Dingsøyr et al., 2012; Dybå and Dingsøyr, 2008). We suggest that team agility is an emergent phenomenon that develops and that evolves over time (Goldstein, 2000; Kozlowski and Chao, 2012).

When investigating emergence (i.e., the result of the process self-organization), scholars often rely on the theory of complex adaptive systems (CAS) (Alaa and Fitzgerald, 2013; Mittal, 2013). Non-linearity, emergence, and self-organization are major characteristics of CAS. Non-linearity refers to the relationship between the system components and the whole. When the relationship is non-linear, a small change in a component can lead to a larger change in the whole (McCarthy et al., 2006). The concepts of self-organization and emergence are often discussed together. While self-organization is described as a process, emergence is the result of such a process (Curşeu, 2006; McCarthy et al., 2006). Both characteristics define the trajectory of the CAS and therefore require further investigation. We need to understand the conditions leading to different emergent states in order to channel them accordingly (Goldstein, 2000; Kozlowski and Chao, 2012). Hence, we investigate the broader picture that considers local, global, and contextual dynamics and their relationship to an emergent phenomenon (Mittal, 2013).

This work seeks to identify the *conditions of team dynamics that explain emergent states of team agility.* To this end, the research uses a multi-level perspective of CAS and considers team agility as an emergent state. The overall

[1] http://thenewstack.io/visual-studio-online-microsofts-road-open-agile-development/

objective is to propose a framework that explains the empirical findings. Thus, the research objectives are: i) to provide an integrated summary of the literature of complex adaptive systems and agility, ii) to derive propositions and an initial framework from the integrated literature, and iii) to investigate the proposed framework empirically.

In order to identify the conditions of team dynamics, we conduct a multi-case study, whose results will help practitioners to enhance their agility (Gregory et al., 2016). The identification of specific conditions guides practitioners along the evolution of their agility. Our theoretical contribution is tripartite: First, while earlier research in the field of ASD focused on the methodology when using CAS as a theoretical lens (Alaa and Fitzgerald, 2013; Kautz, 2012; Meso and Jain, 2006), our work focuses on the SDT as the unit of analysis (cf. Moe, Dingsøyr, & Dybå, 2010; Sawyer, 2004). The focus on teams heeds the call for more research in the area of social and human factors in agile software development (Dybå and Dingsøyr, 2008). Second, we build on a wider body of knowledge from team-focused research (Arrow et al., 2000; Kozlowski and Ilgen, 2006; Mathieu et al., 2008). Therefore, we respond to the call to connect ASD research with existing streams from mature fields (Dingsøyr et al., 2008; Dybå and Dingsøyr, 2008). We leverage the concepts of self-organization and emergence as key characteristics of CAS. CAS is an established theory explaining team-related phenomena (Arrow et al., 2000; Curşeu, 2006; Kozlowski and Chao, 2012; Mathieu et al., 2008). Third, the need for more research into mature agile teams is addressed by comparing them with less mature agile teams to better understand their differences (Dingsøyr et al., 2008). We map the definitions of team agility and emergent states, and suggest team agility to be an emergent phenomenon. Initial conceptualizations to identify SDTs as CAS are conceptual contributions without empirical validation (e.g., Alaa and Fitzgerald 2013; Kautz 2012; Meso and Jain 2006). We identify differences between teams that recently adopted ASD and those that are more mature. We seek to understand the factors that enable self-organization within the team.

2. Theoretical Background

2.1. Complex Adaptive Systems

The theory of CAS stems from the idea that some systems are challenging to simulate (Holland, 1992). This idea was formalized in general system theory (e.g., Boulding 1956) and extended with the notion of complexity. A CAS has three characteristics, i.e., evolution, aggregate behavior, and anticipation, and the evolving nature of such systems provides a key challenge to research (Holland, 1992). Although academics have applied the theory of CAS in different fields, such as control theory, economics, biological cells, and games, the systems share four distinct characteristics across disciplines (Holland, 2006). First, parallelism means that the agents work in parallel while also interacting with each other. Second, the fact that an individual agent's response depends on the communication received indicates the characteristic of conditional action. Third, agents represent modularity in the sense that the formulation of a response is subject to multiple decision points within the agent at which the agents assess the situation. Fourth, agents adapt and evolve over time as part of an ongoing learning process. The behavior of a CAS is attributed to the simultaneous and parallel actions of the system's agents (Choi et al., 2001). These dynamics can be attributed to different sub-systems and lead to new emergence of the system (Goldstein, 1999).

When discussing teams as a CAS, the latter's agents are the team members. The literature presents three relevant sub-systems with their own dynamics (Arrow et al., 2000). Here, the team consists of multiple systems, for example, the individual team members, the team itself, and larger systems such as the organization or community (Mathieu et al., 2008; McGrath et al., 2000). Consequently, the literature introduced local, global, and contextual dynamics that represent changes shaping the systems. While the local dynamics represent individual activities such as tasks and using resources or technology (McGrath et al., 2000), the global dynamics include the team's behavioral variables. The teams may respond differently to various team compositions, task designs, team norms, or compelling direction, i.e., a specification of the team's purpose (Wageman et al., 2005). The contextual dynamics influence the team's direction but are outside its immediate scope. Examples in this respect are management support, corporate incentives, and talent supply (McGrath et al., 2000).

Prior literature on software development also benefits from CAS as a theoretical lens. Research into software development teams' effectiveness argues for the impact of CAS on their emergent states, such as team cognition or cohesion (Curşeu, 2006). The study also suggests the applicability of a power-law distribution toward the contribution of team members to the team's cognition. The theory of CAS has been applied to the context of ASD. In the context of information systems development, seven concepts for a CAS were identified; the article suggests that these concepts lead to emergent effects (Kautz, 2012), such as the balancing of exploration of new knowledge and the exploitation of existing knowledge (Kautz, 2011). In a conceptual work, scholars identify principles of CAS that are derived from principles and practices of ASD and suggest a contextualized conceptual framework of complex adaptive systems.(Alaa and Fitzgerald, 2013). The work highlights emergent attributes that require further exploration in ASD, e.g., diversity, pattern recognition, adaptation as a fit to environment, collaboration, and inter-connectivity. Another study investigates three leading principles when comparing agile with traditional software development (Vidgen and Wang, 2009). The principles of matching co-evolutionary change rate, optimizing self-organization, and synchronizing exploitation and exploration, finds support in later research (Kautz, 2011). An overview of these studies is presented in Table 1.

---------- TABLE 1 ----------

While the conceptualization of CAS includes three sub-systems with three different dynamics affecting the group's shaping over time, we need to understand the characteristics that explain the system behavior of a CAS. Self-organization and emergence are two important characteristics of CAS. They are often jointly discussed, and there is an ongoing debate about their relationship. While some authors understand self-organization as an emergent phenomenon (Curşeu, 2006; Mittal, 2013), others advance the idea that emergence is the result of a process that we call self-organization (Curşeu, 2006; McCarthy et al., 2006). A third characteristic of CAS is the aspect of non-linearity, i.e., the unpredictability of the whole in relation to a change of its parts. For example, a small change in a system component can lead to a significant change in the system as a whole, and vice versa. Non-linearity is a

requirement for self-organization and hence for the development of novel or emergent outcomes (Goldstein, 1999). We will discuss the concepts of self-organization and emergence in greater depth below.

2.2. Self-Organization

Self-organization is defined as "a process in a complex system whereby new emergent structures, patterns, and properties arise without being externally imposed on the system" (Goldstein, 1994). Here, it becomes apparent why the two concepts of self-organization and emergence are often discussed together. The definition suggests the authors' view of self-organization to be a process and therefore of emergence as the product. Self-organization can often be observed in nature, for example in the formation of flocking birds (Choi et al., 2001). Contrary to what some believe, mocking birds do not implement a predetermined plan or follow a lead bird when flying in formation. Rather, it is the result of self-organization whereby each bird derives its flying position from local information. Therefore, the resulting formation is a pattern or structure of emergence that we can observe. A pre-condition of self-organization is the system agent's autonomy (Maturana and Varela, 1980; Vidgen and Wang, 2009). Autonomy is "the extent to which an individual [or team] has considerable discretion and freedom in deciding how to carry out tasks" (Langfred, 2005). Without autonomy, the agent is dependent on external stimuli and guidance for a sense of direction. An external influence would prohibit the emergence of new structures and patterns through a process of self-organization. As a result, some may suggest that self-organization only comes about through a lack of control. In fact, the opposite is the case: It is the result of local dynamics and the adaptation of agents that build a new configuration of the system (McCarthy et al., 2006). Within a new configuration, the connectivity needs to be moderate. While the highly connected system agents follow a structure that is too rigid, loosely coupled system agents lack structure and lead to recursive patterns (McCarthy et al., 2006).

Within teams, the development of a hierarchy in the form of emergent leadership is an example for one form of increasing order (Goldstein, 1999). Self-organization can only occur when the system has more energy on the inside than the pressure external forces exert on it (Anderson, 1999). Hence, SD team members have to be motivated and enthusiastic about their work, which helps them to cope with negative effects for example caused by uncertainty and stress. Larger teams have more energy, as they have more team members to counter negative effects. Moreover, the team members' characteristics and their decision process require consideration as both influence the energy within the system. In contrast to regular teams that only execute the team task, self-organizing teams also monitor and manage their development process and progress (Hackman, 2002).

2.3. Emergence

The close relationship between self-organization and emergence is mirrored in the definition of emergence. Emergence is "the arising of new, unexpected structures, patterns, properties, or processes in a self-organizing system" (Goldstein, 1994). A more granular definition is offered by McCarthy et al. (2006), who define emergence as "the manifestation of new process characteristics due to the collective behavior of the agents, as opposed to the individual behavior of each agent (Anderson, 1999; Holland, 1995; Kauffman, 1995; Waldrop, 1992)" (p.444). While the former definition provides a broader view, the latter limits it to a new process characteristic. In either case, the

definition describes the rise of something new because of collective behavior within a system. Hence, emergence offers a way to experiment with and explore a new configuration and therefore makes it possible to capture a configuration that is different from the previous one. Only through this manifestation are we able to investigate and determine the outcome and use it as a feedback mechanism. If needed, proper adaptations can lead to a more desirable configuration. Over time, such adaptations with interim states help the team to evolve, which results in an emergent capability (McCarthy et al., 2006). Such an emergent capability helps the team to navigate in the quickly changing software development environment, where the challenging environment, in the form of technological innovation and new customer requirements, pressures the team to change (Börjesson and Mathiassen, 2005). The team is required to act and react quickly in such an uncertain environment. The uncertainty theory describes decision making under uncertain conditions (e.g., Gilboa 2009).

Within teams, emergence is a phenomenon that "is a pattern of behavior, a coherent structure or a state between individuals" (Curşeu, 2006, p. 251). Often, team-related literature refers to the term "emergent state," which is applied to many theoretical frameworks (e.g., Ilgen et al., 2005; Kozlowski & Ilgen, 2006). Emergent states can manifest themselves in different forms within a team, such as team trust, team cognition, or team affect. Emergent states are defined as "constructs that characterize properties of a team that are typically dynamic in nature and vary as a function of team context, input, processes and outcomes" (Marks et al., 2001, p. 357). While some may suggest emergence to be a positive phenomenon, it may also have negative effects. We find shitstorms in social media to be an example of an emergent phenomenon with dominantly negative effects (Werder et al., 2014). Most people would agree that such an emergence has led to a worse state of affairs. The same applies to the concept of emergent states. The adaptation to a new emergent state can lead to a negative result. The results of an adaptation can be lower team trust if a team member shares misleading information.

2.4. Team Agility

The literature often refers to the concept of agility, when investigating ASD. Such a concept is also presented in other reference disciplines, such as business research and supply chain management (e.g., Hoek et al. 2001). In the field of manufacturing research, a reference agility model has been proposed (Sharifi and Zhang, 1999). The agility model suggests agility drivers lead to agility by forming agile capabilities such as responsiveness, competency, flexibility, and speed. Given that flexibility is a key aspect of agility, the work by Volberda (1996) is an important source of research into organizational agility. He defines organizational flexibility in terms of the range of management capabilities and the speed at which the latter can be used to enhance the ability to control the organization and to increase the management capacity to control (Volberda, 1996). Management literature tends to focus on the ability to change when defining agility (Conforto et al., 2016). The definition of organizational agility refers to the organizations' ability to sense and respond to changes (Overby et al., 2006). This is closely linked to the initial definition from manufacturing, which expressed (manufacturing) agility as the ability to respond to change and the ability to exploit changes by taking advantage of them (Sharifi and Zhang, 1999). We provide an overview of different definitions of agility in Table 2.

---------- TABLE 2 ----------

Building on various definitions from the field of information systems and other reference disciplines, the term agility has been defined for the information systems development field to rely on two fundamental concepts, namely flexibility and leanness (Conboy, 2009). The flexibility of an information systems development method refers to the reaction and response to change but also to the initiation of change. A flexible information systems development method needs to align with its parts and its environment. The leanness of an information systems development method aims to provide additional value that customers perceive along with the economy, quality, and simplicity of the outcome. This suggests the importance of customers and users as important stakeholders and source for many changing requirements (West and Grant, 2010). Hence, we define team agility as "the continual readiness of [an SDT] to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity) through its collective components and relationships with its environment" (p.340, Conboy, 2009).

The concept of team agility supports the definition of emergent states as put forward by Marks, Mathieu, & Zaccaro (2001). Team flexibility refers to the capability of a team "to rapidly or inherently create change, embrace change and learn from change" (p.336, Conboy, 2009). Creating, embracing, and learning are dynamic elements that can vary based on a team's function, as mentioned by Marks et al. (2001). Earlier the concept of leanness was introduced. A team is lean when it focuses on the customers and their perceived value. The focus on customers' perceived value also changes dynamically with other elements (Marks et al., 2001). The elements mentioned by Marks are context, input, processes, and outcomes. Here, changing customer objectives leads to a shift in their valuation of things. Alternatively, technological changes alter the way the team wants to achieve such customer value. Consequently, we understand team agility as one form of an emergent state.

3. Conceptual Development

We leverage the input–process–state–output framework (Collins et al., 2013) as a starting point for our conceptual development. Therefore, this study focuses on the forming stage and investigates the relationship between input factors and an emergent state (Ilgen et al., 2005). We conceptualize each input factor based on prior literature and the corresponding system's description (Ilgen et al. 2005; Mathieu et al. 2008; McGrath et al. 2000). Consequently, we want to understand the relationship between the team as a CAS and its emergent state of team agility. We suggest that the team's self-organizing characteristics mediate this forming relationship. Self-organization requires that the team and its members be autonomous (Maturana and Varela, 1980; Vidgen and Wang, 2009).

The local system of a CAS refers to the individual team member (Mathieu et al., 2008; McGrath et al., 2000). Therefore, the local system is subject to different dynamics represented by the individuals' activities, such as using resources, tasks, or technologies (McGrath et al., 2000). First, within the system, the individual team member is one type of resource. Particularly in knowledge management, the experience gained by an individual or their collectives is a critical resource that builds up over time. An example of the importance of knowledge is the knowledge-based theory of the firm by Nonaka and Takeuchi (1995), which presents knowledge as a key resource

for providing a strategic advantage to the firm. The theory is based on the resource-based view of the firm and points toward knowledge as an inimitable key resource (Kozlenkova et al., 2014). Hence, an individual's experience reflects an important resource within the local system. Prior experience in the form of task related knowledge increases the individual's self-efficacy (Latham et al., 1994) and therefore, their independence in decision making. Consequently, experienced team members have more autonomy than less experienced team members do. We suggest the individual's experience enables team autonomy.

Second, the task is an example of an individual's activity in the local system. In the work context, an individual's task is closely linked to her role. The sum of the tasks form the overall job, while the job guides the tasks and provides expectations for the developer. The development team is the sum of different roles, such as project manager, developer, and designers. H a clear communication of such roles helps the team to understand the distribution of expertise and responsibilities. Understanding existing expectations toward oneself is often referred to as job clarity (Salas et al., 1999). A software developer who is clear about her job and expectations toward her can work freely on tasks with discretion. When the team has a lot of freedom to perform tasks at their discretion, the team is considered more autonomous (Langfred, 2005). Hence, job clarity of team members enables the team's autonomy.

Third, the use of technology is another element in the local system. Prior conceptualization in information systems suggests that the team members' connectivity indicates local dynamics within the CAS (Alaa and Fitzgerald, 2013). In today's digital environment, access to technology, such as communication and collaboration tools, defines the team member's connectivity. Examples are globally distributed teams, where the team members are spread out around the globe (Sarker and Sarker, 2009). Thus, without access to technology, a team member is not able to communicate effectively with others. Access to technology is critical for a team member in order to be flexible in her communication and collaboration with others. This flexibility of a team member relates to the team's autonomy. Hence, we formulate:

Proposition 1 (P1): *For agile software development teams, the higher a) individuals experience, b) job clarity, and c) technology access are, the higher is the team's autonomy.*

The global system refers to the team (Mathieu et al., 2008; McGrath et al., 2000). The team's behavioral variables indicate different dynamics within the global system (McGrath et al., 2000). Such dynamics are not only the sum of the local dynamics but reflect the network, patterns, and links of the local system (Arrow et al., 2000). First, teams naturally form a network, as they comprise multiple team members (nodes) connecting with each other by means of their interactions (edges). Patterns emerge and can be steered through incentives and the team task itself. While incentives on an individual basis may lead to conflicts within the team, it can be difficult to associate incentives in an organizational system with an individual's work. Hence, a team's incentive can ensure goal interdependency (Zhang et al., 2007). While a common goal is a requirement for a team, such a goal may not be part of the organization's incentive structure. However, established goal interdependence within the team ensures that all team members work together toward a unifying goal and influences the team's orientation. Such unity and an

achievable orientation help the team's autonomy. Hence, clear common goals enhance the team's orientation and therefore, its autonomy.

Second, the team task provides a common pattern and a link between the team members. Complexity is an important characteristic of the team task (Wood, 1986). Task complexity is especially important in the context of software development (Darcy et al., 2005). While prior studies in the area of software development investigated software complexity (Banker et al., 1998) or data complexity (Banker and Slaughter, 2000), the role of task complexity has been studied in other contexts (e.g., Argote et al. 1995). Only few studies investigate team task complexity in the context of software development (Espinosa et al., 2007), suggesting task complexity consists of the task size and structural complexity of the task. Lager tasks and structurally complex tasks often require a logical division into smaller problems. With increasing task complexity, more sub-tasks and smaller problems result. The larger the pool of problems and sub-tasks, the easier it is for the team to assign them freely at their own discretion. Such assignment can happen to the team as a hole, sub-groups, or individuals. Consequently, teams working on complex tasks have the ability to assign tasks freely and hence, are rather autonomous. Thus, higher task complexity enables team autonomy.

Third, the objective of serving the user's needs helps to build a natural link between the team members. While ASD tends to focus on functionality, its extension with and the importance of user research have been suggested (Brhel et al., 2015). User research enables the team to develop usable software and helps them to determine future requirements that will influence upcoming development cycles (Grudin, 1991). Extracting future requirements and understanding the user's needs helps the team to develop a joint team orientation. This team orientation allows the team to act autonomously during the development process. Hence, the degree of user research relates to the team autonomy. Therefore, we propose:

Proposition 2 (P2): *For agile software development teams, a) the higher goal interdependence, b) the higher team task complexity, and c) the higher user research are, the higher is the team's autonomy.*

The contextual system of a CAS is the organization and its environment (Mathieu et al., 2008; McGrath et al., 2000). Contextual dynamics nest within this system and influence the team's emergent state (Ilgen et al., 2005). In return, however, the team is unable to influence contextual dynamics, which are often organizational decisions such as development length. An organization-wide decision to adopt an organizational heartbeat pre-defines the development length. A heartbeat synchronizes existing product management and development processes (Vlaanderen et al., 2011). For example, establishing a clear development length forces re-occurring patterns within the SDT across releases. A release's development length typically reflects the frequency of such a heartbeat. A shorter development length of a release means a higher number of releases and a larger number of synchronization points. Such synchronization points help the team to learn and evolve on their own because of their autonomous behavior.

In a similar vein, management support has a positive influence on teams' self-organization. Hence, we suggest that management support within the organization is important to develop the capability of self-organization. These

organizational decisions are external forces that prevent the project team from developing autonomy (Moe et al., 2008), e.g., when they lack management support or the development length is unclear. Therefore, we formulate:

Proposition 3 (P3): *For agile software development teams, a) the higher management support, the higher the team's self-organization and b) the shorter the development length the higher the team's autonomy.*

Environmental factors and individual factors influence the teams' self-organization (Hoda et al., 2010). While self-organization is a process, emergence is the product (Curşeu, 2006; McCarthy et al., 2006). Teams' self-organization is an evolutionary process (Goldstein, 1999). As a result, we observe an emergent state within the team. Here, we observe team agility as an emergent phenomenon resulting from self-organization. Prior work also suggests the effect of team processes on teams' emergent states (Collins et al., 2013). A pre-condition of self-organization is the system agent's autonomy (Maturana and Varela, 1980). While definitions of agility vary, all seem to agree on the element of flexibility (Conboy, 2009; Lee and Xia, 2010). Hence, agility is often referred to as the responsiveness to change (Lee and Xia, 2010). Prior work has shown that autonomy has a positive effect on software development response efficiency (Lee and Xia, 2010). Therefore, we suggest:

Proposition 4 (P4): *For agile software development teams, autonomy positively influences team agility.*

Figure 1 depicts our proposed team agility framework.

---------- FIGURE 1 ----------

4. Research Method

In the following, the research method is described along with the research design, data collection, and analysis (Dubé and Paré, 2003). Given the phenomenon's complexity and its multiple systems, a holistic multi-case study is the applied research approach (Yin, 2008). Furthermore, we would like to test our team agility framework using a positivist case study.

4.1. Research Design

A multi-case study using theoretical sampling logic was selected as the main research design (Eisenhardt, 1989). Companies are actively approached to identify cases that vary in terms of their agility. We investigate three states of agility, i.e., starting the transition to agility, being in the middle of the agile transition, and being more mature with agile development. Through the varying degrees of agility as the emergent state, we can observe differences in self-organization and CAS. An introductory discussion between the researchers and the case organization determines whether a case is suitable. Both sides commit resources as part of the discussion, and we assess the degree of agility (i.e., to determine the applied development method and the time of its introduction). The later in-depth analysis of the cases confirmed the initial assessment.

Moreover, relying on multiple cases increases the external validity of the findings (Yin, 2008). In total, the study encompasses three case teams from three different organizations. Within each team, at least four different perspectives are obtained: a) management or Mgt. (e.g., product manager, portfolio manager or general manager), b) technical supervisor or Tec. Sv. (e.g., development team leader or head of development), c) developer or Dev. (e.g., junior engineer or senior engineer), and d) designer or Des. (e.g., interaction designer, usability engineering or designer). Investigating at least four perspectives helps us to identify differences between the roles but also to identify common patterns across the team. Table 1 presents an overview of the 16 interviews and the different roles of the interviewees. Each team's interviews were conducted on the same day. Team ALPHA also served as our pilot case, leading to some adjustment and refinement of the semi-structured questionnaire. While the first authors conducted all the interviews, key deliverables such as case reports and interview guidelines are the subject of a discussion within the author team. All teams received the results, and two of three teams were open to an in-person feedback session led by both researchers. The feedback session includes a presentation of the case report's results and allows the case organization to make additional comments.

---------- TABLE 3 ----------

4.2. Case Description

Team ALPHA is a small software development organization with fewer than 25 employees. The founder and owner runs the organization. The main product helps sport associations to manage their business. The organization values sports and serves many sports clubs in Europe. Its development focuses on a core product that assists in the management and operation of sports clubs. The core SDT encompasses 11 team members; seven of them are based remotely in Russia, while four work at the organization's headquarters in Germany. The remote team members have been working for the company for many years. Overall, the team is very specialized with little cross-functionality. The development length of the investigated release is greater than 12 months. Although the organization is mostly located in Germany, the customer market stretches beyond Germany's boarders to include France, Austria, and South Africa as key markets. The team only recently adopted ASD and focuses on evolutionary elements during the development, such as early delivery and welcoming change. They also use basic agile practices, such as sprint-meeting and a backlog. However, the team's use of agile practices has not been ritualized yet.

Team BETA describes a larger medium-sized organization with fewer than 400 employees. The organization is a public entity and part of a larger holding. The organization focuses on the financial services sector. Its objective is to develop innovative and intelligent solutions tailored to the individual user's needs. The organization offers products for controlling, product management, value-based product consulting, and a back-end library. The product investigated is a dashboard application that helps to visualize financial information. The SDT has seven members; five of them are developers, one is a scrum master, and the other is a product owner. While four developers

work close to the customer in Frankfurt area, three members are based at the organization's headquarters in Germany. Cross-functionality within the team is low. The development length is 12 months. The organization is a subsidiary of a larger international firm, and the customer base is mainly German speakers. The team adopted scrum as their leading agile method and have learned to improve it since its introduction two years ago. The team collaborates through face-to-face meetings and tracks their progress through working software. They adopted sprints with a 4-week cycle.

Team GAMMA is a medium-sized software development organization. The organization, a subsidiary of a publicly traded holding, has fewer than 100 employees and focuses on the development of administration add-ons for collaboration platforms. The organization also develops software in the area of business process management, mobility, and security. Here, the team develops an email management system with enhanced security measures. The SDT has 11 members, five of whom are developers. The team has two product managers, a designer, and three quality assurance members. All members are located at the same German site. The team does not extend functional boundaries. The development length of the release investigated is six months. Customers are globally dispersed at locations in USA, Canada, India, and Germany. Team GAMMA is the team with the highest agile maturity, having adopted scrum and used it for the past three to four years. More so, they focus on their people and reflect on their prior experience to improve their agile practices. The team adopts a 2-week cycle and grooms their backlog, i.e., the continuous update of their backlog based on recent discoveries.

4.3. Preparation and Data Collection

Publicly available information (e.g., through websites) is analyzed as a first step in the preparation of the case study. This information helps to form an understanding of the business context and products. In addition, this information reveals organizational information, such as team members and responsibilities. There is particular focus on the preparation of the semi-structured interviews. The development of an interview guideline enhances the main data collection method. During the development of the questions, we avoided any suggestive forms. Such guidelines ensure that permission is received to record the interview, ask pre-defined questions, provide room for notes, and a one-page survey at the end. The survey captured demographic information related to the individual and the team. Information related to the individuals included the years of experience in usability engineering/interaction design, traditional and agile software development. Team related information included the team size, team member's roles, and their location.

The investigators' skills have proved to be a critical factor in carrying out case study research (Yin, 2008). A separate preparation interview with a practitioner helps us to improve our interview guidelines. The practitioner has many years of experience in software development and has a management role in an international software development organization. This helps to prepare the investigator and train the interviewer for data collection in the field. It also provides another means to evaluate the interview guideline. The data sources and their use are an important step to assure the results' reliability and validity (Benbasat et al., 1987), and Table 1 provides an overview. For participation in the case study, we require interviews with at least four different roles to get a comprehensive understanding of team level perspectives. Prior communication and alignment between the case company

and the researchers assured the correct mapping of the company's internal roles to one of the four perspectives. Key decisions, along with the case study's purpose, data collection, and interview questions, are documented in a case study protocol prior to the collection of data for the initial case (Yin, 2008). A similar investigation of the cases increases the findings' reliability.

The data collection relies on multiple sources and triangulates documentary evidence, such as websites, emails, and screenshots, with interviews and observations, as well as survey data (Yin, 2008). The use of multiple sources also increases the validity of constructs used and allows for triangulation (Yin, 2008). However, the key data collection method is semi-structured retrospective interviews (Schultze and Avital, 2011). The use of laddering interviews helped to extract antecedents and their impact. Laddering interviews use comparisons and contrasts to identify patterns that are subsequently understood in greater depth through repeated inquiry related to the how and the why of such patterns (Schultze and Avital, 2011). We conducted the interviews in German and English, following the prepared interview guidelines. Each interview lasted approximately 60 minutes, ranging between 50 and 67 minutes. The interviewer took additional notes before and after the interview to document informal information exchanges and non-verbal information. Table 1 presents an overview of the different cases and their interview length. We conducted all interviews during face-to-face meetings at the company's main location. The transcripts fed the case study database using NVivo to support the coding process.

4.4. Data Analysis

The data analysis includes three steps: preliminary data analysis, within-case analysis and cross-case analysis. During our first step, we use preliminary data analysis techniques and tools (Dubé and Paré, 2003). Using a predefined set of codes, a systematic extraction of interview data helps the later analysis. The initial list of codes included different factors as the result of a literature study on agile teams that also informed the framework. A coding sheet including the codes name, a definition from literature, and example text from the interview data that matches the code, documents the codes. Nevertheless, while the coding is performed, existing codes require adjustment, and new codes emerge (Seaman, 1999). For those cases where a statement does not match any existing code, a new code emerges. This process lead to three tables with 237, 184, and 234 coded text segments. The final list of codes, including their empirical observations, presents the relationship between the conceptual and the empirical level (see Table 2). The use of data displays during the early stages, but also during later analysis stages assisted the researchers to visualize and discuss different interim results.

Second, we conducted within-case analysis for each case. We derive the number of words identified for each code, which indicates the importance of each code: If a factor is not important or has no influence, the interviewee will have little to say. We also analyze such descriptive data along with the various roles and cases. While they only provide an indication, the information certainly requires completion with qualitative data from other sources and a content analysis. Given the assessment of each case with different team members in different roles, we are able to generate a rich case report. The use of multiple roles and different sources of evidence allows us to triangulate findings. Each case concludes with a case report, discussing background information of the company, its status, and future recommendations. Each report builds on a logical chain of evidence (Yin, 2008), taking also contextual

information into consideration. The presentation of such reports allows us to evaluate the findings and recommendations.

Third, we aggregate the data sequentially in order to conduct a cross-case analysis. In an initial step, we extract all data into a spreadsheet that presents all coded quotes. The quotes are a grouped list based on the combination of factors and interviewees. In other words, in this step we consider the full set of codes (number of interviewees times the number of codes). All quotes of a given group present an initial summary. The next step is to aggregate all the summaries for a given code and to structure them along our initial framework. We conduct a cross-case comparison to assess the replicability and variance of individual factors (Yin, 2008). Here, we investigate the influence of such factors and use of CAS as a theoretical lens, which helps us to structure our findings. We investigate additional codes, as there might be some control factors. Moreover, we compare the findings with factors and effects from existing literature. Interim results are discussed between the authors and presented to other scholars in order to strengthen the findings.

---------- TABLE 4 ----------

5. Findings Cross-Case Analysis: Explaining Different States

The teams studied in this article indicated different emergent states of team agility. While we present the within-case analyses of each case in Appendix A, we identify commonalities and differences across the team in this section. The analysis helps us to investigate conditions that lead to different emergent states of team agility.

The case analyses show that all three teams have different emergent states of team agility. As a result, we present an overview of the three cases (see Table 3). With the exception of technology access and team task complexity, the team's antecedents vary across the cases. The *individual's experience* is one of the team's critical resources. The team members' experience related to ASD varies. Individual experience can either facilitate autonomy, as in the case of GAMMA, but it can also prevent autonomy, as indicated by team ALPHA. Hence, when team members gain experience they are less dependent on external stimuli and thus, increase their autonomy. *Job clarity* requires that the individual understands her role and tasks associated with that role in order to manage expectations and to gain autonomy. In an environment where roles and responsibilities are not clearly defined, team members struggle to discretely decide on their tasks (e.g., ALPHA). When it becomes clear what is expected of each role, people have more freedom on how to meet such expectations as seen in case GAMMA. A team member does not have to be assigned to a single project. Rather, when resources are scarce, a team member can work on two or more projects, while still having more freedom over the execution of their tasks (e.g., BETA). *Technology access* enhances a team member's flexibility. Given the ubiquity of technological advances within the software industry, easy access to new technology seems to be a mandatory requirement for developers in order to perform well. New technology also enhances the coverage of developers' needs. While technology access is important, we did not find any differences between the teams. Rather it seems to be a boundary condition for software development in

general. In summary, an individual's experience, job clarity, and access to technology characterize the local system; yet, the teams had a good access to technology.

While team task complexity remains stable within the global system, we find difference in the goal interdependence and user research across the teams. A formal *goal interdependence* that would enhance the leanness of the team is absent in all cases. Yet, we find organizational goals that influence team member rewards and therefore, reduce the teams' dependency on external stimuli (BETA and GAMMA). Team ALPHA does not have a rewards structure that targets the enhancement of joint efforts. *Team task complexity* inhibits team autonomy by introducing chaos and limiting their ability to create redundancies. Teams that develop highly complex products are less autonomous. All three teams report high task complexity, suggesting team task complexity to be a boundary condition applicable to development tasks. The access to and knowledge of users through *user research* help the team to becoming more independent in their task fulfillment. When the team has easy access to users and has already build a substantial knowledge base about its users, they develop a joint vision and are less dependent on other functions to clarify requirements and needs, as in the case of BETA and GAMMA. Contrary, ALPHA depended heavily on the expertise and network of the general manager and therefore, has less freedom in structuring their tasks. Hence, we find the team's goal interdependence, team task complexity, and user research reflect the local system; yet, all teams have a high team task complexity.

---------- TABLE 5 ----------

The contextual system, represented by the management support and development length, varies across the three teams. *Management support* plays an important role in defining boundary conditions that allow the team to self-organize. Within Team ALPHA, support by management is rather superficial, making it increasingly difficult for the team to discretely decide on means to accomplish their tasks. While team BETA receives moderate management support, team GAMMA benefits from true management commitment. Thus, the team and management have a joint vision and understanding of the overall goals and accomplishments, which the team seeks to pursue rather independently. The *development length* resides in the organizational system and influences the team orientation. The organization seeks to introduce a heartbeat that establishes clear deadlines synchronizing the work at critical milestones. The development length observed varies from six months to more than 12 months. The teams with a shorter development length have smaller cycles in which they can adapt and self-organize. Hence, decisions and experiences that only occur at the end of such a cycle happen more frequently. This increased frequency of feedback helps the team to gather more experience and enhance their team orientation. Therefore, it allows the team to self-organize over time. Here, we suggest that the management support to respond to user-related needs and the development length are important.

6. Discussion

The paper proposes, empirically tests, and extends a team agility framework. This section starts by presenting the contribution. Next, we discuss the theoretical and practical implications before pointing out the study's key limitations.

6.1. Theoretical Contribution

We suggest three theoretical contributions. *First*, this study takes a different perspective by adopting the team as the unit of analysis, whereas prior research focused on the development methodology (cf. Alaa and Fitzgerald 2013; Kautz 2012; Meso and Jain 2006). While this process-oriented view toward agility and the adoption of ASD has its benefits, this study takes a different view. Given that the agile manifesto puts a great deal of emphasis on the importance of individuals and their interactions, suggesting a psychological and social phenomenon, this study investigates team agility as an emergent state. Therefore, this study also responds to calls for more research related to human and social aspects (Moe et al., 2010). Adopting a team-level perspective helps us to understand the nature of the agile phenomenon from a team theoretical perspective. While prior research on team theory investigates emergent states, such as trust, cohesion, and emotions within teams, this study suggests team agility as an emergent team phenomenon. This corresponds to practitioner reports that indicate becoming agile takes up to four years. On the one hand, the team learns about agile practices. On the other hand, the team has to adjust to different environmental conditions and expectations, such as increased autonomy and cohesion. Therefore, team agility may not be limited to software development, but can be investigated in different contexts. Prior reports already present the application of agile practices in physical product firms, such as LEGO (Cooper and Sommer, 2016). This suggests a broader implication of team agility within team research. Teams that resemble similar characteristics as software development teams may experience similar benefits from introducing ASD.

Second, following calls for more theory-based research (cf. Dybå and Dingsøyr 2008), the study uses the theory of CAS. Using CAS as a theoretical lens, the study facilitates an understanding of the characteristics of self-organization and emergence within the context of SDT. Following the representation of a local, global, and contextual system, their influence on the self-organization of the team is determined (p1-3). The results suggest that the local system is associated with the self-organization of the team (p1). Here, we find support in job clarity (p1a) and individual experience (p1b), which help the team to withstand outside pressure (Anderson, 1999). For emergence to occur, the team needs to have more energy within the system than they experience from outside. These results specify prior conceptual work in the context of virtual teams. Individuals, the way they interact, the team composition, and the IT they use to accomplish their tasks have been suggested to influence local dynamics (Curşeu, 2006). When an individual knows the expectations toward her and her team members, the team is more likely to self-organize. Similarly, the more experience the individual team members have, the more likely they are to be autonomous, supporting the self-organization of the team. Prior studies observed reciprocal team learning effects as a result of sharing knowledge and experience (Kautz, 2012). The individual's experiences provide an indication for the depth of knowledge someone can share with the team. This knowledge is typically difficult to

imitate and provides great value to the team (Kozlenkova et al., 2014). Contrary to our proposition, we could not find support for p1c (technology access). The individual's access to technology rather seems to be a mandatory condition captured by the studies' context of software development.

The results also suggest that global system relates to the self-organization of the team (p2). We found that goal interdependence (p2a) and user research (p2c) influence the self-organization by assisting the teams' monitoring and managing their own processes (Hackman, 2002). Goal interdependence and user research help the team to identify their common goals. Such a goal becomes the common denominator that drives the team's and the individual's autonomy and therefore, improves their self-organization. Prior study suggests the importance of effective interaction with customers and users, as well as the importance of collective mindfulness to enhance agility (Vidgen and Wang, 2009). Effective interactions with customers and users help to continuously gather requirements and to drive the business value. Collective mindfulness helps the team to share responsibility and establish team discipline. Contrary to our suggestion, we did not find support for an influence by team task complexity (p2b). Since software development in general is associated with complex tasks (Maruping et al., 2009), the selected context seems to predefine the team task complexity. In addition, outcome interdependence is important when investigating collective actions as it is the case with team agility (Maruping et al., 2009).

Contextual system can restrict or enable self-organization within the team (p3). We find management support (p3a) to be associated with the self-organization as it can strengthen the forces of self-organization within the team and prevent external forces from limiting self-organization (Anderson, 1999). On the one hand, we find that the level of support is very important, i.e., not only a verbal commitment but also the corresponding actions and financial support should follow. On the other hand, simply articulating the management's stance toward certain issues already increases transparency, and hence the team can take appropriate actions that are in line with management expectations. Prior conceptual work suggested the influence of organizational context through structure and supervision to influence contextual dynamics (Ilgen et al., 2005). Thus, it is necessary to avoid spending resources on objectives that will not find support from management. Otherwise, management perceives a loss of authority and consequently, wants to increase control, having negative effects on the level of autonomy and agility within the team (Vidgen and Wang, 2009). Both strengthen the autonomy of the team and thus, relate to their self-organization. In addition, a shorter development length (p3b) within the team increases the feedback cycles and learning mechanisms. Prior studies suggested a sustainable work rhythm in order to establish time-pacing in short iterative cycles (Kautz, 2012; Vidgen and Wang, 2009). As a result, the interaction frequency increases and requirements can be discussed in greater depth. Shorter development length enables the team to discuss different releases and integrate new learnings from user and customer feedback that shifts the understanding of needs. Both, increased management support and a shorter development length are important for the team to enable team orientation and self-organization.

Higher extent of self-organization helps the team to improve its agility (p4). First, the team needs to have the corresponding autonomy that is fundamental to self-organization (Moe et al., 2008). Team members need to receive the information necessary to make informed decisions. Thereby, discretion and freedom in making decisions are fundamental (Langfred, 2005). Without autonomy, the team relies on external stimuli and guidance. Second,

the team members need to have the freedom to make mistakes and learn from those mistakes as an ongoing journey toward self-organization and self-regulation (McCarthy et al., 2006). The team evolves and establishes different configurations. Thus, the team increases its repertoire about possible configurations and learns to distinguish between configurations that are reconcilable and configurations that are irreconcilable with the idea and principles of agility. In those cases, we observe the highest extent of team agility. We also suggest boundary conditions, such as team task complexity and technology access. However, the context seems to dominate these boundary conditions.

Our *third* contribution is the investigation of differences between immature and mature agile teams (cf. Dingsøyr et al. 2008). We identify boundary conditions that influence the team's self-organization and therefore, their agility. Such conditions help to understand the characteristics of mature agile teams and explain how autonomy can be achieved (Moe et al., 2008). While prior research suggests the performance impact of agility (Lee and Xia, 2010), this study investigates boundary conditions of team agility. In line with prior studies suggesting a close relationship between self-organization and agility (Vidgen and Wang, 2009), the study suggests self-organization to mediate the effects of other team factors on team agility. Moreover, the results help immature teams to identify areas of improvement when progressing on their agile journey.

6.2. Practical Implications

We find two practical implications of this study. First, practitioners receive a list of influential characteristics. Practitioners can leverage the characteristics in the form of a checklist in order to assess the status quo of SDTs. As a result, they can investigate weak characteristics and strengthen others in order to increase the agility of the team. This is especially useful for teams struggling to move from a process-centered view of agility toward a mature and cost-effective state of agility. Second, practitioners need to be aware of the importance of a common and joint objective. Goal interdependence is one means to formally embed such objectives into an organization. They can also develop organically by adopting user research. For those cases where the organization lacks the expertise to conduct user research, consulting agencies, or experts can extend the team's expertise. They also provide another view onto the issues and challenges faced, allowing the organization to benefit from their experience and expertise.

The research also has two limitations. First, this work seeks to understand different states of team agility and suggests the means to achieve them. Therefore, we focus on the direct relationships of the different systems toward team agility. Hence, future research could investigate the relationships between the systems. Second, while there are potential replications of the concepts to other agile teams, e.g., in new product development, the context of this study does not allow us to generalize to such teams. Hence, future research could replicate aspects of this study to teams beyond agile SDT.

7. Conclusion

This paper proposes, tests, and extends a team agility framework. It contributes to the body of literature by developing and investigating theory-based propositions. Contrary to prior studies, this work identifies agility as a team-

level phenomenon. Starting with a review of extant literature on CASs and ASD in the context of teams, we formulate theory-based propositions. Agile SDTs are one form of a CAS. The study focuses on the characteristics leading to self-organization and suggests team agility as an emergent phenomenon within SDTs. Consequently, this work helps to understand the characteristics of self-organizing teams. We understand teams as one form of a CAS. Furthermore, team agility is an emergent phenomenon that evolves over time, and it is the result of a process of self-organization.

The CAS comprises three systems, i.e., the local, global, and contextual system (Holland, 1992). In the local system, the study finds job clarity and individual experience to enhance self-organization. Within the global system, goal interdependence and user research improve self-organization. Within the contextual system, management support and development length can help the team to self-organize. Technology access and team task complexity are mandatory conditions within the context of SDT.

Future studies are needed to investigate team agility in further depth. While this study adopts a cause and effect view toward explaining team agility, future studies can investigate the phenomenon longitudinally and develop a process-theoretical view toward team agility. Prior research provides support for an event-driven and process-oriented view of team dynamics (Marks et al., 2001). Such process-theoretical perspective helps to understand the evolutionary elements in greater depth. Studies can shed further light into the different configurations of the team. Furthermore, the study relies on different agile development teams that have been studied in the field, rather than a laboratory environment. Hence, further studies are needed in order to understand the nomological net in greater depth by quantifying effect sizes.

References

Alaa, G. and Fitzgerald, G. (2013), "Re-Conceptualizing Agile Information Systems Development as a Complex Adaptive System", *Emergence: Complexity & Organization*, Vol. 15 No. 3, pp. 1–20.

Anderson, P. (1999), "Perspective: Complexity Theory and Organization Science", *Organization Science*, Vol. 10 No. 3, pp. 216–232.

Argote, L., Insko, C.A., Yovetich, N. and Romero, A.A. (1995), "Group Learning Curves: The Effects of Turnover and Task Complexity on Group Performance1", *Journal of Applied Social Psychology*, Vol. 25 No. 6, pp. 512–529.

Arrow, H., McGrath, J.E. and Berdahl, J. (2000), *Small Groups as Complex Systems: Formation, Coordination, Development, and Adaptation*, SAGE Publications, Thousands Oaks, CA, USA, available at: http://books.google.com/books?hl=en&lr=&id=_UhbhVvGeQQC&oi=fnd&pg=PR7&dq=small+groups+as+complex+systems:+formation,+coordination,+development,+and+adaptation&ots=NFR2DVV0m-&sig=YjHgb5WAi70a1MvI61fPipVD0zA (accessed 9 September 2014).

Banker, R.D., Davis, G.B. and Slaughter, S. a. (1998), "Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study", *Management Science*, Vol. 44 No. 4, pp. 433–450.

Banker, R.D. and Slaughter, S. a. (2000), "The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement", *Information Systems Research*, Vol. 11 No. 3, pp. 219–240.

Benbasat, I., Goldstein, D.K.D. and Mead, M. (1987), "The Case Research Strategy in Studies of Information Systems", *MIS Quarterly*, Vol. 11 No. 3, pp. 369–386.

Börjesson, A. and Mathiassen, L. (2005), "Improving software organizations: agility challenges and implications", *Information Technology & People*, Vol. 18 No. 4, pp. 359–382.

Boulding, K.E. (1956), "General Systems Theory-The Skeleton of Science", *Management Science*, Vol. 2 No. 3, pp. 197–208.

Brhel, M., Meth, H., Maedche, A. and Werder, K. (2015), "Exploring principles of user-centered agile software development: A literature review", *Information and Software Technology*, Elsevier B.V., Vol. 61 No. MAY, pp. 163–181.

Campanelli, A.S. and Parreiras, F.S. (2015), "Agile methods tailoring – A systematic literature review", *Journal of Systems and Software*, Elsevier Ltd., Vol. 110, pp. 85–100.

Choi, T.Y., Dooley, K.J. and Rungtusanatham, M. (2001), "Supply networks and complex adaptive systems: Control versus emergence", *Journal of Operations Management*, Vol. 19 No. 3, pp. 351–366.

Collins, A.L., Lawrence, S.A., Troth, A.C. and Jordan, P.J. (2013), "Group affective tone: A review and future research directions", *Journal of Organizational Behavior*, Vol. 34 No. S1, pp. S43–S62.

Conboy, K. (2009), "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development", *Information Systems Research*, Vol. 20 No. 3, pp. 329–354.

Conforto, E.C., Amaral, D.C., da Silva, S.L., Di Felippo, A. and Kamikawachi, D.S.L. (2016), "The agility construct on project management theory", *International Journal of Project Management*, Elsevier Ltd and Association for Project Management and the International Project Management Association, Vol. 34 No. 4, pp. 660–674.

Cooper, R.G. and Sommer, A.F. (2016), "From Experience: The Agile–Stage-Gate Hybrid Model: A Promising New Approach and a New Research Opportunity", *Journal of Product Innovation Management*, Vol. 33 No. 5, pp. 513–526.

Curşeu, P.L. (2006), "Emergent states in virtual teams: a complex adaptive systems perspective", *Journal of Information Technology*, Vol. 21 No. 4, pp. 249–261.

Darcy, D., Kemerer, C., Slaughter, S. and Tomayko, J. (2005), "The structural complexity of software an experimental test", *IEEE Transactions on Software Engineering*, Vol. 31 No. 11, pp. 982–995.

Dingsøyr, T., Dybå, T. and Abrahamsson, P. (2008), "A preliminary roadmap for empirical research on agile software development", *Proceedings - Agile 2008 Conference*, IEEE, Toronto, ON, CA, pp. 83–94.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2012), "A decade of agile methodologies: Towards explaining agile software development", *Journal of Systems and Software*, Vol. 85 No. 6, pp. 1213–1221.

Dubé, L. and Paré, G. (2003), "Rigor in information systems positivist case research: current practices, trends, and recommendations", *MIS Quarterly*, Vol. 27 No. 4, pp. 597–635.

Dybå, T. and Dingsøyr, T. (2008), "Empirical studies of agile software development: A systematic review", *Information and Software Technology*, Vol. 50 No. 9–10, pp. 833–859.

Eisenhardt, K.M. (1989), "Building Theories from Case Study Research.", *Academy of Management Review*, Vol. 14 No. 4, pp. 532–550.

Espinosa, J.A., Slaughter, S., Kraut, R. and Herbsleb, J. (2007), "Familiarity, complexity, and team performance in geographically distributed software development", *Organization Science*, Vol. 18 No. 4, pp. 613–630.

Gilboa, I.I. (2009), "Theory of Decision under Uncertainty", *Economics and Philosophy*, Cambridge University Press, Cambridge, UK, Vol. 26 No. July, p. xiv, 215.

Gill, A.Q., Henderson-Sellers, B. and Niazi, M. (2016), "Scaling for agility: A reference model for hybrid traditional-agile software development methodologies", *Information Systems Frontiers*, Information Systems Frontiers, pp. 1–27.

Goldstein, J. (1994), *The Unshackled Organization: Facing the Challenge of Unpredictability through Spontaneous Reorganization*, Productivity Press, Inc., Portland, OR, USA.

Goldstein, J. (1999), "Emergence as a Construct: History and Issues", *Emergence*, Vol. 1 No. 1, pp. 49–72.

Goldstein, J. (2000), "Emergence: A construct amid a thicket of conceptual snares", *Emergence*, Vol. 2 No. 1, pp. 2–5.

Gregory, P., Barroca, L., Sharp, H., Deshpande, A., Taylor, K., Uk, A.D.A. and Uk, K.A. (2016), "The Challenges That Challenge: Engaging With Agile Practitioners' Concerns", *Information and Software Technology*, Elsevier B.V., Vol. 0, pp. 1–13.

Grudin, J. (1991), "Interactive systems: Bridging the gaps between developers and users", *Computer*, Vol. 24 No. 4, pp. 59–69.

Hackman, J.R. (2002), *Leading Teams: Setting the Stage for Great Performances*, Harvard Business Review Press, Boston, MA.

Hoda, R., Noble, J. and Marshall, S. (2010), "Organizing self-organizing teams", *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, Vol. 1, ACM Press, Cape Town ZA, pp. 285–294.

van Hoek, R.I., Harrison, A. and Christopher, M. (2001), "Measuring agile capabilities in the supply chain", *International Journal of Operations & Production Management*, Vol. 21 No. 1/2, pp. 126–148.

Holland, J. (1992), "Complex adaptive systems", *Daedalus*, Vol. 121 No. 1, pp. 17–30.

Holland, J. (2006), "Studying complex adaptive systems", *Journal of Systems Science and Complexity*, Vol. 19 No. 1, pp. 1–8.

Ilgen, D.R., Hollenbeck, J.R., Johnson, M. and Jundt, D. (2005), "Teams in organizations: from input-process-output models to IMOI models.", *Annual Review of Psychology*, Vol. 56, pp. 517–43.

Kautz, K. (2011), "Investigating the design process: participatory design in agile software development", *Information Technology & People*, Vol. 24 No. 3, pp. 217–235.

Kautz, K. (2012), "Beyond simple classifications: contemporary information systems development projects as complex adaptive systems", *Thirty Third International Conference on Information Systems*, AISeL, Orlando, FL, USA, pp. 1–20.

Kozlenkova, I. V., Samaha, S. a. and Palmatier, R.W. (2014), "Resource-based theory in marketing", *Journal of the Academy of Marketing Science*, Vol. 42 No. 1, pp. 1–21.

Kozlowski, S. and Ilgen, D. (2006), "Enhancing the effectiveness of work groups and teams", *Psychological Science in the Public Interest*, Vol. 7 No. 3, pp. 77–124.

Kozlowski, S.W.J. and Chao, G.T. (2012), "The Dynamics of Emergence: Cognition and Cohesion in Work Teams", *Managerial and Decision Economics*, Vol. 33 No. 5–6, pp. 335–354.

Langfred, C.W. (2005), "Autonomy and Performance in Teams: The Multilevel Moderating Effect of Task Interdependence", *Journal of Management*, Vol. 31 No. 4, pp. 513–529.

Latham, G.P., Winters, D. and Locke, E.A. (1994), "Cognitive and motivational effects of participation: a mediator study", *Journal of Organizational Behavior*, Vol. 15 No. 1, pp. 49–63.

Lee, G. and Xia, W. (2010), "Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility", *Management Information Systems Quarterly*, Vol. 34 No. 1, pp. 87–114.

Marks, M., Mathieu, J.E. and Zaccaro, S. (2001), "A temporally based framework and taxonomy of team processes", *Academy of Management Review*, Vol. 26 No. 3, pp. 356–376.

Maruping, L.M., Venkatesh, V. and Agarwal, R. (2009), "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements", *Information Systems Research*, Vol. 20 No. 3, pp. 377–399.

Mathieu, J.E., Maynard, M.T., Rapp, T. and Gilson, L. (2008), "Team Effectiveness 1997-2007: A Review of Recent Advancements and a Glimpse Into the Future", *Journal of Management*, Vol. 34 No. 3, pp. 410–476.

Maturana, H.R. and Varela, F.J. (1980), *Autopoiesis and Cognition*, Vol. 42, Springer Netherlands, Dordrecht, NL, available at:https://doi.org/10.1007/978-94-009-8947-4.

McCarthy, I.P., Tsinopoulos, C., Allen, P. and Rose-Anderssen, C. (2006), "New Product Development as a Complex Adaptive System of Decisions", *Journal of Product Innovation Management*, Vol. 23 No. 5, pp. 437–456.

McGrath, J.E., Arrow, H. and Berdahl, J.L. (2000), "The Study of Groups: Past, Present, and Future", *Personality and Social Psychology Review*, Vol. 4 No. 1, pp. 95–105.

Meso, P. and Jain, R. (2006), "Agile software development: adaptive systems principles and best practices", *Information Systems Management*, Vol. 23 No. 3, pp. 19–30.

Mittal, S. (2013), "Emergence in stigmergic and complex adaptive systems: A formal discrete event systems perspective", *Cognitive Systems Research*, Elsevier B.V., Vol. 21, pp. 22–39.

Moe, N.B., Dingsøyr, T. and Dybå, T. (2010), "A teamwork model for understanding an agile team: A case study of a Scrum project", *Information and Software Technology*, Elsevier B.V., Vol. 52 No. 5, pp. 480–491.

Moe, N.B., Dingsøyr, T., Dybå, T. and Ict, S. (2008), "Understanding Self-organizing Teams in Agile Software Development", *19th Australian Conference on Software Engineering Understanding*, IEEE, Perth, WA, AU, pp. 76–85.

Nonaka, I. and Takeuchi, H. (1995), *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, 1st ed., Oxford University Press, Oxford, UK.

Overby, E., Bharadwaj, A. and Sambamurthy, V. (2006), "Enterprise agility and the enabling role of information technology", *European Journal of Information Systems*, Vol. 15 No. 2, pp. 120–131.

Salas, E., Rozell, D., Mullen, B. and Driskell, J.E. (1999), "The Effect of Team Building on Performance: An Integration", *Small Group Research*, Vol. 30 No. 3, pp. 309–329.

Sarker, S. and Sarker, S. (2009), "Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context", *Information Systems Research*, Vol. 20 No. 3, pp. 440–461.

Sawyer, S. (2004), "Software development teams", *Communications of the ACM*, Vol. 47 No. 12, pp. 95–99.

Schultze, U. and Avital, M. (2011), "Designing interviews to generate rich data for information systems research",

Information and Organization, Elsevier Ltd, Vol. 21 No. 1, pp. 1–16.

Seaman, C.B. (1999), "Qualitative methods in empirical studies of software engineering", *IEEE Transactions on Software Engineering*, Vol. 25 No. 4, pp. 557–572.

Serrador, P. and Pinto, J.K. (2015), "Does Agile work? — A quantitative analysis of agile project success", *International Journal of Project Management*, Elsevier Ltd, Vol. 33 No. 5, pp. 1–12.

Sharifi, H. and Zhang, Z. (1999), "A methodology for achieving agility in manufacturing organisations: An introduction", *International Journal of Production Economics*, Vol. 62 No. 1–2, pp. 7–22.

Silva, F.S., Soares, F.S.F., Peres, A.L., Azevedo, I.M. De, Vasconcelos, A.P.L.F., Kamei, F.K. and Meira, S.R.D.L. (2015), "Using CMMI together with agile software development: A systematic review", *Information and Software Technology*, Elsevier B.V., Vol. 58, pp. 20–43.

Vidgen, R. and Wang, X. (2009), "Coevolving Systems and the Organization of Agile Software Development", *Information Systems Research*, Vol. 20 No. 3, pp. 355–376.

Vlaanderen, K., Jansen, S., Brinkkemper, S., Jaspers, E., Jansen, S. and Jaspers, E. (2011), "The agile requirements refinery: Applying SCRUM principles to software product management", *Software Product Management, IWSPM*, IEEE, Vol. 53 No. 1, pp. 1–10.

Volberda, H.W. (1996), "Toward the Flexible Form: How to Remain Vital in Hypercompetitive Environments", *Organization Science*, Vol. 7 No. 4, pp. 359–374.

Wageman, R., Hackman, J.R. and Lehman, E. (2005), "Team Diagnostic Survey: Development of an Instrument", *The Journal of Applied Behavioral Science*, Vol. 41 No. 4, pp. 373–398.

Werder, K., Helms, R.W. and Jansen, S. (2014), "Social Media for Success: A Strategic Framework", *Pacific Asia Conference on Information Systems 2014*, AISeL, Chengdu, PRC, p. Paper 92.

West, D. and Grant, T. (2010), *Agile Development: Mainstream Adoption Has Changed Agility*, *For Application Development & Program Management Professionals January*, Cambridge, MA, USA, available at: http://www.ca.com/us/~/media/Files/IndustryResearch/forrester-agile-development-mainstream-adoption.pdf.

Wood, R.E. (1986), "Task complexity: Definition of the construct", *Organizational Behavior and Human Decision Processes*, Vol. 37 No. 1, pp. 60–82.

Yin, R.K. (2008), *Case Study Research: Design and Methods*, 4th ed., SAGE Publications, Thousands Oaks, CA, USA.

Zhang, Z.-X., Hempel, P.S., Han, Y.-L. and Tjosvold, D. (2007), "Transactive memory system links work team characteristics and performance.", *The Journal of Applied Psychology*, Vol. 92 No. 6, pp. 1722–1730.

Appendix A - Within Case Analysis

Team ALPHA: Little Experience and Unclear Expectations

Team ALPHA has the least agile experience with 1.5 years per team member on average. While the technical supervisor is experienced with agile methods, most team members have little experiences and therefore, rely on the supervisor's experience. Besides relying on individual's experience, a team can specifically assign agile roles to the team member to enhance their role clarity. Here, on the contrary, the assignment of two roles, such as scrum master and developer, to a single individual indicates less clarity of the team member's job. This constellation leads to the abatement of the earlier role for the sake of the latter due to partly conflicting expectations. When expectations are not clearly defined, team members have little control over their tasks. The technical supervisor describes the neglect as follows: *"Previously, we received tickets, and we did the most urgent task first. But honestly, many tasks were discarded as a result"*. This suggests, when work packages come in as needed without a clear structure or area of responsibility, but rather as a means to convert needs into features, the team member has only limited information, which prevents an informed decision. In addition, management considers developers as coders instead of problem solvers: *"OK, the design is more in our area of responsibility, and I would express the development mainly as coding. We coordinated the dialogues"*. While some roles are reduced to their basic tasks, other team members struggle to understand each other's role within the team. As a result, an imbalance of responsibilities and expectations occurs, and a single role becomes dominant. In this scenario, other team members are very dependent on such a dominant role, which limits their autonomy according to the technical supervisor: *"[The General Manager] was always in charge of everyone; his responsibilities are only increasing, and his time becomes limited. Now, he delegates tasks. That is fine. Thus, it cannot get out of control within his remit. However, who conserves the consistency of the 'big picture' of the product and is responsible for this? This is one burden over time. There is no product management with the capability to think holistically".* The access technology is easy and the developers are flexible to choose the technology they deem appropriate. While most development applications are open source software, new technology can be acquired when the need occurs.

Evidence suggests that interdependent goals are either informal or allocated on a different level. While there are no formal goals for the team, the technical supervisor reports that customer satisfaction serves as an informal consensus across team members: *"There is no bonus structure here. In addition, I do not receive any direct bonus either. So are there any potential conflicts within the goals or interests between individuals? I do not think that is the case. I think everyone has the objective to satisfy the customer, whether it is verbalized or not"*. While an informal common goal exists, it is not reflected within the application and the unnecessary complexity makes it difficult for the new team members to understand the software in its entirety. As a result, this complexity is not masked through a simple UI. Rather a developer indicated that large parts of the software's functionality are not used. The main information from and about users stem either from own experiences or are acquired through customer interactions during trainings and conferences. Given these infrequent and situation feedback mechanisms, the team has little incentive to act upon them.

Management support of user-related concerns is superficial. While ad hoc support from management is prominent, the technical supervisor reports that teams lack sustainable solutions, especially in times of crisis. As a result, budgets related to the concerns of users are modest and lead to a delay or deferral of investments. Management postpones the investments into user-related needs, such as usability, until another version of their product: *"Well, we invested some money at the beginning. However, rules were established at a later point in time, and as a result, no one thought about the design anymore. So, we delayed the design parts and reserved it for [software version n+1], resulting in the fact that no decision has been made in this direction for a long period of time."* For the current release, the available person-days provide a natural limit on their efforts and rely on management to prioritize tasks accordingly. The team develops more than 12 months on a release and therefore, requires the most time to develop a new release.

Within Team ALPHA, the extent of autonomy and self-organization is low. Since developers are seen as coders, general management takes most decisions. When the company used to be smaller, the General Manager was deeply involved in the development efforts of the product. At present, given the growth of the company, the General Manager's other duties have increased. At the same time, the delegation of authority and responsibilities from the General Manager do not synchronize with the company's growth. Thus, the team has limited autonomy, which challenges a mature adoption of agility. In addition, the General Manager makes many decisions alone, which often leads to an opaque decision-making process. Combined with the lacking authority, the team is not able to make informed decisions. We observe an increased dependency on the General Manager and therefore a deficient ability to develop new structures or patterns within the team. This suggests a lower extent of team agility.

Team BETA: Challenges of Complexity and the Value of User Research

Team BETA has 2.2 years of agile experience per team member on average. The technical supervisor reports that as long as roles are clearly defined and assigned, one team member can work on multiple projects with rather positive effects through the convergence and the combination of efforts by both project teams. However, in the case of an important role, such as the product manager, one team member is solely responsible for one product. Management suggests that new technology also enhances the coverage of developers' needs. While simple chat applications have been available in the past, the introduction of social platforms enriches the interactions between developers by providing additional meta-knowledge about each other.

There is no financial goal interdependence for the team, but a flexible portion of the salary relies on the overall firm performance. In addition, team members' goals are driven by an informal goal interdependence, as management expressed: *"We have feedback talks where we discuss the current performance of the team and whether everything works fine. Hence, naturally I think my success is also measured by the team assessment"*. However, the interviews suggest that all of the goals, regardless of their subject, need careful determination. Inadequate goals can have negative effects on the team's process. Management shares prior experience in which the defined goals were not challenging for the team: *"I personally had the impression that it becomes irrelevant at the end of the year. If I hadn't changed the goals, we would have delivered decently, and there is no point in it".*

Implementing complex changes depend on very specialized individuals and are therefore less autonomous. This results in a delayed implementation of such changes or generally led to time constraints due to such dependency, as management suggests when confronted with externally caused changes: *"We didn't have the time. These would have been profound structural changes, and there was no time".* Moreover, the limited redundancies that result from specialization also decrease oversight, resulting in complexity presented to the user, as described by the designer: *"There are so many field options that [the user] sees, and I think that 90% of the members or users do not need such functionality. There is also too much information for the user. Initially we had to provide this functionality. This functionality has to go somewhere. It would be nice, when we could arrange it according to some logic through which the likelihood increases that the user finds it. This is for those cases where the user does not check the manual, which happens most of the time".* According to management, increased product complexity is often a legacy, i.e., a result of the product's historicity. Often, developers and designers who implemented older features are no longer part of the team. Consequently, when such a feature requires a change, the team has to learn about the structure of such a feature again, management indicates: *"In fact, a lot of options were needed once within the past 20 years and after the fact there was a realization that the functionality was barely used. Consequently, the functionality did not evolve. However, it was still kept in the product. In this case, when the original developer left the company, no one else had any know-how about this functionality. Over time, there have been a couple of these functionalities".* Such learning efforts are often seen as unnecessary investments in old features and are stigmatized with the notion of legacy. Therefore, the team is less motivated to build redundancies for such features either due to an expected timely removal or due to an expected stability.

User research helps the team to develop an orientation that enables autonomous behavior. User research requires the means to both gather information about users, i.e., access to the user base, and store and accumulate information about the user, i.e., knowledge available about the user base. The team's user access allows the team to iterate ideas, concepts, and problems quickly with users of the software. Such a direct link avoids miscommunication that can result from introducing intermediaries. Evidence suggests that user access improves the team's ability to interact swiftly with users and extract requirements. The expected results in the form of additional feedback clarify user needs and help the team to act autonomously to develop better software. However, it is often difficult for the companies to get access to users as expressed by a developer: *"Yes, in terms of user friendliness, they will surely have many suggestions. However, often in larger organizations this is not supported, and it becomes very difficult for us to get an appointment, as those businessmen surely have other tasks, especially within the administration area".* Depending on the user base, the term "user" may lead to some irritation, and the use of roles is preferred. For example, when someone is an administrator within the customer's organization, naming him a user might be considered offensive. Using the role helps the team to understand the user's perspective and further shapes the team orientation: *"A customer survey is a great thing. For us, the customer is the administrator who decides that his company needs such a product. From our point of view, the administrator is the customer, as he orders the product for his firm, needs them, and in fact he is not the end-user who actually works with the product. Thus, it is not the end-customer who actually works with the product".* Besides good user access, the team should also encompass sound knowledge about its user base. The sheer existence of user access does not imply a good under-

standing of the user's tasks and context. This knowledge will help to enhance the lean aspect of agility, streamlining activities and limiting them to those that drive the perceived user value and therefore increase the team's orientation. A negative effect of little or no user knowledge is extra development efforts. Such efforts are caused by the team's disorientation and result in a mismatch of user needs with the designed solution as expressed by a technical supervisor: *"Absolutely, this often occurs in a painful way, probably with your first customer, if the requirements were not exact but loosely defined and not questioned afterwards. So you give [the software] to the customer, and then you realize, 'Oh, the technological landscape is different from ours.' There is an additional layer, and [the software] does not fit their environment and will not work".*

The initial interest and willingness of management to address user-related concerns quickly fade away when the team requires actions or the allocation of budgets, as mentioned by a technical supervisor: *"[Top management] is interested in improvements that are also communicated as the need to become more user-friendly. The fact that this also requires the availability of resources is not necessarily seen and supported"*. The team appreciates receiving initial support; however, when funding is required, progress slows down. Therefore, the BETA case builds on current planning activities by putting more emphasis on user-related concerns for their future development efforts. The team's release cycle is aligned to the annual release cycle of changing formal requirements related to financial transactions.

We observe a higher extent of autonomy and self-organization in Team BETA. After the team had made the decision to adopt ASD, the technical supervisor reports experienced difficulties and challenges that follow the introduction of a new paradigm. However, the supervisor has reported positive effects experienced during the past year. While the team presents a higher extent of agility because of increased autonomy and self-organization, challenges remain. Management reports that there are still some traditional views, e.g., starting with the development of the back-end and continuing with the front-end. Yet, techniques such as prototyping and sketches that work well within agile environments suggest a different order. They identify and acknowledge the problem. Management is committed to addressing these challenges in the future. While they have already taken the first steps in this direction, more time will have to pass before the team has truly incorporated the idea.

Team GAMMA – Open-minded Management and Short Development Cycles

Team GAMMA's team members have the most experience with ASD (3.75 years on average). The team members in Team GAMMA are less dependent on each other and are rather autonomous. The team defines roles clearly and establishes a contractor/client relationship within the organization. As a result, team members clearly assign themselves to one of the two perspectives, hence increasing role clarity for that member. Ensuring this is a management responsibility: *"We are considering this as a management task as we want to establish a principal agent situation, where some people are responsible for ensuring the attainment of customer requirements and economic success. Taken as a whole, it is the separation of the product owner and the development team".* While the assignment of two projects to one team member is beneficial, the assignment of two roles to one team member results in a conflict. In the latter scenario, the team member's performance and contribution decrease, as explained by a developer: *"I assumed a twofold role at this point, 50% | 50%, based on restricted thinking. This means that it is not possible*

without one [scrum master], and [interests] collide. We tried it, but I noticed that I could not fulfil the scrum master role. It doesn't work and is not justified for the team and leads to worse results". The designer positively mentioned that a clear separation of tasks helps the team members to understand their contribution and the contribution of each member with whom they are interacting. In addition, this meta-knowledge helps the team members to enhance efficiency, as they know when to contact a colleague for further information in order to make an informed decision.

While there are no team related financial incentives, the organization measures its organizational success. When goals are achieved, employees receive a reward. Thus, team members and the team as a whole work thrive to achieve success to which they have only limited influence. The team members report the application to be highly complex in the backend, yet they seek to provide a simple and easy to use interface. The team reports that they have a good and continuous access to the user. The access to the user can vary with the team member's role, but regular workshops, occasional on-site visits and accessibility through phone allows the team to build substantial knowledge of their user base. This is important, as the technical supervisor realizes that both, the benefit of user knowledge and obstructions or the lack thereof is only visible after the fact: *"[Usability] was not a major topic in the old product version. We thought we had understood the customer. Only after the fact do you even realize that you did not. In the new product that we are currently working on, we used prior prototypes and interviews with recordings to consider the whole task and in the work environment."*

Instead, management focuses on the core workflow with clear and visible benefits to the team's operations and its organization, as mentioned by management and designer: *"It is a question whether I integrate [usability] into an already existing [software] and how I can integrate it into a grown software with the budget that the customer has already paid for. I can certainly change smaller things, but I cannot change the basic workflow again. The other thing is that, when you start doing something really new, management is pretty engaging, so then it has to be delivered".* Such prioritization helps the team to understand management's expectations, and therefore the team can adapt appropriately. Knowing the internal budgetary limitations, the team can find an alternative approach to fund such initiatives and efforts through specific customers. Management suggests: *"There are budget restrictions where I say that I won't invest a few hundreds of thousands [of euros] in the current product line so that the customer gets along better. For that, he should pay money for the next product line, which has great usability".* The team has the most experience with agile software development and the shortest release cycle. The team releases a new version every six months. Team GAMMA also follows the scrum method.

The team mentioned the benefits of accessing users directly. The feedback from users enables the team to align the product vision with the users' needs. Hence, the technical supervisor reported that they consider such early feedback to be a valuable source of information: *"This approach was beneficial as we did not say what the customers had to do, but we developed a vision that was subsequently adjusted in collaboration with the customers to meet the customers' expectations exactly".* Such collaboration with the user is a result of their high extent of autonomy and their ability to self-organize. The team's high extent of autonomy leaves them to self-organize according to the feedback from the users. Such feedback allows the team to enhance their understanding of the perceived user value and therefore increase their extent of agility. In the interviews, we learned that the value of

such feedback is different in each team. As Team GAMMA developed a new application, the technical supervisor pointed out that they had the chance to be more open to new ideas as part of their development process. Overall, we find a relationship between the self-organization of the team and team agility.