2014 47th Hawaii International Conference on System Science

Coordination in Large-Scale Agile Software Development: A Multiteam Systems Perspective

Alexander Scheerer
SAP AG &
Institute for Enterprise Systems,
University of Mannheim
alexander.scheerer@sap.com

Tobias Hildenbrand
SAP AG
tobias.hildenbrand@sap.com

Thomas Kude
University of Mannheim
kude@uni-mannheim.de

Abstract

The widespread use of lean and agile development methods shows a fundamental shift in how organizations try to cope with complexity and volatility issues. In large-scale settings, the coordination of many people often results in a team of teams setup. We introduce the multiteam systems perspective to describe different conceptual strategy types for inter-team coordination. These types are illustrated with examples from a large enterprise software development organization.

1. Introduction

Lean and agile development methods have become widespread in use and are the de facto standard in large parts of many software organizations of different sizes [7,18,31,41,52,65,66]. The introduction of these approaches shows a fundamental shift in how organizations try to cope with complexity and volatility issues [13].

Previous development processes have tried to cope with these problems by "risk minimization" measures in the form of large upfront planning and rigid stage-gated process steps and structures. This led to inflexible requirements management, long time-to-market and a fear of delivery, as markets evolved in the months from project start [29].

The introduction of agile development and lean principles over the last decade [6,45,46] have shifted coping strategies for complexity and volatility towards more collaborative and cooperative approaches [19] with empirical process controls [55]. Self-empowered teams are one of the main changes regarding this issue. Planning Poker, Pair Programming, User Story Mapping [44] and other cooperative approaches in the development process are only a few examples of the methodological shift. Many of these new approaches have been regarded in light of small company or single team settings, or with student developer teams. However, these development methods have gained prominence in large-scale settings as well. These contexts show particular challenges as large groups of people need to be coordinated, which usually results in a hierarchical team of teams setup [24] where several teams have to work closely together in order to release a single software product. This organizational setup has been defined as a multiteam system (MTS) by Mathieu et al. [36] who assert that MTSs are "two or more teams that interface directly and interdependently in response to environmental contingencies toward the accomplishment of collective goals" [36:290].

As the beginnings of Agile Development lie in small team contexts, the available literature on inter-team coordination in large-scale setups is sparse. The main inter-team coordination mechanism in these types of development environments is, according to practitioner literature, the Scrum-of-Scrums approach [25,56]. Previous publications on this topic remain scarce, only seven papers could be identified by the authors [5,26,42,43,57,60,61] that come to the conclusion that coordination on an inter-team level remains extremely challenging [42].

Previous studies found that the theoretical understanding in the field of agile development is lacking and have called for more studies on the underlying fundamental concepts of agile software development [1,2,13]. With this work, we intend to advance the conceptual understanding of coordination in multiteam agile software development systems and try to answer the research question:

How can coordination theoretically be achieved in large-scale software development systems?

In order to do so, this paper is structured as follows. Section 2 gives an overview of the literature streams which need to be incorporated for a comprehensive view on the above research question. Section 3 derives possible coordination archetypes within MTS contexts in agile IS development and gives an illustrative example for two coordination types. Section 4

978-1-4799-2504-9/14 $31.00 © 2014 IEEE
DOI 10.1109/HICSS.2014.587

concludes this paper with implications and future work.

2. Foundations

Starting with the environment in which large-scale development finds itself, we briefly show how complex adaptive systems theory and Agile Development fit together. Subsequently, we introduce the concept of multiteam systems from organizational psychology and finish with the conceptual fundamentals of coordination.

2.1. Large-Scale Software Development

The previously dominant scientific management [62] oriented methods and the move towards more cooperative lean and agile development methods signify a shift in coping strategies for complexity and volatility. The full scope of the problem space where software development finds itself was not captured by previous approaches. Incomplete and ever-changing requirements together with complex interdependencies in the requirements as well as the existing software stack are only a few attributes of this problem domain. First steps in describing this problem context have been made by DeGrace and Stahl [12] in referring to Software Development as wicked problems [50]. More recently the Cynefin Framework (see Figure 1) [58] has laid out the relationship between work contexts and possible solution approaches based on, among others, complex adaptive systems theory [20].

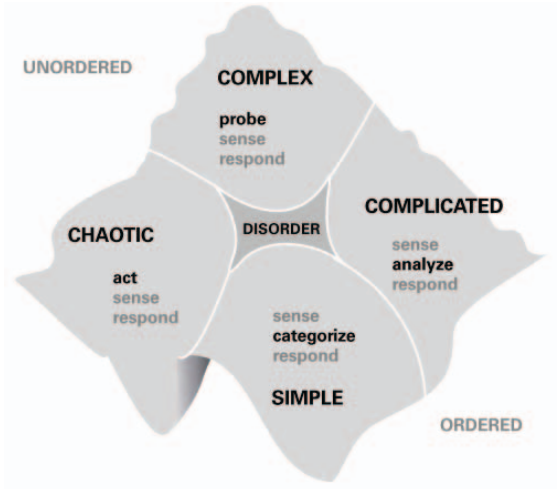

Figure 1. The Cynefin Framework [58]

The framework provides five contexts which are defined by their relationship between cause and effect. The simple, complicated, complex, and chaotic domains necessitate contextually appropriate responses, while disorder is prevalent when it is uncertain which other context seems dominant.[58]

Within the framework the move towards agile and lean development can be viewed as a shift from a complicated environment to understanding software development as a complex environment [49]. The leadership paradigm of probe, sense, respond can clearly be seen in agile's quest for shorter intervals with working software to validate customer requirements early. According to the Cynefin Framework, complex domains require more interaction and communication of actors than any other, which shows the clear push towards more collaborative/cooperative work as mentioned earlier [58].

2.2. Multiteam Systems Research

The type of large-scale development system examined is one in which several teams have to work together in order to complete a release of a software product. This type of organizational setup has been described within the organizational psychology domain as a multiteam system (MTS). The collective goal of this system can be broken down into a goal hierarchy and constitutes a key characteristic of any MTS. The goal hierarchy marks the boundary of a MTS in that all teams within the system share at least a distal goal while the individual teams pursue their more proximal goals individually. This structure of goals leads to teams displaying input, process and outcome interdependence with at least one other team [36].

While MTS have received increasing attention in organizational psychology over the last decade [3,10,11,23,33,34,37], the aspect of coordination is underdeveloped. So far, especially the areas of compositional attributes and linkages have been explored. Marks et al. [33] found that cross-team processes had the most value in MTSs with a high interdependent goal hierarchy. Well-managed MTS transition processes influenced MTS performance positively, but did not support team-level action processes. Decentralized planning led to enhanced multiteam performance by fostering proactivity and higher aspiration levels. Nevertheless, strong negative effects were found in excessive risk seeking and coordination failures [23].

Asencio et al. [3] propose multiteam charters as a means to develop efficient leadership structures and communication networks. Boundary spanners and communication norms across teams are mentioned as important considerations in MTS collaboration. These differentiated team roles are viewed by Davison et al. [10] as a key factor in performance. Teams which included boundary spanning roles consistently

outperformed teams which had not. The reasoning lies in the information processing complexity inherent in large organizations which lead to the need for formalized boundary spanning [10].

In their study of leadership in multiteam systems DeChurch and Marks [11] trained leader teams in two ways, either by facilitating strategy development or coordination. They found that strategy training was positively related to explicit coordination, with coordination training affecting implicit coordination (see section 2.3) stronger. However, an unidentified mechanism, such as shared mental models, seem to be influencing inter-team coordination. They conclude that the study of mental models in MTSs constitutes an interesting path for future research [11].

2.3. Coordination Research

Coordination is a multi-faceted research area which takes its inputs from a variety of fields including but not limited to Economics, Organization Theory and Computer Science. Coordination Theory presents a framework for analysis of coordination in that it defines coordination as the management of dependencies. These dependencies are to be managed by coordination mechanisms [30]. However, no predictive power arises from this theory as no hypotheses or propositions are stated [59]. Crowston et al. [9] recognizes these limitations and calls for future research to develop testable hypotheses.

The study of coordination in organizational theory has identified several mechanisms to coordinate workers [32,38,63,64]. Thompson [63] who cites March and Simon [32] in his description of three key generic coordination approaches: standardization or rules, plans and schedules, and mutual adjustment. Van De Ven et al. [64] added a fourth dimension of team, which extends mutual adjustment by joint simultaneous interactions within a usually collocated team. Similarly, Mintzberg [38] proposes mutual adjustment, direct supervision, and standardization (of work processes, of work outputs, of norms and of worker skills) as basic mechanisms for coordination.

The mix of mechanisms according to situational context factors is of interest when regarding coordination strategies [28,59]. Strode et al. [59] present first insights into the combination of coordination mechanisms in agile development. They present a coordination strategy which includes synchronization, structure and boundary spanning as key elements which influence coordination effectiveness [59]. Furthermore, intensified communication was observed as a facilitator of mutual trust and shared cognition by Li and Maedche [28].

In an effort to classify coordination mechanisms, Espinosa et al. [14] present three types of coordination: mechanistic, organic and cognitive coordination. While mechanistic coordination includes coordination by plan or rules with little communication, organic coordination refers to coordination by means of mutual adjustment or feedback via interaction. This communication can be formal and planned or informal and spontaneous. Cognitive coordination, on the other hand, is based on knowledge the actors have about each other and is achieved implicitly. Shared mental models [8,15] and transactive memory systems [39] are two examples of this type of mechanism. Cognitive Coordination is viewed by Espinosa et al. as a key enhancer of mechanistic and organic coordination [14].

This position is also supported by Rentsch and Staniewicz [48], who emphasize the role of cognitive coordination by suggesting cognitive similarity as the key driving component of coordination mechanisms in multiteam systems. These cognitive similarity types are identified by the form of similarity, the form of cognition, and the cognitive content domain.

The form of similarity among individuals includes congruence, accuracy and complementarity. The emergence of cognitive congruence among teams constitutes the matching of cognitions between individual team members when there is no correct value or target. Accuracy reflects exactly this degree of fit towards a predetermined target value. Complementarity, on the other hand, represents the differing cognitions of team members that fit together like puzzle pieces [48]. These types of similarities have been found to positively influence team performance, e.g. via shared mental models [8,15,16,27,35,51], as an example of a congruent cognition, or through transactive memory systems [17,21,22,39,40], which depicts a complimentary cognition.

Lastly, the form of cognition pertains to the type of information in consideration, with the cognitive content domain referring to the knowledge areas which are dealt with [47].

Our conceptual framework (Figure 2) shows the process theoretical view of inter-team coordination effectiveness and especially the interplay of mechanistic, organic and cognitive coordination mechanisms within the coordination strategy.

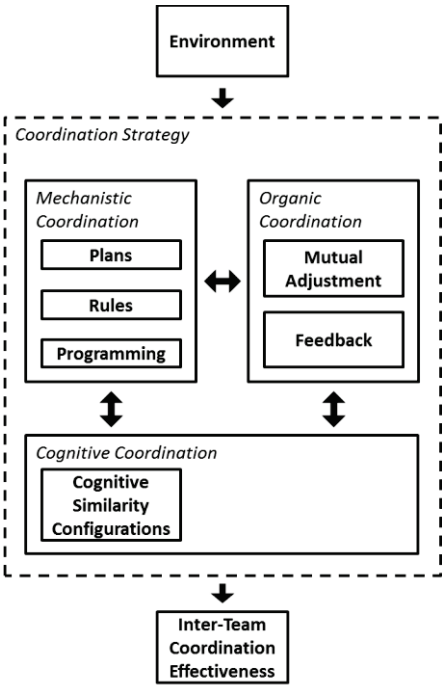

Figure 2. Conceptual Framework
Adapted from [14,16]

3. Archetypes of Coordination in MTS

According to our previously introduced coordination framework of mechanistic, organic and cognitive coordination, we introduce conceptual archetypes of coordination modes within MTSs. Table 1 shows all possible strategy types when considering only low and high degrees within the three coordination types. This reduced view was chosen in order to keep the overview simple. Further subtypes mentioned in Section 3.1 can be seen in Table 2.

Table 1. Strategy Types

Strategy Type	Coordination type			plausible
	Mechanistic	Organic	Cognitive	
1	high	low	low	✓
2	low	high	high	✓
3	low	low	high	(✓)
4	high	high	high	(✓)
5	low	low	low	(✓)
6	low	high	low	-
7	high	high	low	-
8	high	low	high	-

3.1. Coordination Strategy Types

Out of the identified possible configurations, only some appear plausible, as will be discussed next. Hence we first discuss in the following section plausible types 1, 2 and possible subtypes of 2 which differ in their combination of mechanistic and cognitive coordination but remain high in their organic coordination. After that, we will continue with the plausible types 3, 4 and 5 which have significant drawbacks in practice and finish with the implausible types 6, 7 and 8.

Strategy type 1 ("The Perfect Plan") is characterized by high mechanistic, low organic and low cognitive coordination. This archetype depicts the perfect plan-based approach to software development. While within teams, coordination may well be achieved through organic or cognitive mechanisms, the focus of multiteam coordination in this strategy lies solely on mechanistic coordination with little communication between individual actors. This type assumes that software development can be "programmed" from a coordination perspective, e.g. through complete upfront planning all dependencies as well as all contingencies can be resolved and accounted for. Since the coordination is programmed through upfront planning with little communication, one person or a very small set of people, needs to have a deep insight into the full technical details of the entire software system in order to specify all details necessary for individual work packages and correct integration. While this type, taking aforementioned assumptions into account, is entirely plausible from a theoretic viewpoint, it has deep implications for large-scale software development, especially in the enterprise software domain where requirements are often in a state of flux. In an environment where large existing codebases have complex interdependencies within and to other software modules and products, the assumption of an omniscient person or small group who has the capability to plan an entire software release down to individual work packages with their respective technical dependencies seems elusive. This type can be illustrated with the previously introduced complicated domain of the Cynefin Framework (see Chapter 2.1). The leadership paradigm of sense, analyze and respond is the core of strategy type 1. First, the problem space of the software is understood (sensed). Typically a small set of people (including the chief architect for example), will analyze the problem and develop a technical plan (architecture) and a sequence plan (schedule). The enactment of both plans, e.g. the actual development phase, is then considered the response in this case.

Strategy type 2 can be viewed as the antipode to type 1. Through the low to medium occurrence of mechanistic coordination, this strategy relies on organic and cognitive mechanisms in order to achieve coordination effectiveness. This type includes three subtypes which differ in their amount of mechanistic and cognitive mechanisms but remain high on organic coordination throughout.

Table 2. Strategy Type 2 Subtypes

Strategy Type	Coordination type			plausible
	Mechanistic	Organic	Cognitive	
2.1	medium	high	high	✓
2.2	very low	high	high	✓
2.3	low-medium	high	medium	✓

Strategy Type 2.1 ("Organic Planning") relies on high organic and cognitive coordination with medium mechanistic coordination. An assumption inherent in this type is that there are limits to the planning capability of individuals and small groups. Nevertheless, a certain amount of planning is seen as required in order to achieve a general alignment between individual teams and to reduce unnecessary rework and communication overhead introduced by organic coordination.

Strategy Type 2.2 ("Communication Focused") shows little to no mechanistic and a maximum amount of organic and cognitive coordination. The lack of planning and other mechanistic coordination is based on the assumption that any type of investment in plans and rules goes to waste as they will need to be adjusted continuously anyway. In order to achieve coordination effectiveness teams need to communicate comprehensively and adjust their actions mutually which relies heavily on feedback and a common understanding or shared knowledge base, thus questioning the initial subdivision into a team of teams setting.

Strategy Type 2.3 ("Selective Communication") is characterized by low-medium mechanistic, high organic and medium cognitive coordination. Inherent in this type of coordination is the notion that organic coordination has limits to the amount of people or teams that can be effectively coordinated. In order to lower the necessary shared understanding among participants and thus increase organic coordination effectiveness, only subparts of the development system engage in "costly" coordination activities and system wide coordination is established via hierarchical procedures where the amount of involved people is reduced the more subparts or teams are involved.

Overall, the lower reliance on upfront planning and rigid rules allows for a leadership paradigm comparable to the complex domain in the Cynefin Framework of probe, sense and respond. Fitting to the lean principle of customer value, it is possible to quickly experiment with requirements and present prototypes to the customer for idea validation and respond to the feedback given.

The following types remain plausible but have significant disadvantages in a practical setting.

Strategy type 3 ("Coordination Heaven") A hypothetical type, in which only cognitive coordination is high. This cannot be achieved in reality as one would have to employ people that already have a high cognitive coordination without using the other types, e.g. by developing the exact same piece of software with the same people. As cognitive coordination needs to be established between the actors somehow, the prohibited use of mechanistic and organic coordination also prevents the establishment of cognitive coordination.

Strategy type 4 ("Extensive Coordination") in which every coordination type is high will most likely be very well coordinated. However, coordination is not an end in itself. Its right to exist is dependent on the actual tasks and work to be done. To deliberately implement this coordination strategy would mean to accept high overhead costs for coordination with unclear benefits in comparison to other strategies.

Strategy type 5 ("Coordination Deficit") shows little coordination activities in any of the three coordination types. While theoretically plausible and perhaps also practically present, the absence of coordination activities in a discussion on coordination strategies seems unrewarding and can hardly lead to coordination effectiveness.

The following three types are implausible from a conceptual standpoint.

Strategy type 6 shows low mechanistic, high organic and low cognitive coordination. If only implementing organic coordination activities without some sort of common understanding or knowledge sharing to base communication on, this strategy promotes aimless communication and feedback. While not plausible as a desirable strategy to achieve coordination effectiveness, this coordination strategy might depict an intermediate state for attaining strategy type 2.2 or 2.3.

Strategy type 7 promotes high mechanistic and high organic coordination. If we think back to the definitions of both coordination types, we see that mechanistic relies heavily on plans and rules with little or no communication, while organic coordination is based upon communication. This is a direct

contradiction and implausible from a theoretic standpoint.

Strategy type 8, with high mechanistic, low organic and high cognitive coordination is equally implausible as high cognitive coordination (e.g. shared understanding) among the teams is mainly established through communication which is not desirable here, as the amount of organic coordination is low.

3.2. Illustrative Examples

Two illustrative examples from a large-scale development organization will be used to demonstrate the previously discussed archetypes of coordination in multiteam systems.

3.2.1 Example for strategy type 1. The software company in our example followed this strategy for a long time before introducing lean management and agile development in 2009. As the company grew massively since the 1970s, a process framework needed to be in place in order to (a) coordinate development projects with 10.000 and more person days and (b) establish a quality management system to achieve an ISO certification. This process framework followed a typical "waterfall pattern" when it was introduced in the mid-1990s, i.e. there was a phase with basic portfolio decisions, then requirements were specified, the architecture was designed and refined and then development across multiple teams started. When the software was put into code, testing started, often by a central testing department located offshore as well as with consultants and customers. The tested software was validated by another central team that adopted the role of first customer, i.e. they installed, configured and re-tested the software in their lab. Only after this sequential process of about 12 to 18 months the software released to market – a market that often changed substantially in the meantime. [54]

3.2.2 Examples for strategy type 2. Working according to strategy type 1 led to numerous problems as the company grew further and the product portfolio became even broader. Among other things, a high degree of bureaucracy emerged due to the division of labor along the process. Moreover, it took too long until software could be evaluated with customers to get feedback and minimize planning risks. Therefore, starting in 2009 after a longer pilot phase, the company implemented lean management and agile development [53]. One major change was the introduction of Scrum as process framework on the team level. Along with Scrum there was no longer a "project manager" but the roles of "Scrum Master", responsible for the team process, and "Product Owner", responsible for the product towards customers.

As most enterprise applications are built by more than one single team, inter-team coordination and scaling was a key issue. Therefore, the role of "Chief Product Owner" (CPO) was established. The CPO is responsible for an entire application or solution. Depending on the size of the product, there are 1-2 levels of (area) product owners, i.e. the overall product is divided into several "product areas" and then feature sets on team level. Within these teams, architects, developers, user interface designers, documentation writers, etc. are included to implement features end-to-end. In this setting the coordination follows a mixture of type 2.1 and 2.3 depending on the respective product development area. Some departments tend to lean more towards a centrally focused coordination, e.g. CPO, APOs and POs are responsible for inter-team coordination in a "product team" who then carry decisions into the individual teams. Other departments implemented a Scrum hierarchy, e.g. Scrum-of-Scrums in which delegates from the teams manage coordination tasks together with the APOs and POs. The strategy 2.2 ("Communication Focused") is only seen in areas of high exploratory nature. Here teams decide to participate in "open spaces", meetings which have little predefined structure and purely serve the purpose of increasing cognitive coordination and thus promote organic coordination.

4. Discussion and Future Research

The presented archetypes of coordination in MTS constitute a first step to establish the multiteam level of analysis in studies of software development organizations. The described coordination strategies lie on a continuum between organic and mechanistic coordination types. On the far side of pure organic/cognitive coordination (see strategy type 2.2), one has to recognize the absurdity of dividing the system in individual teams if the communication network is completely interconnected between teams. This goes against the initial reasoning to divide the systems into teams, e.g. to reduce overhead communication and coordination through establishing modularity [4]. On the other side of the spectrum, the purely mechanistic strategy (see strategy type 1), contradicts the lean and agile principles of empowered teams and embracing change. The problems addressed by lean/agile methods in small settings are only lifted to the inter-team level in large-scale settings.

The described trade-off can only be managed by a balanced approach to coordination within MTSs. One possible tactic is to institutionalize cognitive coordination via boundary spanners. First insights into

this strategy have been published by Davison et al. [10], albeit in a military setting. The differences in the environment, e.g. task types or dependencies between teams, associated with software development are sufficient to necessitate an exploratory approach for this research area.

Based on the understanding gathered from these conceptual insights, we see a high value in pursuing an exploratory approach to shed light on coordination in large-scale software development. The exploration of coordination could be based on our conceptual framework (Figure 2), in which the core part depicts the reciprocal influence of mechanistic, organic and cognitive coordination within the coordination strategy. A focus on the link between organic and cognitive coordination seems promising, as cognitive similarities are likely to influence mutual adjustment and feedback processes. In addition, the impact of the environment, e.g. multiteam characteristics [67], may affect the entire coordination system. We deem the multiteam system level of analysis appropriate, in order to make sense of the multi-level data to be gathered, as cognition rests exclusively within individuals [47] while cognitive similarity configurations are a multiteam emergent state which in turn drives a multiteam process, namely coordination [48].

5. References

[1] Abrahamsson, P., Conboy, K., and Wang, X. 'Lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems 18*, 4 (2009), 281–284.

[2] Ågerfalk, P.J., Fitzgerald, B., and Slaughter, S. Flexible and distributed information systems development: State of the art and research challenges. *Information systems research 20*, 3 (2009), 317–328.

[3] Asencio, R., Carter, D.R., DeChurch, L.A., Zaccaro, S.J., and Fiore, S.M. Charting a course for collaboration: a multiteam perspective. *Translational Behavioral Medicine 2*, 2 (2012).

[4] Baldwin, C.Y. and Clark, K.B. *Design rules: The power of modularity*. The MIT Press, 2000.

[5] Bannerman, P.L., Hossain, E., and Jeffery, R. Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage? *System Science (HICSS), 2012 45th Hawaii International Conference on*, (2012), 5309–5318.

[6] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2001.

[7] Begel, A. and Nagappan, N. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. *International Symposium on Empirical Software Engineering and Measurement*, IEEE Computer Society (2007), 255–264.

[8] Cannon-Bowers, J.A., Salas, E., and Converse, S. Shared mental models in expert team decision making. *Current issues in individual and group decision making*, (1993), 221–246.

[9] Crowston, K., Rubleske, J., and Howison, J. Coordination theory: A ten-year retrospective. In P. Zhang and D. Galletta, eds., *Human-Computer Interaction in Management Information Systems*. M. E. Sharpe, Inc., 2006, 120–138.

[10] Davison, R.B., Hollenbeck, J.R., Barnes, C.M., Sleesman, D.J., and Ilgen, D.R. Coordinated Action in Multiteam Systems. *The Journal of applied psychology 97*, 4 (2012), 808–24.

[11] DeChurch, L.A. and Marks, M.A. Leadership in Multiteam Systems. *The Journal of applied psychology 91*, 2 (2006), 311–29.

[12] DeGrace, P. and Stahl, L.H. *Wicked Problems, Righteous Solutions*. Yourdon Press, Upper Saddle River, NJ, USA, 1990.

[13] Dybå, T. and Dingsøyr, T. Empirical studies of agile software development: A systematic review. *Information and Software Technology 50*, 9-10 (2008), 833–859.

[14] Espinosa, J.A., Armour, F., and Boh, W.F. Coordination in enterprise architecting: an interview study. *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, (2010), 1–10.

[15] Espinosa, J.A., Kraut, R., Lerch, J., Slaughter, S., Herbsleb, J., and Mockus, A. Shared mental models and coordination in large-scale, distributed software development. *Proceedings of the International Conference on Information Systems (ICIS 2001)*, (2001), 64.

[16] Espinosa, J.A., Lerch, F.J., Kraut, R.E., Salas, E., and Fiore, S.M. Explicit vs. implicit coordination mechanisms and task dependencies: one size does not fit all. In *Team cognition: understanding the factors that drive process and performance.* American Psychological Association, Washington, DC, 2004, 107–129.

[17] Faraj, S. and Sproull, L. Coordinating Expertise in Software Development Teams. *Management Science 46*, 12 (2000), 1554–1568.

[18] Fry, C. and Greene, S. Large Scale Agile Transformation in an On-Demand World. *Proceedings of the AGILE Conference 2007*, (2007), pp. 136 – 142.

[19] Hildenbrand, T., Rothlauf, F., Geisser, M., Heinzl, A., and Kude, T. Approaches to collaborative software development. *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, (2008), 523–528.

[20] Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[21] Hollingshead, A.B. Retrieval processes in transactive memory systems. *Journal of Personality and Social Psychology 74*, 3 (1998), 659.

[22] Kanawattanachai, P. and Yoo, Y. The impact of knowledge coordination on virtual team performance over time. *MIS quarterly 31*, 4 (2007), 783–808.

[23] Lanaj, K., Hollenbeck, J.R., Ilgen, D.R., Barnes, C.M., and Harmon, S.J. The Double-Edged Sword of Decentralized Planning in Multiteam Systems. *Academy of Management Journal of Management Journal*, (2012), 1–61.

[24] Larman, C. and Vodde, B. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley Professional, Upper Saddle River, N.J, 2008.

[25] Larman, C. and Vodde, B. *Practices for Scaling Lean and Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley Professional, Upper Saddle River, N.J, 2010.

[26] Lee, E.C. Forming to performing: Transitioning large-scale project into agile. *Agile, 2008. AGILE'08. Conference*, (2008), 106–111.

[27] Levesque, L.L., Wilson, J.M., and Wholey, D.R. Cognitive divergence and shared mental models in software development project teams. *Journal of Organizational Behavior 22*, 2 (2001), 135–144.

[28] Li, Y. and Maedche, A. Formulating Effective Coordination Strategies in Agile Global Software Development Teams. *Proceedings of the International Conference on Information Systems (ICIS 2012)*, (2012).

[29] Mackert, O., Hildenbrand, T., and Podbicanin, A. Vom Projekt zum Produkt - SAP's Weg zum "Lean Software Product Development". *Vom Projekt zum Produkt. Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik (WI-MAW), 01.-03. Dezember 2010 in Aachen*, (2010), 13–25.

[30] Malone, T.W. and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys 26*, 1 (1994), 87–119.

[31] Mangalaraj, G., Mahapatra, R., and Nerur, S.P. Acceptance of software process innovations – the case of extreme programming. *European Journal of Information Systems 18*, 4 (2009), 344–354.

[32] March, J.G. and Simon, H.A. *Organizations*. Wiley, New York, 1958.

[33] Marks, M.A., DeChurch, L.A., Mathieu, J.E., Panzer, F.J., and Alonso, A. Teamwork in Multiteam Systems. *The Journal of applied psychology 90*, 5 (2005), 964–71.

[34] Mathieu, J.E., Gilson, L.L., and Ruddy, T.M. Empowerment and Team Effectiveness: An Empirical Test of an Integrated Model. *The Journal of applied psychology 91*, 1 (2006), 97–108.

[35] Mathieu, J.E., Heffner, T.S., Goodwin, G.F., Salas, E., and Cannon-Bowers, J.A. The influence of shared mental models on team process and performance. *Journal of Applied Psychology 85*, 2 (2000), 273–283.

[36] Mathieu, J.E., Marks, M.A., and Zaccaro, S.J. Multiteam systems. In N. Anderson, D.S. Ones, H.K. Sinangil and C. Viswesvaran, eds., *Handbook of Industrial, Work and Organizational Psychology Volume 2 Organizational Psychology*. Sage Publications Ltd, London, 2001, 289–313.

[37] Mathieu, J.E., Maynard, M.T., Taylor, S.R., Gilson, L.L., and Ruddy, T.M. An examination of the effects of organizational district and team contexts on team processes and performance: a meso-mediational model. *Journal of Organizational Behavior 28*, (2007), 891–910.

[38] Mintzberg, H. Structure in 5's: A Synthesis of the Research on Organization Design. *Management science 26*, 3 (1980), 322–341.

[39] Moreland, R., Argote, L., and Krishnan, R. Socially shared cognition at work: Transactive memory and group performance. In *What's social about social cognition? Research on socially shared cognition in small groups*. Sage Publications, Inc, 1996, 57 – 84.

[40] Moreland, R.L. and Myaskovsky, L. Exploring the performance benefits of group training: Transactive memory or improved communication? *Organizational Behavior and Human Decision Processes 82*, 1 (2000), 117–133.

[41] Nerur, S., Mahapatra, R.K., and Mangalaraj, G. Challenges of migrating to agile methodologies. *Communications of the ACM 48*, 5 (2005), 72–78.

[42] Paasivaara, M., Lassenius, C., and Heikkila, V.T. Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work? *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*, (2012), 235–238.

[43] Paasivaara, M. and Lassenius, C. Scaling scrum in a large distributed project. *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, (2011), 363–367.

[44] Patton, J. User Story Mapping. 2008. http://guide.agilealliance.org/guide/storymap.html.

[45] Poppendieck, M. and Poppendieck, T. *Lean software development: an agile toolkit*. Addison-Wesley Professional, 2003.

[46] Reinertsen, D.G. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

[47] Rentsch, J.R., Delise, L.A., and Hutchison, S. Cognitive Similarity Configurations in Teams: In Search of the Team MindMeld. In E. Salas, G.F. Goodwin and C.S. Burke, eds., *Team Effectiveness in Complex Organizations: Cross-Disciplinary Perspectives and Approaches*. Psychology Press, New York, NY, 2009, 241.

[48] Rentsch, J.R. and Staniewicz, M.J. Cognitive similarity configurations in multiteam systems. In *Multiteam systems : an organization form for dynamic and complex environments*. Routledge, New York, NY, 2012, 225–252.

[49] Rikkila, J., Abrahamsson, P., and Wang, X. The Implications of a Complexity Perspective for Software Engineering Practice and Research. *Journal of Computer Engineering & Information Technology*, (2012).

[50] Rittel, H.W.J. and Webber, M.M. Dilemmas in a General Theory of Planning. *Policy Sciences 4*, 2 (1973), pp. 155–169.

[51] Rouse, W.B., Cannon-Bowers, J.A., and Salas, E. The role of mental models in team performance in complex systems. *Systems, Man and Cybernetics, IEEE Transactions on 22*, 6 (1992), 1296–1308.

[52] Rudorfer, A., Stenzel, T., and Herold, G. Case for Feature-Oriented Requirements Engineering. *IEEE Software 29*, 5 (2012), 54–59.

[53] Scheerer, A., Schmidt, C.T., Heinzl, A., Hildenbrand, T., and Voelz, D. Agile Software Engineering Techniques: The Missing Link in Large Scale Lean Product Development. *Proceedings of the "Multikonferenz Software Engineering"*, (2013).

[54] Schnitter, J. and Mackert, O. Introducing Agile Software Development At SAP AG - Change Procedures and Observations in a Global Software Company. *Proceedings of the 5th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, (2010).

[55] Schwaber, K. and Beedle, M. *Agile Software Development with Scrum*. Prentice Hall, 2001.

[56] Schwaber, K. *Agile project management with Scrum*. Microsoft Press, 2004.

[57] Smits, H. and Pshigoda, G. Implementing scrum in a distributed software development organization. *Agile Conference (AGILE), 2007*, (2007), 371–375.

[58] Snowden, D.J. and Boone, M.E. A leader's framework for decision making. *harvard business review 85*, 11 (2007), 68.

[59] Strode, D.E., Huff, S.L., Hope, B., and Link, S. Coordination in co-located agile software development projects. *Journal of Systems and Software 85*, 6 (2012), 1222–1238.

[60] Sutherland, J., Schoonheim, G., and Rijk, M. Fully distributed scrum: Replicating local productivity and quality with offshore teams. *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, (2009), 1–8.

[61] Sutherland, J., Schoonheim, G., Rustenburg, E., and Rijk, M. Fully distributed scrum: The secret sauce for hyperproductive offshored development teams. *Agile, 2008. AGILE'08. Conference*, (2008), 339–344.

[62] Taylor, F.W. *The principles of scientific management*. Harper & Brothers, New York, London, 1911.

[63] Thompson, J.D. *Organizations in Action: Social Science Bases of Administrative Theory*. McGraw-Hill, 1967.

[64] Van De Ven, A.H., Delbecq, A.L., and Koenig Richard, J. Determinants of Coordination Modes within Organizations. *American Sociological Review 41*, 2 (1976), 322–338.

[65] VersionOne Inc. 7th Annual State of Agile Development Survey. 2012. http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf.

[66] West, D., Grant, T., Gerush, M., and D'Silva, D. Agile Development: Mainstream Adoption Has Changed Agility. *Forrester Research*, (2010).

[67] Zaccaro, S.J., Marks, M.A., and DeChurch, L.A. Multiteam Systems: An Introduction. In S.J. Zaccaro, M.A. Marks and L.A. DeChurch, eds., *Multiteam Systems An Organization Form for Dynamic and Complex Environments*. Routledge, New York, NY, USA, 2012, 3–32.