Knowledge Sharing and Process Improvement in Large-Scale Agile Development

Finn Olav Bjørnson
Department of Computer and
Information Science, Norwegian
University of Science and Technology
7491 Trondheim
+47 97726490
bjornson@idi.ntnu.no

Kathrine Vestues
Department of Computer and
Information Science, Norwegian
University of Science and Technology
7491 Trondheim
+47 92853543
kathrine.vestues@idi.ntnu.no

ABSTRACT

When scaling agile development methods, the principle of continuous improvement is challenged since most practices to support it is described on the team level and not on the project level. In this study we compare practices from two published empirical case studies on large-scale agile development implementations, with a focus on knowledge networks and process improvement. Our findings indicate that continuous process improvement will be influenced by the project type. There is a clear need for dynamic structures for learning and coordination, and these structures need to be supported by a clear decision making process. We also describe informal arenas as an alternative to knowledge sharing in the traditional communities of practice suggested by large scale frameworks.

CCS Concepts

Software and its engineering → Software creation and management → Software development process management → Software development methods → Agile software development

Keywords

Agile software development; large-scale software development; inter-team coordination; continuous software process improvement

1. INTRODUCTION

The agile manifesto was formulated in 2001, since then agile methods have transformed software development practice by emphasizing tolerance for changing requirements, evolutionary delivery and more end-user involvement [2]. Agile development has received widespread interest and “Scrum is now a de facto standard for development in many countries” [5]. Rajlich describes agile development as a paradigm shift in software engineering [14]. This shift has brought about many new topics in software engineering research, but as Dybå and Dingsøyr [7] found in a systematic literature review of the field in 2008, there was initially a lack of empirical data to support the claims of agile, and they

made a call to research for more empirical studies particularly on more mature teams.

In the beginning, agile methods were seen as best suited for small co-located teams, with easy access to users and business experts and developing systems that are not life-critical [16]. However, with the success of agile methods in smaller teams, as well as increased market demand that software value should be delivered in a more continuous manner, the methods were soon applied in large scale projects. With the application in large scale of a method originally designed for small teams, new challenges arose, and a call went out to researchers for more focus on “Agile in the large” which was voted top burning research question by practitioners at the XP2010 conference [8]. The industry also responded to the challenge of scaling agile methods, which have resulted in several frameworks like the Large Scale Scrum (LeSS) [10] and Scaled Agile Framework (SAFe) [11].

The challenge of large scale agile was addressed at a workshop at XP2013, which produced a set of research challenges [3]. This was followed up with a workshop at XP2014 to define principles for large scale agile development [5], and practices of large-scale agile development was again discussed at the workshop at XP2015.

In this paper we report our findings related to one of the research topics identified as “High” priority in the XP2014 workshop, namely “Knowledge sharing and improvement”. The topic deals with ensuring feedback for learning, use of knowledge networks and learning practices. We adopt the view of Wenger [15] that knowledge sharing does not take place in a vacuum but is always embedded in the context in which it occurs. We define improvement in this context as software process improvement, where we adopt the definition of Hansen et al. [9] that it deals primarily with the professional management of software firms, and the improvement of their practice.

One of the principles of agile is that the team at regular intervals should reflect on how to become more effective, then tune and adjust their behavior accordingly. In small scale scrum this is often achieved by the retrospective meeting, but when we scale up the project, new mechanisms are needed in order to coordinate lessons learned from the different teams, and make sure the teams do not adjust their process sub-optimally at the team level at the expense of other teams at the project level.

Our findings are based on a recent project in our research group, where one of the largest software development programmes in Norway was studied, using an exploratory case-study methodology. The detailed findings from the study is described by Dingsøyr et al. [6]. The authors did not participate in the data gathering, but we had access to the data material collected from this

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

XP '16 Workshops, May 24 2016, Edinburgh, Scotland UK

© 2016 ACM. ISBN 978-1-4503-4134-9/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2962695.2962702>

project. The programme is anonymized and referred to as the Omega case.

We compare and contrast our findings with the findings of Paasivaara et. al. [12, 13] who report a descriptive longitudinal case study on a product development project in a large telecommunication company. This case is the only one we have been able to identify that reports empirical data on large-scale scrum implementations with regards to our focus area on knowledge networks and improvement. For comparison in this paper we will refer to this project [13] as the Ericsson case. In accordance with the chosen research topic we aim to provide an answer to the question: How do these two cases differ, and how does this affect their approach towards knowledge sharing and process improvement?

For the remainder of this paper we will first examine the two cases and their respective findings in chapter 2. In chapter 3 we discuss the similarities and contrasts, before wrapping up with conclusions and future work in chapter 4.

2. EMPIRICAL CASES

The Omega case was chosen for study as it was described as the most successful large scale agile development project in Norway at the time. The case also had the benefit of all teams being collocated so as to remove possible bias and interference from teams being distributed. The Ericsson case provides an interesting contrast since it was performed by teams distributed in three different countries. Both cases were chosen because they provide thick empirical descriptions of mature large-scale agile software development efforts, and they represent two different project settings: A temporary development project organization versus a continuous product development organization. We first provide an overview of the context of the two cases, focusing on research methodology, type of software developed, people and teams involved, and timeline and scrum maturity.

Table 1. Context Comparison

Omega [6]	Ericsson [12,13]
Research Methodology	
Exploratory case study conducted retrospectively using documents and focus groups with a total of 25 participants.	Descriptive longitudinal case study, using 52 semi-structured interviews and longitudinal non-participant observation during over 20 site visits over a period of two years
Software developed	
New office automation system for a public department, necessary due to public reform, the content of the reform was not known at the start of the project. Also, the existing office automation system was on a platform that was to be abandoned.	A large and complex systems product - a node that handles a specific type of traffic in telecommunications networks.
People and teams involved	
Single Site. 12 development teams. Three from one consulting company, three from another consulting	Three global sites: Finland, Hungary and the US. 400 people in approximately 40 teams.

company and six internal teams from the public department, consisting of internal people and hired consultants. The project involved 175 people, of whom 100 were external consultants.	
Timeline and Scrum maturity	
Project ran from January 2008 until March 2012. Using Scrum as development framework. At the time of the study the participants had 4+ years of experience using scrum in large scale.	Development started “over 10 years ago” (Paper from 2014 last interviews performed in 2013). They transitioned from waterfall to Scrum during the period. Full scale agile rollout occurred in 2010. At the time of the last interviews they had 3+ years of experience using Scrum in large scale

Next we delve a bit deeper into the organization of the Omega project since it pertains to how knowledge was ultimately shared, before we sum up some of the main findings related to knowledge sharing and process improvement in the two cases.

2.1 Structure of the Omega programme

Omega was managed as a matrix programme, with several projects intersecting. At the top was a programme director, focusing mainly on external relations and a programme manager who focused on operations. Below them four projects, each with their own project manager. These four projects were the business, architecture, development and test projects. The business and development project ran separately in principle, but were intersected by both the architecture and test project to allow information to flow between them.

The business project was responsible for analyzing needs, defining and prioritizing epics and user stories. The architecture project was responsible for defining the overall architecture. The Test project was responsible for testing procedures and approving deliverables. All of these projects were staffed with few lead personnel, and part time staffed by members of the development projects on a needs basis.

The development project was itself split into three subprojects. One lead by the public department Gamma, consisting of six teams. And two more lead by consulting company Alpha and Beta respectively, each with three teams. These development teams worked according to scrum with three-weeks iterations, delivering on a common demonstration day. Each team was staffed with a scrum master, a technical architect with a 50-50 split between architecture work and development, a functional architect split 50-50 between analysis and design, and development, a test responsible who made sure that testing was conducted at team level, this person delivered test criteria to the test project, and finally 4-5 pure developers, a mixture between junior and seniors.

2.2 Findings in Omega

In the Omega case three main findings were identified related to learning and coordination:

First, there were a number of arenas for coordination and knowledge sharing. Each feature team followed scrum practices like the planning meeting, daily stand-up, retrospectives, and demos for each iteration. They used formal arenas such as the

scrum of scrum on company specific teams. To coordinate between these they held what they called a metascrum with project managers from the development, architecture, test, and business projects, as well as subproject managers from the development teams. In the Omega case the scrum of scrum was perceived as a well functioning coordination mechanism. Retrospectives was mainly used on the team level, but a few times on subproject level as well. In addition, there were a number of informal arenas such as board discussions, experience forums, technical corner, lunch seminars, and instant messaging.

Second, the use of arenas changed over time. There were need for information that was not covered and new arenas was established, also arenas were abandoned when they were no longer useful. There was also a change from formal to more informal mechanisms over time. Also, some arenas changed function over time, like internal meetings for one of the consultant companies that changed into an arena for sharing technical knowledge.

Finally, one year into the project, the entire development staff was co-located on an open-plan office floor. The teams were organized around tables and project management for the whole programme and for the subprojects were on the same floor. The open work area was perceived to contribute to efficient coordination and knowledge sharing. Decisions were usually discussed between relevant stakeholders in informal arenas, and then decided upon in one of the formal arenas like the daily scrum, scrum of scrum, or metascrum meetings.

With respect to the development process, most aspects remained the same. Team roles and iterations are example of this. Few changes in the large scale structures in the project were found. Learning and change were, however, evident at both the team level and above. At the team level, the retrospectives contributed to learning and change, at the inter-team level some joint retrospectives were arranged as well as the different coordination arenas that changed over time. Further, deficiencies in information dissemination were met with active experimentation with different new arenas for communication and coordination. Also Omega gradually transformed towards an organizational culture with more trust and collaborative relationships. As more and more developers gained experience with the solution description teams, their overall knowledge of the domain increased and this allowed them to reduce the level of detail in the solution description.

2.3 Findings in Ericsson

One of the key differences between the two cases was the perception of the scrum of scrum (SoS) meeting. Paasivaara [12] describes how scrum of scrum was perceived as dysfunctional coordination mechanism in the Ericsson case, when the number of participants increased. The SoS was gradually replaced by feature SoS meetings that concentrated on coordinating large features involving 3-8 teams. Due to the SoS having bad connotations after the not so successful organization-wide SoS meetings, the feature SoS was later renamed feature communities of practice, since the communities of practice culture was then starting to emerge in the company.

This brings us to the second key difference between the cases, arenas for learning and inter team coordination. In the Ericsson case [13] communities of practices (CoP) [15] were seen as a central mechanism behind the success of large-scale agile implementation. In contrast no official communities of practice were discovered in the Omega case. The Ericsson case is described as having more than 20 CoPs, gathering bi-weekly or on a need basis. CoPs had several purposes including knowledge sharing and learning, coordination, technical work, and organizational development. The

communities were initially used to support the agile transformation. As time progressed they also took on the role of supporting continuous organizational improvements, and helped mitigate some of the most pressing problems with the agile transformation. Most importantly though these communities had decision making authority and could effectively decide on and make changes to processes.

Paasivaara [13] describes eight characteristics of a successful CoP: Interesting topic and concrete benefits, passionate leader, proper agenda, decision making authority, open community, supporting tools to create transparency, suitable rhythm, and Cross-site participation when needed. The evolution of the role of CoPs are also described as going through three phases: First as a support mechanism for agile transformation, secondly as supporting scaling of the agile process and finally establishing its place as the main forum for continuous improvement. Paasivaara also gives a classification of the purpose of the CoPs at Ericsson, four purposes are identified: Knowledge sharing and learning, coordination, technical work, and organizational development.

3. DISCUSSION

How do these two cases differ, and how does this affect their approach towards knowledge sharing and process improvement?

We have a clear difference in the type of project, Omega was a software development project with a clear deadline for delivery with external consultants making up more than half of the project organization, whereas the Ericsson case is a continuous product development project internal to a large company. In other words, we have a temporary project organization versus a continuous product organization. When it comes to size, both can be classified as very large-scale agile development efforts according to the taxonomy in [4]. However, the Ericsson case is clearly larger with a factor of more than three when it comes to number of teams, and in addition the teams are spread over multiple sites and time zones. Finally, the development process over time is different since Omega started the development process with scrum while Ericsson started the agile transformation after a waterfall model had been used for some time. It is hard to specify how much these factors impact the different studies. Temporary organizations like Omega is an area of research in itself, and as Burke et al. [1] describes in their systematic review on temporary organizations, team processes and dynamics in these organizations have remained largely unexplored and offer one major research opportunity in the research agenda. It is beyond the scope of this paper to delve deeper into the team processes which may have resulted from the organization of the project, we will instead offer some takeaways from what we find to be the most interesting comparisons and contrasts between the projects.

The first of the key differences we observe, at least on the surface, is the perceived usefulness of the scrum of scrum meetings. Omega perceives them as a useful mechanism for coordination and knowledge sharing, in Ericsson they were perceived as dysfunctional. The difference in opinion can probably be linked to the size of the meetings. In Omega the meetings were held to coordinate up to six teams, depending on the subproject. Thus they had less participants and a narrower focus since they were all working on related functional areas. The Ericsson case describes how the participants felt a lack of relevance when the meetings were held for upwards of 25 teams. This was alleviated some by introducing feature specific scrum of scrums where 3-8 teams coordinated on similar features. Here the participants felt more relevance, and these meetings were perceived as more useful. These meetings eventually took on the new name of feature communities

of practice, so in the end we can probably compare the scrum of scrum in Omega [6] to the Feature Coordination CoPs of Ericsson [13]. This implies that coordination arenas whatever they are called are most useful when the areas being coordinated are closely related, and the meetings do not grow too large.

At the level above feature coordination, Omega used the metascrum which consisted of managers from the development subprojects as well as architecture, test, and business parts of the project. The most closely related mechanism in Ericsson is probably the End-to-end CoP whose goal was to improve the development flow through the whole organization. In this regard there is a clear difference between the two cases when looking at it from a process improvement angle. In Omega there were extensive use of unofficial meeting arenas, but no organized or supported communities. In order to achieve process improvement at a level above the team-level, it would have to be discussed informally in one of these arenas and then be decided upon in a formal forum like the scrum of scrum or metascrum. In Ericsson, the CoPs had decision making authority and were more open to participants from lower levels, this resulted in them taking on the role of supporting continuous process improvement. We see this reflected in the rate of process improvement in the two cases, in Omega where most aspects of the process remain static, with some optimization at lower levels, while Ericsson is described as having more continuous improvement.

The second clear difference between the two cases is the arenas for learning in Omega versus the communities of practice in Ericsson. These two mechanisms show some resemblance and some differences. In Omega there was clearly a focus on the need for coordination and communication of information outside of the project structures. This was supported by unofficial arenas for discussions and information sharing which changed and adapted over time. However, they did not take the step of forming a structure for supporting communities of practice like the Ericsson case. One factor that might have contributed to this is the colocation of the entire project on the same floor, which meant that with informal channels that could bring ideas to formal forums like the scrum of scrum or metascrum, the project had a mechanism that functioned well enough for their setting. Another factor that might have affected this structure could be the project timeframe and the fact that it was a temporary project setup with several consultants from different companies that would go their separate way at the end of the project. This is contrasted by the Ericsson case which had a continuous product development organization and where the need for continuous improvement was recognized at high levels of the organization. In either case it seemed that they achieved a good balance between the need for self-management and central control. Also, both cases show signs of adaptation over time with Omega changing arenas for coordination, and content of arenas over time, and Ericsson changing communities of practice over time as new needs were discovered.

How do these cases compare to the recommendations from the different large scale frameworks for agile? LeSS [10] prescribes continuous improvement through creating a clear direction, experimentation and focusing on problems not tools. SAFe [11] prescribes relentless improvement as one of four pillars for their SAFe House of Lean. Both frameworks prescribe CoPs for knowledge sharing and exploration. Both cases differ from the frameworks in some regards. Omega chose not to implement CoPs and chose instead to implement and experiment with different informal arenas for information exchange, coupled with a formal decision structure in SoS and metascrum and the whole team collocated on the same floor, this provided a good enough practice

for information sharing and coordination. Ericsson implemented CoPs but took this a step further than the frameworks and gave the CoPs decision making authority and a purpose of continuous improvement.

4. CONCLUSIONS AND FUTURE WORK

We return to our original research topic of knowledge sharing and improvement in large-scale agile. Can we, based on these two cases extend our knowledge of this topic?

While large scale frameworks like LeSS and SAFe suggest CoPs for sharing knowledge and stresses the importance of continuous improvement, we have seen evidence that the picture is not always so clear on what to do, and different approaches can be taken based on the project type. In a temporary project structure like Omega, supporting unofficial meeting arenas was enough, while Ericsson went beyond the CoP as arenas for learning and gave them decision making authority and responsibilities for continuous process improvement. We have also seen that the need for continuous process improvement might differ from the different projects, where a temporary project structure might optimize on the team and inter-team levels and keep the overarching structures in place. How this affects the development process and whether the process could have been optimized further remains an open question.

What we have seen in both cases is the need for dynamic structures in learning. The literature today suggests creating unofficial arenas for learning and coordination, like CoPs, but little is written on how these arenas change over time. We have also seen the importance of establishing multiple arenas for knowledge sharing in the beginning of projects in order to accelerate learning across organizational and project boundaries.

Considering process improvement at the level above the team, there was a clear need for a decision making process. This process might differ between cases, like the unofficial to official process in Omega, or the granting of decision making power to officially supported CoPs in Ericsson. Roughly speaking, we can say that Omega used a more static approach with informal arenas supported with specific roles and projects to ensure knowledge sharing and process improvement, while Ericsson relied on a more dynamic model with CoPs relying on volunteers of people who were passionate about the subject. Both approaches seemed to work for their respective case, however, Ericsson seemed to be more dedicated to continuous improvement at a larger level than Omega. This raises the question of what obstacles might hinder a temporary project organization from having enough focus on process improvement.

Table 2 sums up the most relevant lessons learned from comparing and contrasting the two cases.

Table 2: Lessons learned

Omega	Ericsson
Coordination useful in scrum-of-scrums with 3-6 teams participating.	Coordination useful in featurescrums consisting of 3-8 teams.
To achieve effective coordination, ensure relevance of coordination meeting for all participants.	
Improvement efforts discussed in unofficial forums, decided on in official meetings like metascrum.	End-to-end CoP with decision making authority. With goal to improve the development flow of the organization
To achieve process improvement, ensure a clear decision making process above the team level.	

Unofficial forums for discussion and information sharing that changed and adapted over time.	Process for setting up new CoPs and abandoning old, based on what works and what is needed.
To ensure learning, use dynamic structures and be willing to experiment.	

Returning to the principles of large-scale agile development [5] we can provide some context to principle four: Effective knowledge networks are essential in large-scale development due to the knowledge-intensive nature of software development. We believe the two cases we have examined support this principle. We can also say that while agile methods primarily support the sharing of tacit knowledge in teams, larger projects require new mechanisms. CoPs have been suggested in large-scale frameworks and have successfully been proven to work in some cases, but that does not preclude other mechanisms. Informal arenas that are well supported and recognized by management might be an alternative to CoPs. A CoP might even grow out of such an informal arena. The important part is that the developers know that they have a clear way of taking something from an informal arena to a formal arena where an official decision can be made, and achieving a good balance between self-management and central control.

In our project Agile 2.0 dedicated to large scale agile, we aim to continue investigating mechanisms for supporting continuous improvement both at the team, inter-team and project level.

5. ACKNOWLEDGMENTS

The work on this article was supported by the project Agile 2.0 which is funded by the Research council of Norway through grant 236759, and by the companies Kantega, Kongsberg Defence & Aerospace, Sopra Steria, and Sticos.

6. REFERENCES

- [1] Burke, M. B., Morley, M. J. 2016. On temporary organisations: A review, synthesis and research agenda, *human relations* 69(6), 1235-1258
- [2] Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B. 2012. A Decade of Agile Methodologies: Towards Explaining Agile Software Development. *Journal of Systems and Software* 85, 1213-1221.
- [3] Dingsøyr, T., Moe, N.B. 2013. Research Challenges in Large-Scale Agile Software Development. *ACM Software Engineering Notes* 38, 38-39.
- [4] Dingsøyr, T., Fægri, T. Itkonen, J. 2014. What is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development in *Product-Focused Software Process Improvement*. vol. 8892, A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, M. Raatikainen, Eds., ed: springer International Publishing, 273-276.
- [5] Dingsøyr, T., Moe, N. B. 2014. Towards Principles of Large-Scale Agile Development: A Summary of the workshop at XP2014 and a revised Research Agenda in *Agile Methods. Large-Scale Development, Refactoring, Testing and Estimation*. Vol. 199. T. Dingsøyr, N.B. Moe, S. Counsell, R. Tonelli, C. Gencel, and K. Petersen, Eds., ed Berlin: Springer, 1-8.
- [6] Dingsøyr, T., Moe, N. B., Fægri, T. E., Seim, E. A. 2016. Exploring Software Development at the Very Large-Scale: An Exploratory Case Study and Research Agenda for Agile Method Adaptation, *work in progress*.
- [7] Dybå, T., Dingsøyr, T. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 833-859.
- [8] Freudenberg, S., Sharp, H. 2010. The top 10 Burning Research Questions from Practitioners. *IEEE Software*, 8-9.
- [9] Hansen, B., Rose, J. 2004. Prescription, description, reflection: the shape of the software process improvement field, *International Journal of Information Management* 24, 457-472
- [10] <http://less.works> visited 26.02.2016
- [11] <http://www.scaledagileframework.com/> visited 26.02.2016
- [12] Paasivaara, M., Lassenius, C., Heikkilä, V. T. 2012. Inter-team Coordination in Large-Scale Globally Distributed Scrum: Do Scrum-of-scrums Really Work? In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ed New York: IEEE, 235-238.
- [13] Paasivaara, M., Lassenius, C. 2014. Communities of practice in a large distributed agile software development organization – Case Ericsson. *Information and Software Technology* 56 (Dec 2014), 1556-1577.
- [14] Rajlich, V. 2006. Changing the paradigm of Software Engineering. *Communications of the ACM* 49, 67-70.
- [15] Wenger, E. 1998. *Communities of Practice: Learning, Meaning, and Identity*, Cambridge University Press.
- [16] Williams, L., Cockburn, A. 2003. Agile Software Development: It's about Feedback and Change. *IEEE Computer* 36, 39-43.