Perspectives on Productivity and Delays in Large-Scale Agile Projects

Deepika Badampudi, Samuel A. Fricker[1], Ana M. Moreno[2],

[1] Blekinge Institute of Technology, 371 79 Karlskrona, Sweden
[2] Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain

deba10@student.bth.se, samuel.fricker@bth.se, anamaria.moreno@upm.es

Abstract. Many large and distributed companies run agile projects in development environments that are inconsistent with the original agile ideas. Problems that result from these inconsistencies can affect the productivity of development projects and the timeliness of releases. To be effective in such contexts, the agile ideas need to be adapted. We take an inductive approach for reaching this aim by basing the design of the development process on observations of how context, practices, challenges, and impacts interact. This paper reports the results of an interview study of five agile development projects in an environment that was unfavorable for agile principles. Grounded theory was used to identify the challenges of these projects and how these challenges affected productivity and delays according to the involved project roles. Productivity and delay-influencing factors were discovered that related to requirements creation and use, collaboration, knowledge management, and the application domain. The practitioners' explanations about the factors' impacts are, on one hand, a rich empirical source for avoiding and mitigating productivity and delay problems and, on the other hand, a good starting point for further research on flexible large-scale development.

Keywords: Inductive process improvement, large-scale agile development, grounded theory.

1 Introduction

Agile methods promise lightweight, fast, and nimble development of software solutions [1]. The values and principles of agile methods suit project environments particularly well that are characterized by small, competent, and collocated teams that aim at creating rapid value with small products for well-collaborating customers. The methods' rapid and continuous feedback from customer to development team allows a shared understanding to emerge, rather than requiring requirements to be pre-determined and specified up-front.

Many organizations are appealed by the idea of generating rapid value with emergent requirements. However, when attempting to use agile methods for large-scale product innovation, these organizations discover misalignments between method and environment [29]. Large scale implies distributed collaboration, coordination

among teams, and the presence of many stakeholders that need to be satisfied in addition to the project customer. Product and technology novelty imply competence gaps, potentially both for customer and project team [15]. Misalignments affect project success negatively or lead to failure [9].

To improve project performance, companies invest in process improvement. Such learning organizations actively collect experience and modify their behavior to reflect the insights they have gained [16]. In mature areas, such process improvement is often based on prescriptive frameworks, such as CMMI [10], that benchmark industry best practices. When best practices for specific improvement goals have not been established yet, inductive approaches are used to guide process improvements [5, 27]. An inductive approach exposes past experience and allows the organization to learn from it. If they are attractive enough, the results from inductive process development ultimately become part of prescriptive benchmarking frameworks.

This paper reports early results of such inductive process improvement that aimed at enhancing productivity and reducing delays of large-scale agile development in a particular software development organization. The organization enabled large-scale software product innovation for a multi-national company, a market and technology leader in multiple industry sectors. The organization noticed a misalignment of project needs for predictability and dependability with agile practices. It used inductive process improvement to assess and improve the productivity of their development projects. The assessment elicited challenges and their impact on project roles to identify how to avoid these challenges and to mitigate their effects. The results are a condensed rich description of real-world experiences that enables evidence-based definition of a prescriptive framework to diagnose and improve agility for large-scale software product innovation.

The remainder of this paper is structured as follows. Section 2 describes related work and motivates the research. Section 3 describes the research method. Sections 4 and 5 characterize and discuss the results. Section 6 summarizes and concludes.

2 Related Work

Many organizations feel pressure to produce more at lower costs [23]. Productivity improvements require software projects to reduce development cost, while still ensuring that solutions are technically correct and satisfactory to stakeholders. Usually, this is achieved by increasing development efficiency and avoiding rework. Productivity is also closely related to predictability. Wrong estimates and scheduling problems increase the error rate of investment decisions [12]. Productivity problems and delays affect the company's bottom line because market share erodes rapidly and the market is entered with a little attractive product [3].

A variety of factors affect productivity and delays. The ability to plan is a key determinant: requirements engineering, prototyping, and reuse reduce the need for error correction and rework [2, 3]. Requirements engineering, in particular, enables effort estimation, project negotiation, progress tracking, and high test coverage [12]. Project management problems such as customer and management changes, unrealistic project plans, staffing problems, and inability to track problems early lead to delays

[17]. Other determinants are software architecture, team size, and tooling. Software architecture limits the number of developers that can effectively work together on a software solution [7]. Small teams with better programmers are more productive than large teams [3]. Tools, finally, have positive or negative effects on productivity [6].

Many companies believe that agile methods effectively address productivity problems, in particular because they enable continuous change instead of costly upfront requirements specifications [23]. Some companies were successful: phenomenal productivity was achieved with a shared backlog, shared code ownership, and joint daily Scrum meetings even in globally distributed development [32, 33]. Productivity improvements were also reported in other studies [8].

Productivity suffers if methods are used in an incompatible environment, however. Agile methods shift success determinants from good planning to frequent releases and strong communication [9]. This shift is difficult for companies that are used to heavyweight sequential processes [25] and companies that are confronted with interdependent teams and stakeholders located at different locations [9]. Challenges appear in development and management processes [4, 11]. However, they can be addressed with appropriate practices for improving communication, sharing knowledge, managing trust, and adapting processes [22, 28].

Companies that have adopted agile practices and discover that their development environment is incompatible have limited support for improving development performance. An enabler for identifying effective practices is to understand the development context and how it enables, respectively inhibits success [19]. Without such knowledge, projects outside the agile "sweet spot" [21] feel forced to change again the development method and, due to lack of alternatives, will probably fall back to the old traditional way of development, losing some of the benefits that agile frameworks can provide.

3 Research Methodology

Our work aimed at understanding productivity impediments of projects for large-scale software product development. The here presented embedded multi-case study [34] was part of an inductive process development effort [27] in a software development organization of a multi-national company. The effort aimed at improving the organization's agile development practices by capturing the experience of the employees. Members of multiple projects were interviewed to identify challenges in the application of agile techniques. Grounded theory [31] was used to analyze the impact of these challenges on the various roles involved in the software projects and to understand how productivity problems can be avoided and mitigated. The cause-effect form of the resulting analysis not only supports the specific process improvement, but also represents an empirical basis for the definition of a situational framework with guidelines for flexible large-scale development.

To understand the challenges of the agile projects and the impacts of these challenges, the following research questions were posed:

- RQ1: Which challenges led to productivity problems and delays?
- RQ2: How were the involved project roles affected by these challenges?

Data collection and analysis proceeded iteratively and in parallel. The collected data was analyzed to build a model of causes and effects for the observed challenges. Information needs from the analysis indicated the roles that needed to be interviewed. For example, if an interviewee mentioned a challenge in a particular activity, then the project role responsible for that activity was selected for the next interview. Our industry partner's quality manager identified relevant projects and interviewees to ensure representativeness for the organization and the application domains of the developed software products. A total of 14 representatives for the following roles were selected: product manager, global project manager, architect, integration manager, technology manager, scrum master, developer, and tester. A brief description of each role is provided in the next section. The data collection continued until the saturation point where no new data about the challenges could be obtained.

Semi-structured interviews [30] were used for data collection. During the interviews, the purpose of the study was explained and open-ended questions about challenges and the impact of the challenges were asked. The initial questionnaire was continuously refined on the basis of the analysis results and the information needs discovered during the analysis. Each interview lasted approximately 120 minutes. The interviews were recorded and transcribed.

The transcribed data was analyzed using grounded theory [31] by following the steps iteratively. *Pre-coding:* we have identified the parts of our transcripts that pertained to challenges and impacts. *Open coding:* we have defined codes to label the identified challenges, their causes, and their impacts. Each challenge and impact then was described from the interviewees' perspectives in terms of its characteristics (Table 1). *Axial coding:* we connected challenges with conditions that gave rise to the challenges (Table 2) and the impacts of the challenges (Tables 3 and 4). *Selective coding:* we then identified the central traits of the observed large-scale challenges and discussed them in relation to previous literature and potential solutions.

Flexible research is confronted with the following threats to validity [30]: reactivity, respondent bias, researcher bias, reliability, and generalizability.

Reactivity refers to the way in which the researcher's presence alters the behavior of the subjects involved in the research. This threat to validity was addressed by letting the interviewing researcher stay at the development organization for a prolonged period of time and develop trusted relationships with the interviewees.

Respondent bias refers to the risks of obtaining answers that respondents judge are those the researchers want and of having information withheld that can be used against the respondents. This threat to validity was reduced by aligning the study goals with the interests of the study participants: understanding how to improve their development processes. To check correctness of the obtained answers, one of the researchers studied the company's standard processes and participated as observers in project meetings. Further threat reduction was achieved by triangulating the data among the interviewees.

Researcher bias refers to the preconceptions and assumptions the researcher brings into the study. Researcher bias could have manifested in the selection of the projects and interviewees. This threat to validity was reduced by letting the quality manager of the organization, who was interested in correct and useful results, select the projects and the interviewees and review the questionnaire. The open-ended

interview questions allowed the interviewees to share answers they judged to be important.

Reliability refers to how carefully the research was performed and how honestly the results were presented. Reliability was achieved by following the above-described research design, by transcribing all interviews, by managing coding results with a qualitative analysis tool, and maintaining a chain of evidence.

Generalizability refers to how far the obtained results are applicable and valid. To support generalization beyond the studied organization we developed a model of challenges, causes, and impacts that can be used for generating hypotheses about determinants for productivity and delays in large-scale software product development. In addition, the results were compared with related work to indicate consistency and differences with previous state of knowledge.

4 Results

4.1 The Development Organization

We studied a development organization of a Global 500 company. The company served a large number of markets with a widely diversified portfolio of products, systems, and services. Many of the products were built on leading technologies and contained a significant amount of software.

The development organization developed software solutions with projects requested by product managers. Many of these solutions were established for 5 to 10 years and represented critical parts of products and larger systems that included both hardware and software. The largest software had approximately 5 million lines of code. The products and services targeted customers in a number of industry sectors. They were managed by product managers that worked remotely and acted as product owners to the development projects. The projects were globally distributed with 25 to 100 members allocated to up to 10 Scrum teams and located at up to 4 development sites. Important roles of the Scrum teams were the Scrum master responsible for a team's work process, the developers responsible for component design and implementation, and the testers responsible for quality assurance. Important members of the global project teams were the product manager responsible for product success, the project manager responsible for coordinating the Scrum teams in the global project, the architect responsible for overall product design, the integration manager responsible for composing the overall product, and the technology manager responsible for the development organization. An independent organization verified compliance to regulations.

The development organization had adopted agile development processes, in particular Scrum, for over 5 years and followed agile practices like short iterations, daily stand-up meetings, pair programming, and test-first development. Kano analysis was used for prioritization at the project level and planning poker for development iterations. State-of-art tools were used to manage the product repository that included requirements, agile project management, code, and testing artifacts.

The projects were not implementing Scrum to the letter. Deviations were due to compliance, business practices, and distribution of teams. FDA regulations imposed documentation and traceability requirements, and external testing of the product was not possible until the entire product was ready. Most projects had contracts signed early, and workshops disrupted the regular flow of work. These deviations were one source for the challenges the study discovered.

4.2 Challenges that Affected Productivity and Delays

The practitioners reported a wide range of challenges they perceived affected development productivity and timeliness of software releases. The challenges related to requirements creation and use, collaboration, knowledge, and the product repository. Table 1 gives an overview of the most problematic productivity and delay-affecting challenges that have been reported by the interviewees.

Table 1. Challenges that affected productivity and delays in the agile projects (*italics*: quotes).

Category	Challenge	Characteristics
Requirements Creation	Requirements Quality (RQ)	*Sometimes requirements are not mature enough. If a new technology needs to be implemented, then the requirements are not always well understood.*
	Non-functional Requirements (NFR)	*Sometimes product managers are not fully aware of the non-functional requirements. Also the demonstrations do not demonstrate the NFRs. This comes as a defect when the product goes into system testing.*
	Estimates (ES)	*If the project is for 2 years, it is ok to have estimates much bigger. They might be more than even a man month. In the last step where we plan the single sprint they should be down to single days.*
	Competitors' Influence (CI)	*Time-to-market is influenced by the competitors. It may happen that the competitors come up with a similar product. Then the product needs to release earlier with at least the same features like the competitors to avoid sales loss.*
Requirements Use	Requirements Selection (RS)	*How to split the requirements, how to phase them across different phases of the project, I would say, continues to be a challenge.*
	Requirements Stability (RV)	*When there is a change, it takes a couple of sprints to align everything together. The impact of change can be felt for a longer time*
	Testing Completeness (TC)	*Incomplete testing is another aspect. Some workflows are not fully tested.*
	Integration (IN)	*Global integration reports defects for skipped functionality. This happens because some other team changes something which was not available to us for testing and that has cost us the defect.*
	Clarity of Done (CD)	*Perspective of developers and product managers differ sometimes due to poor understanding of requirements. What we consider done is not considered done by product manager sometimes*
Collaboration	Communication Quality (CQ)	*Communication is another aspect as all team members are not in the same place. The communication between the engineering project head and the product manager is less.*
	Decision-making (DM)	*All the key stakeholders from all teams should be involved in release planning along with the product manager and customer. That would lead to development of a concrete plan.*
	Team Dynamics (TD)	*The scrum masters have own motives in completing their tasks.*
	Test Infrastructure (TI)	*When the developers finish the development the scrum server is not available for testing.*
	Team Stability (TS)	*Sometimes resources leave the project. Then recalculations need to be done. The project may take 15 months instead of 10 months.*

Category	Challenge	Characteristics
Knowledge	Domain and Technology Knowledge (DTK)	*Using a new technology without evaluating it could be a potential risk that can cause a plan to get derailed.*
Product Repository	Progress Measurement (PM)	*Testing may result in re-implementing the user story. This is not always updated to the release backlog.*
	Documentation Quality (DQ)	*Developer pairs code without writing comments because they know each other. But a third person faces lot of issues.*

A majority of the challenges related to requirements creation or use. The creation of clear, mature, and complete-enough requirements that are correctly estimated and lead to a stable project and a well-integrated accepted solution was here as important as in other software development efforts. The use of an agile process did not change this need. The challenges RQ, NFR, ES, and RV are consistent with those reported by studies of large-scale market-driven requirements engineering [20]. New challenges were CI and CD that reflected the importance of product management decisions. Connected to the agile development process were RS, TC, and IN that were due to splitting complex requirements and implementing them stepwise.

The next important group of challenges related to collaboration, knowledge, and the product repository. The collaboration challenges CQ, DM, and TD and the product repository challenges PM and DQ are well-known global software development challenges [18]. The challenge of domain and technology learning DTK is well-known in product innovation [24]. New challenges were the problems of test infrastructure TI and team stability TS. None of these challenges were removed by the agile processes.

4.3 Causes for the Challenges

The practitioners suggested that the challenges were caused by six conditions present in the environment of the development projects. The rationales for why the causes gave rise to the challenges characterize the misalignment of the organization's characteristics and how the agile processes were implemented. Table 2 gives an overview.

Table 2. Conditions that gave rise to productivity and delay-affecting challenges (*italics*: quotes).

Condition	Challenge	Rationale
Project Complexity	Requirements Quality (RQ)	*If a new [complex] technology needs to be implemented then the requirements are not always well understood.*
	Requirements Stability (RV)	*Requirements changes might also come from the development teams. Sometimes a team realized that they needed support from other teams or other components.*
Multiple Teams	Integration (IN)	*Other teams change something which was not available to us for testing and that has cost us the defect.*
	Clarity of Done (CD)	*People don't want to report yellow or red. If you have many teams that all report green, but still have open tasks, this does not give a correct indication of work done*
	Test Infrastructure (TI)	*It happens that the team is ready for beta testing but the beta sites are not available for testing as other teams use the same site.*

Condition	Challenge	Rationale
	Progress Measurement (PM)	*Teams do not always updated requirements that result from defects to the [global] release backlog. As a result release burndown gives a wrong indication on the project progress.*
Multiple Sites	Communication Quality (CQ)	*Lack of communication across locations. Time zones are different. This causes delay when queries need to be answered.*
	Decision-making (DM),	*Sometimes meetings are not done jointly due to time differences. Then only the minutes of meeting are shared after the meeting.*
	Team Dynamics (TD)	*Every scrum group would have their own priorities to finish their tasks. This creates more problems when teams are multi-site.*
Product Characteristics	Non-functional Requirements (NFR)	*There is no denial that NFRs like scalability are ignored in agile projects.*
	Documentation Quality (DQ)	*Test cases need to be written at a later stage as a cleanup process due to FDA regulations.*
	Requirements Selection (RS)	*How to split the requirements, how to phase them across different phases of the project, I would say, continues to be a challenge.*
	Testing Completeness (TC)	*Due to FDA regulations external testing is only done at the end of release and not after each sprint.*
Knowledge Limitations	Domain and Technology Knowledge (DTK)	*If the domain is not understood there could be lot of errors.*
	Estimates (ES)	*It depends how mature the team is. There are overestimations because of which the team needs to stretch.*
	Integration (IN)	*The involved people are still learning about the system. We want to be more efficient and have better quality of the integrated product.*

Product complexity, multiple teams, and multiple sites were conditions related to the scale of the development effort. The development was highly parallel and introduced a need for coordinating teams with joint meeting, shared documentation, consistent progress measurement, and a joint product repository. Shared resources such as test infrastructure needed to be managed. Perturbations, such as requirements changes, perturbed the development streams that needed to be stabilized again.

NFR, RS, and TC were challenges due to misalignments of the agile process with product characteristics. NFR cut across implementation activities of many iterations and were not easily handled with backlogs. Similarly, implementation with short iterations required splitting features, such as the support of a workflow, into multiple parts even-though they would have been preferred to be implemented as a whole. The product domain was regulated and imposed constraints on development documentation and process such as traceability and certification tests.

Deep knowledge of the domain, technologies, the product, and the development organization was needed for effective development. The unavoidable learning was accompanied with estimation and product quality problems.

4.4 Impact of Challenges on Productivity and Delay of Scrum Teams

The Scrum team members reported that the challenges caused problems in project planning, in shared understanding (SU) and coordination between the team, other teams, and stakeholders, and in software quality assurance (SQA). Table 3 gives an overview of the impact of the challenges on the Scrum teams.

Table 3. Impact of Challenges on Scrum Team (*italics*: quotes).

Role	Challenge	Impact	Rationale / Mechanism
Scrum Master	Requirements Quality (RQ)	Planning: Planning uncertainty and overestimation.	*There are overestimations when requirements not clear or they are missing.*
	Estimates (ES)	Planning: Overestimation.	*If I didn't estimate the size of the feature or predict the feature to be unstable then my schedule gets extended.*
Developer	Estimates (ES)	Planning: Inadequate time budget for implementation.	*Estimates from unqualified people do not match real effort.*
	Requirements Stability (RV)	Planning: Deviations from software design and project schedule.	*The reasons for deviation are evolving requirements and some technical challenges.*
	Decision-making (DM)	Planning: Project plan was not concrete enough.	*Involvement of all the key stakeholders from all teams … will lead to development of a concrete plan.*
	Test Infrastructure (TI)	Planning: Deviations from project schedule.	*Also external factors affect the schedule. Developers finish the development, but the scrum server is not available for testing.*
	Domain or Technology Knowledge (DTK)	Planning: Deviations from project schedule.	*Using a new technology without evaluating it could be a potential risk that can cause a plan to get derailed.*
	Communication Quality (CQ)	SU and Coordination: Coordination problems and misunderstandings between stakeholders and developers.	*We lack communication across locations. Time zones are different. This causes delay when queries need to be answered. They think about dependencies, but forget to tell.*
	Decision-making (DM)	SU and Coordination: Team coordination and component consistency problems.	*Workshops should be conducted by having all stakeholders in one place.*
	Domain or Technology Knowledge (DTK)	SU: Software design conflicts between teams at different sites.	*The European architects are not aware of the latest technology and still implement [the old] concepts in new solutions. We learned SE much later with new technology. So we have a problem in accepting that.*
	Documentation Quality (DQ)	SU: Code understanding difficulties and delayed code changes and bug fixing.	*Developer pairs code without writing comments because they know each other. But a third person faces lot of issues.*
	Clarity of Done (CD)	SQA: Failed acceptance of features.	*Perspectives of developers and product managers differ sometimes. What we consider done is not by product manager.*
Tester	Requirements Selection (RS)	Planning: Varying test effort between sprints with ineffective use of test resources. Re-work of tests.	*During the sprints the test cases are written just for requirements without considering the [whole] workflow … When the workflow starts coming, the test cases have to be modified to a large extent.*
	Decision-making (DM)	SQA: Insufficient alignment of software design and tests.	*Testers don't always get into design discussions because they are pre-occupied with testing the previous sprints.*

Most of the planning problems were visible in the uncertainty of estimates and plans that led to inadequate time budget and deviations from project schedule. They affected Scrum masters and developers. The uncertainties were caused by unclear and unstable requirements, insufficient qualification, competence, and participation of decision-makers, and scarce shared resources. Testers were confronted with another kind of planning problem. Splitting the implementation of requirements over multiple releases led to uneven distribution of effort and to re-work of test cases.

The problems in shared understanding with other teams and with stakeholders were encountered by developers. These problems were visible in misunderstandings and coordination problems that led to inconsistent design and development results and ultimately resulted in re-work. The problems were caused by challenges of insufficient knowledge, communication, documentation, and participation in decision-making.

The quality assurance problems were encountered by developers and testers. Software and tests were insufficiently aligned and features failed acceptance by stakeholders and users. These problems were caused by problems in shared understanding and insufficient participation in decision-making had to be corrected with re-work.

4.5 Impact of Challenges on Productivity and Delay of Global Project Teams

The managers and architects indicated that the challenges caused problems in plan quality, in development capacity, in coordination between teams, in shared understanding between teams and stakeholders, and in software quality assurance. Table 4 gives an overview.

Table 4. Impact of Challenges on Global Projects (*italics*: quotes).

Role	Challenge	Impact	Rationale / Mechanism
Product Manager	Competitors' Influence (CI)	Planning: Changes in time-to-market and priorities.	*The competitors come up with a similar product. Then the product needs to release earlier with at least the same features.*
	Team Stability (TS)	Planning: Re-planning with scope reduction or deadline postponement.	*[When project members leave] the management does not have budget for additional head count. In that case the deadline is increased or the scope reduced.*
Project Manager	Testing Completeness (TC)	Planning: Underestimated effort for bug-fixing.	*The external testing is only done at the end of a release and not after each sprint. It might reveal that the algorithm is not fully tuned to real world cases. Another 2 or 3 weeks are spent on adjusting the product.*
	Communication Quality (CQ)	SU: Misunderstandings between product and project managers and remote team members	*The reason for delay is lack of clarity at each development step - design, coding, and testing. This is because of limited communication across multiple sites.*
	Team Dynamics (TD)	Coordination: Coordination problems between teams.	*Typically interdependency is not really considered. Every scrum group has its own priorities to finish its tasks.*
	Progress Measurement (PM)	Coordination: Coordination problems among development teams.	*Re-implementation due to bugs or changes from customer is not updated to the release backlog. Some scrum teams may not be aware of the changes.*

Role	Challenge	Impact	Rationale / Mechanism
Architect	Non-functional Requirements (NFR)	Planning and SQA: Defect discovery in system testing or feature delivery. Late costly changes.	*Demonstrations do not demonstrate the NFRs. This comes as a defect in large scale testing or in test of the system limits. This requires change of design, which is costly.*
	Requirements Stability (RV)	Planning and Coordination: Solution redesign during development. Plan changes. Increased coordination effort.	*Changes in NFRs caused refactoring of design and code.*
	Integration (IN)	SQA: Irreproducible defects at integration testing and difficult root-cause analysis.	*Other components may have caused the defect. We see a trend that defects at this stage are not reproducible or consistent. Fixes for these defects are not easy.*
Integration Manager	Integrations (IN)	Capacity: Not enough people working on integration	A dedicated integration team was not setup until last year: *there are not enough people working on it as the people who are involved are still learning about the system.*
	Communication Quality (CQ)	Coordination: Incomplete awareness of dependencies.	*Even though at some instance they think about dependencies then they may forget to tell. Then we don't find out.*
Technology Manager	Estimation (ES)	Capacity: Teams overloaded with work.	*Actual work is much more than people would think.*
	Requirements Stability (RV)	Capacity: Congested backlogs.	*The impact of change can be felt for a long time.*
	Clarity of Done (CD)	Coordination: Wrong understanding of real progress.	*If all [teams] report green but they still have some open tasks then at the end to the management it is all green.*

Many of the problems at the global project level were not visible at the Scrum team level and related to enabling and coordinating the teams and integrating their results. Problems of shared understanding were less a concern than on Scrum team level. Planning problems were experienced at a similar extent, but with different causes.

The planning problems affected first product managers, project managers, and architects. Market changes and resource problems led to scope and deadline changes. Requirement changes and failed external regulatory tests led to redesign, delays, and increased coordination effort. Related were capacity problems that were stated by the integration and technology managers. The learning process and repercussions of changes congested backlogs, overloaded teams, and introduced delays.

Coordination problems were mentioned by all interviewed roles except the product managers. Sub-optimized plans, inconsistent reporting, insufficient communication, and requirements changes caused misaligned work, inconsistent work results, and wrong understanding of real progress. The communication challenges also introduced problems of shared understanding between management and remote teams. Together with ignored and unstable NFR they led to hard problems in quality assurance.

5 Discussion

Many of the reported challenges were well known. They represented a selection of challenges reported in market-driven requirements engineering [20], global software engineering [18], and innovation [24]. Agile development did not change importance

of these challenges. Instead it added the previously hidden angle of product management and introduced new problems such as those due to stepwise implementation of complex requirements.

The development organization showed a need for predictability, dependability, stability, and effective use of an appropriate amount of resources. On a global level, it turned to solutions offered by planning, coordination, and communication. The complexity of the products and of the organization, however, led to the described challenges that generated productivity problems and delays. The problems generated by these challenges differed depending on the organizational level. The Scrum teams struggled mainly with plan stability and adherence, shared understanding, and quality assurance. The global projects battled mainly with project plans, enabling and coordinating the Scrum teams, and integrating results.

The study results were partially consistent with previous research on determinants for productivity and delays. Many determinants of the studied projects were the same as the determinants of pre-agile projects [2, 3, 12, 17]. Requirements could only be stabilized when the product and product use were clear enough. Unclear requirements, limited knowledge of domain, technology, and the organization, and communication problems led to uncertain estimates, unstable plans and integration, and quality problems with the consequent need for rework.

Determinants that were not reported by the subjects to be problematic were team sizing [3] and tooling [6]. They seemed to have been adequately addressed by the organization.

The study discovered new determinants that affected productivity and delay: stability of markets and organization, consistency of the development process with product characteristics, and support of complexity of the organization. Releases of competitive products and personnel fluctuation affected scope, deadlines, and capacity. Complex requirements, regulations that imposed documentation and product certification, and separation of product development and maintenance were difficult to handle with the chosen agile approach. Shared understanding, collaboration between teams, and consistent reporting were addressed unsatisfactorily, especially because they led to costly ripple effects.

Solutions are known that can help to avoid many of the challenges and mitigate their impact on productivity problems and delays. For example, an approach to stabilizing requirements is structured handshaking between stakeholders and development teams with implementation proposals [13]. Implementation proposals allow focusing design and prototyping on critical features and stabilizing the concerned requirements with stakeholder feedback. Sufficient coverage of requirements with implementation proposals increased reliability of project plans.

Requirements structuring with feature trees modularizes specifications and plans according to alternative decision options [14]. Such modularization reduces planning complexity, simplifies progress reporting, and integrates backlogs of individual teams in a consistent manner.

Collocation of some members of distributed teams with scrum masters and product owners and regular, well-prepared global scrum team meetings improves shared understanding and team coordination, and reduces integration problems [33]. Other

collections of practices exist and provide concrete approaches for addressing challenges related to scaling agile development [22].

The overall development throughput can be improved by capturing the flow of software development by tracking the lifecycle stage of features and visualizing progress with cumulative flow diagrams [26]. This specific approach can be used as an early warning system and for identifying bottlenecks.

The list of solutions is by far not exhaustive. Selection of an appropriate combination of practices and evaluation of their effects is the concern of the next process improvement steps at the studied organization. Research towards understanding the fundamental principles of productivity and delays in large-scale agile development will support that work.

6 Conclusions

This paper presented the results of an empirical study that examines the challenges that affected productivity and delays encountered in large-scale agile development of a global product company. Data was collected from 14 interviewees and covered 8 roles in 5 relevant projects.

In relation to RQ1, which challenges led to productivity problems and delays, 17 challenges were identified that were caused by project and organizational complexity, by product characteristics, and by knowledge limitations. Many of the challenges were well known, but have not been removed by the agile development process. Instead, the agile focus added new problems such as those due to stepwise implementation of complex requirements.

RQ2 asked how the involved project roles were affected by these challenges. The interviewees identified 28 mechanisms of how the challenges affected the roles. The problems at the global project level were mostly about enabling, planning, and coordinating the Scrum teams and integrating their results. The problems at the Scrum teams level were about shared understanding, planning, and quality assurance.

Interestingly, the organization did not abolish planning for their large projects. Instead, consistent with previous research on productivity and delays, it wanted predictability, dependability, stability, and effective use of resources. Known determinants for productivity and delays were confirmed and new ones related to software product management, process-product alignment, and process-organization alignment discovered.

In sum, the study describes an in-depth analysis of an organization that has adopted agile processes for large-scale product development, discovered misalignments of this approach with the project context, and intends to adjust its processes to improve productivity and delays. The results are a basis for selecting appropriate solutions and for better understanding principles of productivity and delays with future theoretical and empirical studies.

Acknowledgments

This work was funded by The Knowledge Foundation in Sweden under a research grant for the Blekinge Engineering Software Qualities (BESQ) project. We would like to thank our anonymous industry partner for enabling the here reported research.

References

1. Abrahamsson, P., et al., *Agile software development methods: Review and analysis*. VTT Publications 478. Vol. 478. 2002, Espoo: VTT.
2. Basili, V.R., L. Briand, and W. Melo, *How Reuse Influences Productivity in Object-Oriented Systems.* Communications of the ACM, 1996. **39**(10): p. 104-116.
3. Blackburn, J., G. Scudder, and L. Van Wassenhove, *Improving Speed and Productivity of Software Development: A Global Survey of Software Developers.* IEEE Transactions on Software Engineering, 1996. **22**(12): p. 875-885.
4. Boehm, B. and R. Turner, *Management Challenges to Implementing Agile Processes in Traditional Development Organizations.* IEEE Software, 2005. **22**(5): p. 30-39.
5. Briand, L., K. El Emam, and W. Melo, *An inductive method for software process improvement: concrete steps and guidelines*, in *Elements of Software Process Assessment & Improvement*, K. El Emam and N. Madhavji, Editors. 2001, Wiley-IEEE Computer Society.
6. Bruckhaus, T., et al., *The Impact of Tools on Software Productivity.* IEEE Software, 1996. **13**(5): p. 29-38.
7. Cain, J. and R. McCrindle, *An Investigation into the Effects of Code Coupling on Team Dynamics and Productivity*, in *26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*. 2002: Oxford, UK.
8. Cardozo, E., et al., *SCRUM and productivity in software projects: a systematic literature review*, in *14th international conference on Evaluation and Assessment in Software Engineering (EASE'10)*. 2010: Keele, UK.
9. Chow, T. and D.-B. Cao, *A survey study of critical success factors in agile software projects.* Journal of Systems and Software, 2007. **81**(6): p. 961-791.
10. CMMI Product Team, *CMMI for Development, Version 1.3*. 2010, Carnegie Mellon University.
11. Cohn, M. and D. Ford, *Introducing an Agile Process to an Organization.* IEEE Computer, 2003. **36**(6): p. 74-78.
12. Damian, D., et al., *Requirements payoff: An empirical study of the relationship between requirements practice and software productivity, quality and risk management*. 2003, University of Victoria.
13. Fricker, S., et al., *Handshaking with Implementation Proposals: Negotiating Requirements Understanding.* IEEE Software, 2010. **27**(2): p. 72-80.
14. Fricker, S. and S. Schumacher, *Release Planning with Feature Trees: Industrial Case*, in *Requirements Engineering: Foundations for Software Quality (RefsQ 2012)*. 2012: Essen, Germany.

15. Garcia, R. and R. Calantone, *A Critical Look at Technological Innovation Typology and Innovativeness Terminology: A Literature Review.* The Journal of Product Innovation Management, 2002. **19**(2): p. 110-132.
16. Garvin, D., *Building a Learning Organization.* Harvard Business Review, 2000. **71**(4): p. 78-91.
17. Genuchten, v., *Why is Software Late? An Empirical Study of Reasons For Delay in Software Development.* IEEE Transactions on Software Engineering, 1991. **17**(6): p. 582-590.
18. Herbsleb, J. and D. Moitra, *Global Software Development.* IEEE Software, 2001. **18**(2): p. 16-20.
19. Hoda, R., et al., *Agility in Context*, in *OOPSLA/SPLASH'10*. 2010: Reno/Tahoe, Nevada, USA.
20. Karlsson, L., et al., *Requirements Engineering Challenges in Market-Driven Software Development - An Interview Study with Practitioners.* Information and Software Technology, 2007. **49**(6): p. 588-604.
21. Kruchten, P., *Scaling Down Large Projects to Meet the Agile Sweet Sport*, in *IBM developerWorks*. 2004, IBM.
22. Leffingewell, D., *Scaling Software Agility: Best Practices for Large Enterprises.* 2007: Addison-Wesley.
23. Lindvall, M., et al., *Agile Software Development in Large Organizations.* IEEE Computer, 2004. **37**(12): p. 26-34.
24. Lynn, G., J. Morone, and A. Paulson, *Marketing and Discontinuous Innovation.* California Management Review, 1996. **38**(3): p. 8-37.
25. Nerur, S., R.K. Mahapatra, and G. Mangalaraj, *Challenges of Migrating to Agile Methodologies.* Communications of the ACM, 2005. **48**(5): p. 73-78.
26. Petersen, K. and C. Wohlin, *Measuring the flow in lean software development.* Software Practice and Experience, 2010. **41**(9): p. 975-996.
27. Pettersson, F., et al., *A practitioner's guide to light weight software process assessment and improvement planning.* Journal of Systems and Software, 2007. **81**(6): p. 972-995.
28. Ramesh, B., et al., *Can Distributed Software Development be Agile?* Communications of the ACM, 2006. **49**(10): p. 41-46.
29. Reifer, D., F. Maurer, and H. Erdogmus, *Scaling Agile Methods.* IEEE Software, 2003. **20**(4): p. 12-14.
30. Robson, C., *Real World Research: A Resource for Social Scientists and Practitioner Researchers*. 2nd ed. 2002: Blackwell Publishing.
31. Strauss, A. and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. 1998: SAGE Publications.
32. Sutherland, J., et al., *Fully Distributed Scrum: Linear Scalability of Production between San Francisco and India*, in *Agile Conference (AGILE'08)*. 2009: Toronto, Canada.
33. Sutherland, J., et al., *Disributed Scrum: Agile Project Management with Outsourced Development Teams*, in *40th Hawaii International Conference on System Sciecnes*. 2007: Hawaii, USA.
34. Yin, R.K., *Case study research: Design and methods*. 2008: SAGE Publications.