Group Learning and Performance in a Large-scale Software Project: Results and Lessons Learned

Ricardo Britto
Blekinge Institute of Technology
SE 37179 Karlskrona, Sweden
ricardo.britto@bth.se

Darja Šmite
Blekinge Institute of Technology
SE 37179 Karlskrona, Sweden
darja.smite@bth.se

Lars-Ola Damm
Ericsson
SE 37133 Karlskrona, Sweden
lars-ola.damm@ericsson.com

ABSTRACT

Background: Research on teams originated from the social sciences and brought a number of new topics into the repertoire of software engineering. Teams and teamwork are recognized for the promised benefits of i.e. increased performance. Performance is often linked to experience gains, and along with individual learning teamwork facilitates what is recognized as group learning.

Aims: In this paper, we report our lessons learned from an attempt to study the relationship between group learning and performance in a large-scale software project.

Method: We conducted an exploratory case study of an on-going large-scale distributed project in Ericsson. The data collected included archival data and both unstructured and semi-structured interviews. The data was analyzed using descriptive statistics, charts and regression analysis.

Results: The results suggest that some teams improved their performance over time until they were forced to work cooperate with several other teams. However, it is not completely clear role of accumulated experience and other aspects, such as team stability and number of developers.

Conclusions: We believe that ad-hoc team formation and team member rotation might have resulted in a failure to benefit from group learning. In addition, the fact that developers worked simultaneously on different tasks could have hindered their performance. However, we believe that further research must be conducted to provide a stronger evidence about the identified result. Based on the experience acquired by conducting this study, we report some lessons learned that can support researchers and practitioners when investigating the topic addressed in this paper.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – Programming teams, Productivity.

General Terms

Performance, Human Factors.

Keywords

Group learning, Team performance, Large-scale software development, team turnover.

ESEM Conference'16, September 8–9, 2016, Ciudad Real, Spain.

1. INTRODUCTION

Instead of individual developers and their performance, modern and especially innovative software organizations shifted their focus towards teams as the basic work unit, giving rise to a radically new approach to managing software projects [1]. This, in turn, has inspired many researchers in software engineering and related domains to conduct empirical studies with software teams as the main units of analysis. Cultivation of teams and teamwork has been associated with increased performance, innovation, and employee satisfaction [1]–[5]. But what is the role of teamwork in determining higher performance?

Performance is said to improve over time, as individuals and the team accumulate experience doing their work (autonomous learning), i.e. there is a learning curve that relates experience and performance [6] (see Figure 1). When it comes to teamwork, individuals in a team support each others learning through e.g. asking questions, seeking feedback, experimenting, reflecting on results, and discussing errors or unexpected outcomes of actions [3]**Error! Reference source not found.**. Thus teamwork enables greater performance improvements than individual work.

Interestingly, while performance of teams has attracted a lot of attention, group learning as a construct has not been included in the common agenda of software engineering research.

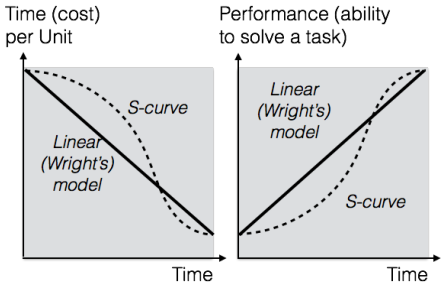

Figure 1: Examples of different learning curves (based on Anzanello and Fogliatto [6])

In this paper, we aimed at investigating the relationship between group learning, focusing on autonomous learning, and performance in a large-scale software project that involved several teams distributed around the globe. Our findings are a part of a larger empirical investigation about how individuals and teams learn in large-scale distributed software projects. Our empirical study discussed in this paper addressed the following research questions:

RQ1: Do software development teams in large-scale distributed projects improve their performance over time?

RQ2: What factors impact autonomous group learning in large-scale distributed software projects?

The main contribution of this paper is two-fold:

- We present empirical results on how newly on-boarded software teams perform over time in a large-scale distributed project that has evolved for over fifteen years.
- We share the lessons learned from our attempt to investigate autonomous group learning in the aimed context.

The reminder of the paper is organized as follows. Related work and the motivation for our study is summarized in Section 2. Section 3 outlines our research methodology. Section 4 presents and discusses our results. Section 5 presents the lessons learned during the conduction of the study reported herein. Finally, Section 6 concludes the paper with the summary of our findings and plans for future work.

2. RELATED WORK

Research on teams originated from the social sciences and brought a number of new topics into the repertoire of software engineering, information systems and other related research disciplines. These include teamwork, personality characteristics of team members, interpersonal relationships among team members, and issues with team composition [7], to name a few. Soon teamwork in software teams has been recognized as an enabler and determinant of performance [8].

Performance (e.g. productivity, quality, efficacy and efficiency) in software development is expressed in different ways, such as the effort needed for creating the wanted outcome or a number of post-delivery defects, among others. Performance is often linked with experience, which can be related to learning and the acquisition and accumulation of knowledge, skill and competence [9]. In this section, we summarize the key concepts and existing research on learning, learning curves and their application in software engineering.

2.1 Learning

Learning is defined as "the acquisition of knowledge or skills through study, experience, or being taught" [10]. To better understand the relationship between teamwork and learning, we hereby describe the key concepts related to learning [11]:

Knowledge is the result of an interaction between the capacity and the opportunity to learn. It is in general associated with formal learning (organized and structured learning), although can eventually be the result of non-formal (unplanned learning) or informal learning (experiential or accidental learning).

Skill is the combination of mental and physical capabilities that demand practice to acquire. In many cases, the previous obtainment of knowledge is a pre-requisite for the achievement of a particular skill. It is in general associated with informal learning, although can eventually be the result of non-formal or formal learning.

Competence is the extent to which individuals interact effectively with the environment. It describes personality aspects associated with better performance and higher motivation of individuals. It is acquired through informal learning.

Most of the studies focusing on learning and performance are based on the assumption that performance improvements are due to autonomous learning (learning by doing). However, individuals in a team support each others learning in a number of ways. ***Group learning*** is a process of reflection and action, which encompasses different learning behaviors, such as asking questions, seeking feedback, experimenting, reflecting on results, and discussing errors or unexpected outcomes of actions [3]. In fact, team learning is about a new understanding shared by all the team members about how the team is expected to behave [12].

2.2 Learning curves

A ***learning curve*** describes the performance of teams or individuals in a mathematical way. It was proposed by Wright [13] based on observations of how the costs associated with assembling airplanes decreased as the involved workers accumulated experience in doing the same type of task.

Learning curves can be modeled using univariate or multivariate models, e.g. log-linear, hyperbolic and exponential models [6]. Log-linear models are most frequently employed due to their simplicity. The original model proposed by Write [13] is a log-linear model, which is represented by Equation 1.

$$Y = CX^b \quad (1)$$

In Equation 1, y is the average time (or cost) per unit demanded to produce X units (cumulative experience) and C is the time (or cost) to produce the first unit. The parameter b, also known as ***learning rate***, represents the slope of the learning curve. Equation 1 has been modified by the research community and resulting in other versions of the aforementioned power law, e.g. the De Jong and S-curve models [6].

2.3 Learning curves in software engineering

The learning curve phenomenon is well explored in domains such as electronic, automotive, construction and chemical industries [6]. Nevertheless, the topic of learning curves in software engineering is relatively recent, although the applicability of learning curves models to relate cumulative experience and performance in software development has been confirmed in a number of studies, as described in related literature reviews [14], [15].

In a systematic review on team learning in information systems development, Spohrer *et al.* [15] concluded that accumulation of experience determines team performance. However, none of the five studies focusing on learning curves reviewed by the authors provided an empirically based explanation for the identified causal relationships.

In the following, we summarize the results of the studies on learning curves in software engineering.

Tüzün and Tekinerdogan [14] investigated the impact of the learning curve phenomenon on the return of investment (ROI) in the software product line engineering and concluded that the learning curve has a clear impact on the ROI of software development companies, although such an impact gets lower when the number of products of a particular software product line increases.

Huntley [16] investigated the learning curve phenomenon in open-source programming projects. He applied regression analysis on the data collected from Apache and Mozilla projects to analyze the relationship between experience and bug cycle times and identified a higher impact of the learning curve for a mature project compared to an emerging project, i.e. the learning curve phenomenon is dependent on the context of each project.

Boh *et al.* [9] examined learning curves on an individual, group and organizational level. The authors performed regression analysis of data from 14 years of systems development work in a telecommunication domain and concluded that specialized experience impacts primarily the productivity associated with tasks fulfilled independently by team members (the more

specialized experience from the same product, the greater the productivity in doing individual tasks), while diverse experience has the biggest impact on the productivity associated with tasks performed by more than one person, on both group and organizational levels (the more diverse experience from related and unrelated products, the greater the productivity in doing group or organizational tasks).

Huckman *et al.* [17] investigated the impact of team stability, team familiarity and role experience on performance. The authors performed regression analysis of 1,004 development projects completed over two years and found that team familiarity and role experience impacts the quality of the developed software (in terms of number of post-delivery defects) and the adherence to the planned budget and schedule; the more experience, the smaller the number of post delivery defects and the greater the adherence.

Narayanan *et al.* [18] focused on investigating the relationship between task variety and individual learning in software maintenance tasks. They performed regression analysis of the data from 88 individuals organized in 20 different groups, covering 5,711 different tasks fulfilled over six years. The authors identified that specialized experience lowers the effort to fulfill a task; task variety can lower effort and increase productivity, however too much variety can hinder productivity. Finally, the authors identified that team turnover impacts negatively the productivity of teams; the higher the team turnover, the lower the productivity.

Zorgios *et al.* [19] proposed an explanatory theory for team learning related to software development. The authors modeled the interaction between learning rates of development teams and improvements in their productivity, establishing a causal relationship between the human capital and different types of learning of organizational teams and the productivity curve. Regression analysis was conducted on the data from 3,104 projects extracted from the repository CD10 of the ISBSG[1] and confirmed the proposed theory; the learning curve model explains the productivity variation caused by different types of organizational learning.

The studies discussed above all focus on either confirming that cumulative experience impacts performance in software teams or to presenting models to estimate the effort/cost related to software projects. However, none of the current studies have deeply investigated the impact of different team-related factors on the learning rate in the software development team context. They also fall short of suggesting recommendations to speed up learning or support decision-making process based on the learning profiles of different teams.

3. RESEARCH METHODOLOGY

To address our original research question we have conducted an exploratory case study [20]. The initial purpose of our study was to explore team performance changes over time based on a retrospective analysis of archival data, aiming at understanding the relationship between autonomous group learning and performance improvements in large-scale distributed software projects (RQ1), and identifying factors that impact autonomous group learning in this context (RQ2). However, our preliminary findings from analyzing a subset of the gathered data indicated that the expected analysis is not as straight forward as initially planned.

[1] International Software Benchmarking Standards Group.

3.1 Case and unit of analysis selection

The studied case is a large-scale distributed project at Ericsson, a large telecommunication company headquartered in Sweden. The company is one of the industry participants in the research project focusing on global software development, and thus was selected through convenience sampling. The project for the study was selected in consultation with the company representatives as a case suitable for studying multi-team project, whose degree of global distribution increased significantly during the studied period including a large ramp-up at one new site. The unit of analysis in this case study is a software team. All teams in the project are included in the overall investigation, and a subset of teams (all teams in one location) are used for analysis in this paper.

The studied project is related to the development of a part of a large system in the telecommunication domain, which originated in Sweden and has evolved for over 15 years. Many different technical and methodological changes were introduced during this time, such as changing the programming language used to develop/maintain the product (from C++ to Java) and changing the software development methodology (from plan-driven to agile). By the time of our investigation, the product was being developed/maintained by over 150 employees working in teams located in Sweden, India, Italy, USA and Turkey.

The offshore locations were added in response to the growing demands for resources and in order to implement market-specific customizations. The last expansion happened in India, where ten teams were on-boarded in the project in late 2013.

The work in the studied project follows agile software development principles. All teams are cross-functional by design, i.e. all members ought to be able to perform design, testing and programming duties. In India, there are ten development teams consisting of five-six members in each team. Teams receive an end-to-end responsibility for designing and implementing a development task, such as product customization in case of Indian teams. Indian teams and other remote teams are supported by software architects (eleven in our case) from the original development location in Sweden in addition to the the local supporting roles. The architects supported the teams in India by responding to questions related to the software architecture and also by providing feedback on their work results through code reviews. In some urgent or particularly complex situations, the architects also participated in actual code implementation.

Some work items were co-developed by the teams in India and more experienced teams located in Sweden, the USA and Italy. In those work items, the more experienced teams supported the teams located in India, which were not experienced enough to conduct the demanded the work by themselves.

In this paper, our analysis was based on a sub set of the collected data, specifically focusing on the Indian site. The decision to focus on the Indian site first was driven by the interest of the company in the results and ability to provide access to the data necessary for the researchers, since this was the last site to be incorporated in the project.

3.2 Research design

The first two authors of this paper conducted unstructured interviews with the main responsible for the investigated case. We learned that, when comparing the Indian teams with more experienced teams in other locations, there is a clear difference in the respective performances. Thus, project managers at Ericsson

were interested in understanding the reasons for the perceived differences, leading to strategies to support the teams in India in improving their performance, and was for onboarding new teams in the future.

To do so, we reviewed existing research literature on group learning and decided to first investigate whether the teams located in India were improving over time by doing their work, i.e. whether there is evidence of autonomous learning. Thus, we decided to conduct a quantitative analysis through descriptive statistics, charts and regression analysis. We complemented the quantitative approach with some unstructured and semi-structured interviews.

3.3 Data collection

The data in the study was collected from a number of sources and analyzed in iterations, as follows.

We first conducted several **unstructured interviews** to understand the context and the motivation for the study, which were performed independently by the first and the second author, having as the interviewee the project responsible (the third author).

To model the learning curves of the investigated teams, we surveyed **archival data** related to fifteen work items (product customizations) carried out over a period of two years, between 2014 and 2015. The data was extracted from company project management systems into an Excel spreadsheet .

The task complexity data for the studied work items was missing. Thus, we conducted four **semi-structured group interviews** with eight software architects who participated in the fulfillment of the work items and had extensive knowledge about the project. The group interviews were conducted by the first author, with participation of the third author and consultation of the second author. The relative complexity of the work items was measured by the architects, using a planning poker based approach. As a result, a numerical value was attributed to each of the investigated work items.

We later on used the complexity values to calculate the productivity of each team in each work item. We calculated the productivity of a work item as the ration between its complexity and its associated actual effort. Note that we were not able to calculate in isolation the productivity of each team in a particular work item, since it was not possible to identify the part of the complexity carried by each team in each work item. We tried to do so by tracking the number of lines of code committed by a team during the fulfillment of a work item. However, in some work items no line of code was implemented, i.e. either testing was carried out or documentation was made and vice-versa.

Rather, we calculated the productivity using the overall complexity and the total actual effort, including the effort spent by all the teams in each work item. For example, if team x and team y participated in work item 1, the productivity of both is considered as being the same in relation to work item 1.

Finally, to be able to conduct a regression analysis and identify the impact of the cumulative experience (learning by doing) on team performance, we obtained the following additional variables:

- **Cumulative experience –** It is represented by *X in* the original power law used to calculate learning curves (see Equation 1). It is calculated by Equation 2, where X_{ki} represents the cumulative experience of team *k* after the conclusion of work item *i*, with $X_{k1} = 1$ (after the conclusion of the first work item). Note that we adapted this equation to the specifics of our study, i.e. by in the given work item we considered the cumulative experience from both finished and on-going work items. We did so because in our case there were several situations, in which the teams were conducting work items at the same time.

$$X = \sum_{i=1}^{n} X_{ki-1}, X = 1 \quad (2)$$

- **Number of developers –** It represents the number of developers of a particular team involved in the implementation of a work item.

- **Team stability –** It relates to the experience of working together and reflects the differences in the team composition, given the formal team boundaries, between two subsequent work items. This variable relates to the formal teams (in our case, five teams, as described in Section 4). It is calculated by Equation 3, where TS_{ki} represents the number of changes in participants from team *k* when comparing work item *i* and *i-1*, DO_{ki} represents the number of members from team *k* that were involved in work item *i-1* but were not involved in *i*, while DI_{ki} represents the number of members from team k that were not involved in work item *i-1* but were involved in *i.*

$$TS_{ki} = DO_{ki} + DI_{ki} \quad (3)$$

- **Cross-team cooperation** – The number of other teams in addition to the investigated team that are involved in the fulfillment of a work item.

- **Cross-team cooperation continuity –** It relates to the difficulties of inter-team cooperation (on a team level) and dispersed work (on a site level) and represents the changes in the number of additional teams between two subsequent work items and it is calculated by Equation 4, where TD_{ki} represents the number of changes in the composition of teams/sites working on two sequential work item *i* and *i-1*, TO_{ki} represents the number of teams that were involved in work item *i-1* but were not involved in *i*, while TI_{ki} represents the number of teams/sites that were not involved in work item *i-1* but were involved in *i.*

$$TD_{ki} = TO_{ki} + TI_{ki} \quad (4)$$

- **Number of architects –** It represents the number of software architects involved in the execution of a particular work item.

3.4 Data analysis

Before performing the data analysis, we performed a sanity check of the data, to remove any existing inconsistency. To do so, the first and third authors of this paper performed independent analysis of the data.

To analyze the data, we calculated descriptive statistics, we plotted charts for each team to show the relationship between productivity and cumulative experience, and we also employed linear regression analysis to identify factors impacting autonomous group learning.

The preliminary analysis results were discussed in a **feedback meeting** and a number of **informal follow-up discussions**. Reflections from the feedback were discussed among the authors and resulted in the formulation of the lessons learned presented in this paper (see Section 5).

4. RESULTS FROM THE CASE STUDY

When investigating the fifteen work items, we identified that out of the ten teams located in India, only seven teams participated actively in the implementation (coding and testing) of the studied work items (T1, T2, T3, T4, T6, T8 and T9). Five other teams, located respectively in Italy (IT), Sweden (ST1, ST2, ST3) and the USA (UST) also participated in the analyzed work items. Eleven software architects located in Sweden participated actively supporting the execution of the work items.

4.1 Teams, team composition and work items

As aforementioned, we focused on the teams located in India. Thus, first we ordered the work items by their respective starting dates, to see whether the productivity of each team improved over time, as it is advocated by other studies on learning curves in software development teams [9], [14], [16]–[18].

To identify the work items (i.e. product customizations - PC) related to each team, we analyzed time reports made available by Ericsson. Table 1 shows the number of developers of team involved in the investigated PCs. The highlighted cells show situations where all members of a particular team participated in a PC. Note that only four teams located in India participated in at least one PC with all their members at once (T1 twice, T2 twice, T3 three times, T8 once and T9 once).

Table 1: Number of developers from each team per work item.

PC	T1	T2	T3	T4	T6	T8	T9	UST	IT	ST1	ST2	ST3	A
1	2	3	1	0	1	0	0	0	0	0	0	0	2
2	3	1	3	0	1	0	0	0	0	5	0	0	0
3	0	2	4	0	1	0	0	0	0	0	0	0	0
4	0	1	0	0	0	1	0	0	0	0	0	0	3
5	0	2	6	0	0	1	0	0	0	0	0	0	5
6	5	5	6	1	1	3	0	0	5	0	0	0	7
7	0	0	0	2	0	0	0	3	0	0	0	0	2
8	0	0	0	0	0	6	3	0	0	0	0	4	6
9	0	0	0	0	0	2	5	0	0	0	0	0	3
10	0	0	4	0	0	0	0	0	0	0	0	0	2
11	0	0	2	0	0	0	0	0	0	0	0	0	2
12	6	1	0	0	0	1	0	0	0	0	1	0	4
13	0	0	1	0	0	0	0	0	0	0	0	0	2
14	0	5	2	0	1	0	0	0	0	0	0	0	6
15	0	0	6	0	0	0	0	0	0	0	0	0	0

Table abbreviations: T1, T2, T3, T4, T6, T8, T9 – the studied teams, UST – supporting team from the USA, IT – supporting team from Italy, ST1, ST2 and ST3 – supporting teams from Sweden, A – supporting architects.

Since it was not possible to measure the productivity of each team in isolation, we analyzed each team using the overall productivity in each PC, i.e. the ratio between the complexity of the PC and the effort spent by all the teams involved in the PC.

Tables 2 to 9 show descriptive statistics related to the variables that we accounted for in our analysis of the studied teams (complexity of the work item, actual effort, productivity, cumulative experience, number of developers, team stability, involvement of other teams, distribution stability and number of architects). Only those teams that participated in at least four work items were included in the tables. Descriptive statistics about cumulative experience are not displayed because this variable has values in an incremental scale with step equals to 1.

Table 2: Descriptive statistics related to team productivity.

Team	Average	Std. dev.	Max	Min
T1	20.49	5.93	26.73	12.44
T2	29.19	11.68	12.44	51.19
T3	39.17	25.82	12.44	104.17
T6	25.61	9.24	12.44	35.75
T8	39.30	28.36	12.44	90.84

From Table 2 we see that T3 and T8 have the best average productivity, although it is hard to find any pattern based on the data presented in this table because the standard deviation is high in relation to all teams, especially for T3 and T8.

Table 3: Descriptive statistics related to the work complexity.

Team	Average	Std. dev.	Max	Min
T1	80	63.77	170	20
T2	72.5	52.51	170	15
T3	63.89	53.10	170	10
T6	77	60.17	170	20
T8	135	116.88	350	15

Table 3 shows that T8 dealt with more complex PCs on average. However, again, the standard deviations in this table are too high to indicate it as a pattern.

Table 4: Descriptive statistics related to the team actual effort.

Team	Average	Std. dev.	Max	Min
T1	5017.30	5836.68	3208	954
T2	3480.13	4279.68	13662	293
T3	2884.89	4209.88	13662	144
T6	4190.2	5372.34	13662	954
T8	4653	4600	13662	293

In Table 4 we see that T1 and T8 were involved in PCs that demanded the greatest effort to be fulfilled, although there is no huge difference between the average actual effort figures between the teams. The standard deviations are also too high to identify any patterns in this table.

Table 5: Descriptive statistics related to the number of developers.

Team	Average	Std. dev.	Max	Min
T1	4	1.83	6	2
T2	2.5	1.69	5	1
T3	3.78	1.92	6	1
T6	1	0	1	0
T8	2.33	1.97	6	1

Table 5 shows that there is no significant difference between the average number of developers of the teams involved in a task, except for T6, which despite of having six members, had only one developer participating in the investigated PCs.

Table 6: Descriptive statistics related to the team stability.

Team	Average	Std. dev.	Max	Min
T1	1	0.82	2	0
T2	2	1.51	4	0
T3	1.56	1.33	4	0
T6	0	0	0	0
T8	1.83	1.47	4	0

In Table 6 we can see that involvement of team members working together from one task to another changes often, i.e. the ‘task

teams' are unstable. Note that team stability for T6 reflects the fact that one and the same developer participated in the investigated PCs.

Table 7: Descriptive statistics related to cross-team cooperation.

Team	Average	Std. dev.	Max	Min
T1	3.50	1.73	6	2
T2	2.63	1.60	6	1
T3	1.67	2	6	0
T6	3	1.73	6	2
T8	2.17	1.94	6	1

Table 7 shows the number of teams involved in the execution of supporting a particular team. The highest number of sites involved in a single PC execution was six. Except for T3, all teams cooperated with other teams on task completion, most of which were located also in India, except the few teams from other locations, UST, IT, ST1, ST2 and ST3.

Table 8: Descriptive statistics related to the stability of cross-team cooperation.

Team	Average	Std. dev.	Max	Min
T1	1.25	0.96	2	0
T2	2	1.31	4	0
T3	1.67	1.66	5	0
T6	1.6	1.34	3	0
T8	0.8	0.84	3	0

In Table 8 we looked at the stability of the cross-team cooperation and it shows that it was often the case that the investigated teams cooperated with a different sets of team over time.

Table 9: Descriptive statistics related to the number of architects.

Team	Average	Std. dev.	Max	Min
T1	3.25	2.99	7	0
T2	3.38	2.62	7	0
T3	3	2.5	7	0
T6	3	3.32	7	0
T8	4.67	1.63	7	3

Table 9 shows that all teams have been often supported by the architects from the original development location. The maximum number of architects supporting each of the studied teams was seven, which occurred in the PC that demanded greatest effort and support by architects to be fulfilled (PC:6, see Table 1).

4.2 Productivity curves

To graphically see whether the productivity of each team was going up as they accumulated experience (autonomous learning), we plotted the relationship between cumulative experience (participation in work items – x-axis) versus overall productivity (y-axis). The charts for all teams are presented in Figure 1.

Figure 1: Team cumulative experience vs overall productivity.

Figure 1 shows that T1 improved from the first PC to the second, then showed decreased performance in the third PC and returned to a similar productivity level in the forth PC. Interestingly, lowest productivity point, the disruption point in the productivity curve, was PC:6, the one that involved the largest number of teams and architects, including also a team from another site located in Italy. Although this was not the most complex PC in terms of complexity points assigned in the expert evaluation, it was the one that demanded most effort to be carried out, which was related to the criticality of the PC for the client.

The data on T2 from in Figure 1 shows that T2 improved its performance steadily in the first four tasks, and then showed a decrease in the next two PCs, and finally continuous increase in the last two PCs. In contrast to T1, T2 curve demonstrates two periods of gradual performance improvements over time. Like T1, T2 also participated in PC:6, which is the lowest performance point on its curve.

T3 curve does not show any significant improvement over time, although this team was assigned two PCs that did not involve other teams (carried out within one team). Teams best performance was achieved in PC:13, which involved only one developer assisted by two architects from the original development location. This means that the "team's" peak performance is not related to team cooperation and group learning.

When it comes to T6, the productivity curve reflects individual performance of the one developer involved in the investigated PCs. Thus, it is not possible to related it to group learning within T6, although the curve does reflect a continuous performance improvement in the first three tasks. The disruption, as for the other teams, occurred in PC:6.

Interestingly, T8 demonstrates a decrease in performance in the second and the third tasks, in comparison with the first one, then an increase, with the peak performance for the fifth task, and a relatively low performance in the subsequent task. The forth task was PC:8, in which T8 participated as a complete team. In both forth and fifth tasks (PC:9) T8 worked only with T9 and the architects, and thus means that the cumulative experience from working together (and potentially group learning) could determine the better performance.

4.3 Identifying the role of different factors

Based on the analysis of the teams' engagement in different work items presented above was, we performed linear regression analysis for each team. The tested model included productivity as the dependent variable, cumulative experience as the dependent variable and as control variables – complexity of the work item, actual effort, productivity, cumulative experience, number of developers, team stability, cross-team cooperation, cross-site cooperation and the number of architects (see Section 3.3 for details about each variable).

We designed the following hypotheses to be analyzed herein:

- H1: The bigger the experience of a team, the better is its performance.
- H2: The more stable is a team from one work item to another, the better is its performance.
- H3: The fewer teams involved in one work item, the better is the performance.
- H4: The more stable is the cross-team cooperation setting, the better is the performance of the involved teams.
- H5: The more architects involved, the better is the performance.

H1 is about the impact autonomous learning in the development teams' productivity. H2 relates to the role of team stability in relation to team performance. H3 aims at show the relationship between performance and the number of other teams supporting a particular. H4 is about the stability of the cross-team cooperation setting, i.e. how stable is the composition of teams supporting a particular team. Finally, H5 relates to the impact of architects in team performance.

We used SPSS to run the linear regression analysis, first using the enter method and second using stepwise method. The significance of the results was evaluated at $\propto = 0.05$. Unfortunately, all the calculated models were statistically insignificant, which does not allow us to use such a results to evaluate our hypothesis. The only statistically significant model was the stepwise regression for T8, where the obtained model is presented in Equation 4 with R^2 of 0.516.

$$Productivity = 42.98 - 5.25 * Cross\_team\_cooperation \quad (4)$$

The results of the significant model demonstrate support for H3, i.e. the fewer teams involved in the execution of a work item, the better the performance.

The results of the regression analysis do not support H1, H2, H4 and H5. However, we can see some indication in the plotted charts that teams that worked in more PCs (H1), specially in a more continuous way, performed better (see Figures 1; curves of T1, T2 and T6). The curve of T8 indicates that the longer different teams work together, the better their performance may get (H4).

The results of the conducted regression analysis support only H3, i.e. the fewer teams involved, the better the performance of the involved teams.

4.4 Discussion of the results

The results of our investigation provide partial support for the research questions. In relation to RQ1, we can see that most of the investigated teams improved over time until they faced a work item that was more critical (time wise), which demanded the participation of several teams distributed in different locations. As per the result of the conducted regression analysis (H3), the learning process of the teams might be disturbed by the need for working with several other teams (PC:6). This is also indirectly supported by existing literature on global software development suggesting that projects employing many distributed teams and especially complex projects that require great amount of domain expertise are less successful [21], which was the case with PC:6. Unfortunately, based on our results we don't have an explanation for why T3 and T8 did not improved over time in the same way as compared to T1, T2 and T3.

In relation to RQ2, the conducted regression analysis did not show a significant correlation between team productivity and accumulated experience. Nevertheless, the charts plotted for T1 and T2 show that the teams' performance improved over time until a certain point, which indicates that maybe there is some relationship between productivity and accumulated experience in the investigated case. We intend to investigate further such aspect, incorporating data from work items to be conducted during 2016.

Although we did not conduct a regression analysis including learning as a dependent variable, we believe that it is possible to relate our results with group learning. The regression analysis indicates that the number of teams supporting a particular team (cross-team cooperation) is negatively correlated to team performance and thus is probably also negatively correlated with group learning. As per existing literature [3], [4], [12], to enable group learning it is mandatory to have an environment that fosters learning behaviors such as sharing of information, asking for help and talking about errors, to name a few. Team members are able to exercise these behaviors only when there is psychological safety within the team, i.e. the shared belief that team members feel safe for interpersonal risk taking [3]. In our investigation it is fair to assume that a lack of cooperative tasks for the entire team over time negatively influenced the team safety and hindered the exercise of learning behaviors, since a team or a fraction of a team often had to work with members of several other teams.

Challenges with team psychological safety and learning behaviors were probably also introduced by the changes in team members working together over time. Although our results did not support H2, the fact that the members of the studied teams were involved in different work items at the same time, and every next task brought along new cooperation mates hindered group learning and thus better performance of the teams over time. We identified that it was often the case that "temporary task teams" were composed to work on the work items. As identified by Narayanan *et al.* [18], turnover of the members in a team was found to be negatively correlated to team performance. In our case, the involved assets did not leave the company within the investigated time period. However, they were very often allocated to different **temporary task teams** that changed very often. The high turnover of work mates must have hindered the cultivation of psychological safety within the **formal teams** (determined by the formal membership).

While discussing the results of our investigation with the project responsible in Ericsson, we focused on understanding why the work items were often assigned to "ad-hoc teams" rather than to the formal teams. The project responsible reported the following:

"*The heavy ramp-up puts management in a tough spot. They cannot give features to completely new teams to develop and at the same time keep the other teams intact. The new teams will get nothing done then. So they instead need to add new people to existing teams and move out some experienced developers to build new teams instead. The "fork" approach is simply taken. This is from a practical point of view the best way to deal with the situation, but of course not optimal from a group learning perspective.*

Management people in general and in particular in our site located in India have to very often deal with work items that risk becoming delayed. So, they deal with that by putting more experienced people on the prioritized task."

Thus, the fact that the investigated work items were mainly done by ad-hoc teams is related to tight time restrictions and to the fact that there were many new developers in the teams, who had to be supported by experienced developers from other teams.

Performance is higher when assigning tasks to the most competent people (e.g. in many situations software architects). However, while cultivating individual competence in focused areas might be economically feasible short term, research suggests that higher performance can be achieved by cultivating teamwork and thus fostering group learning [2], [3]. Companies accumulating what is regarded as social capital, i.e. knowledge resources that can be obtained through teamwork and networking, are rewarded with higher performance [22]. To achieve such a thing, it is mandatory to keep team members working closely with the other members of their formal team, reducing as much as possible the fulfilment of work items by ad-hoc teams with high team member turnover.

The differences between formal teams and ad-hoc teams have probably one additional important implication. It is fair to assume

that formal teams will show higher performance than groups. However, the vast majority of performance oriented studies in software engineering are based on archival data, often coming from publicly available repositories. However, what has been studied as a "software team" is not always clear and/or accurate (formal or ad-hoc). As such, many team- and learning-related aspects in these studies might be overlooked or impossible to judge due to the missing or incomplete data.

Hackman in 1987 has defined a team as a work group that exists within the context of a larger organization and shares responsibility for a team product or service [23]. Katzenbach and Smith suggest looking into four key elements that determine a team – common commitment and purpose, performance goals, complementary skills, and mutual accountability [24]. And finally there are project teams of often temporary nature. The question is then whether it is sufficient to simply put individual programmers together and expect them to work effectively.

Furthermore, existing software engineering literature related to learning and performance does not make any distinction between individual learning within teams and group learning and more research is required to better understand the impact of the teamwork on performance [9], [14]–[16], [18], [19].

Thus, we believe that it is mandatory for software engineering researchers to clarify whether what is being studied are formal or ad-hoc teams. It is also important to distinguish between what is being learned by individuals within a team (individual learning) and what is learned by a team as an "atomic" entity (group learning), since group learning is different than the aggregation of what is learned by team members; rather, team learning is about a new understanding shared by all the team members about how the team is expected to function [12].

We intend to further investigate how team stability and cross-team cooperation stability relates to psychological safety of teams and thus to learning behaviors and team performance. To do so, we plan to conduct semi-structured interviews with software architects involved in the investigated work items, focus groups with the involved teams, and apply the instrument developed by Edmondson to measure psychological safety and learning behaviors [3]. At the same time we also consider studying individual learning curves for members of unstable formal teams or projects employing temporary task team strategy.

In this paper. we focused on the role of autonomous learning in development teams involved in large-scale projects. However, individuals and teams can also learn by means of deliberated activities, which is know as induced learning, i.e. this kind of learning is triggered by activities like trainings and investments in research & development [25]. This type of learning has been very rarely investigated by software engineering researchers [19]. Thus, we also intend to investigate what is the interplay between autonomous learning and induced learning in teams involved in large-scale software projects.

4.5 Validity threats

The threats to the validity of findings in this paper are discussed using the classification by Runeson and Höst [20].

Reliability validity threats are related to the repeatability of a study, i.e. how dependent are the research results on the researchers who conducted it [20]. We minimized this threat by involving several researchers in the design and execution of our investigation. Furthermore, our observations and findings were verified with the company representatives to avoid false interpretations. We also designed an explicit case study protocol, following the guidelines by Runeson and Höst [20].

Internal validity threats are related to factors that the researcher is unaware of or cannot control the extent of their effect in the investigated causal relationship [20]. Learning and productivity are constructs that are influenced by a myriad of factors, such as task complexity and personnel attrition, which make the identification of causal relationship between the two constructs challenging; there are several confound factors. To mitigate such threats, we accounted for factors that are reported by related literature. In addition, we performed a sanity check of the factors based on the expertise of the third author.

Construct validity threats reflect whether the measures used really represent the intended purpose of the investigation [20]. To mitigate this threat, we used data from different sources, such as archival data and interviews with the project responsible and software architects. However, it should be noted that our work has a severe limitation related to the way we dealt with teams' productivity. We were not able to isolate the productivity of each team in each related work item. Thus, it may be the case that our results would be different with we had managed to isolate the productivity of each team in each work item. However, in practice it was impossible, which is one of the lessons we learned and share in this paper. Nonetheless, we made an attempt to mitigate this threat by plotting the charts and cross checking them with the data related to other accounted factors, such as the number of involved developers in each team and the number of other involved teams. We intend to further conduct interviews with people from the management level and try to elicit at least a rough estimate about the contribution of each team to the task (the share of work from the total size expressed in complexity points).

External validity threats limit the generalization of the findings of the investigation [20]. Since we employed the case study method to conduct the investigation reported herein, i.e. our findings are strongly bounded by the context of our study. In addition, the investigated case involved only one company. To mitigate this threat, we made an attempt to detail the context of our study as much as possible, complying with corporate confidentiality concerns. We believe that the results reported herein are of particular interest for researchers and practitioners involved in large-scale distributed software projects that share a similar context, and generally to anyone studying learning in software teams.

5. LESSONS LEARNED

During the course of our research reported herein, we learned some lessons that can help other researchers when conducting studies related to learning and performance, and in particular in the context of large-scale software development projects.

First of all, we would like to emphasize that most existing research on autonomous learning in teams in general and in software development teams in particular relies on quantitative data and thus quantitative data analysis methods, such as regression analysis. However, to achieve statistically significant results, it is mandatory to have a considerable amount of data points. Green [26] argues that a sample must be as big as 104 data points + K, where K means the number of independent variables in the model. Maxwel [27] extended the work by Green, accounting for the correlation between independent variables in the model. According to him, the smallest size of a sample that is able to lead to significant regression analysis results is 191 data points. These points in software teams are related to the work items performed by the teams.

In our case, we have studied teams responsible for implementation of product customization tasks. Although we collected data that covered two years of the project, the involved teams managed to implement only fifteen unique work items, which appears to be way too few to enable statistically significant regression analysis. In practice, we believe that the product customization tasks must have been broken down to smaller tasks carried out by individual task team members, or sub-groups of members. Unfortunately, we could not identify any sub-tasks in our case, which would have the actual effort data measured, and involve cooperative work. Thus, the first two lessons learned are:

Lesson #1: *Gathering the required number of data points to perform regression analysis in large-scale projects may be hard.*

Lesson #2: *Consider what is a work item in a particular context and whether larger tasks can be broken down to smaller yet traceable and measurable sub-tasks.*

Based on the first few learnings we assumed that studying teams that have been working for a prolonged period of time (such as teams from the original development location in our case) would have provided enough data points to study group learning successfully. However, to be able to analyze whether teams are learning by doing (autonomous learning), it is necessary to detect the inception date of the teams and the sequence of each work item carried out [6].

In the attempt to define the inception of teams we also tried to understand what a team is. What we learned severily challenged our further analysis. First of all, for many teams that have existed for a long time, the team inception date was difficult to trace, and what is more important, since the team members changed and teams were also reformed, studying group learning for those teams retrospectively became meaningless. Therefore, the third lesson learned is:

Lesson #3: *Historical data from team inception maybe hard to collect if teams are unstable due to attrition or frequent team reformation.*

In our case, many work items were carried out in parallel, which made it hard to identify the experience accumulated through on going work items. Hence, the fourth lesson learned is:

Lesson #4: *Detecting the step-wise accumulation of experience may be hard when teams carry out parallel work.*

Our attempt to understand who are the members of teams working on the fifteen customization tasks led us to one of the most important findings. We identified that work items were very seldom assigned to a single formal team. Instead, we found that it was often the case that temporary ad-hoc task teams were created for the work items. This means that in practice we could not study group learning, where the group boundary was related to the formal boundaries of the teams. We also assumed that formation of task teams, in fact, might have prevented group learning, since unstable working groups were not exposed to the benefits of teamwork. Thus, the fifth lesson learned is:

Lesson #5: *Due to specific circumstances, work may be carried out by temporary "ad-hoc" teams rather than "official" teams, which should be taken into account when collecting and analyzing the data.*

To check and supplement the archival data used in our investigation, we conducted some interviews. However, as time goes by it becomes harder and harder for people to remember details regarding work items finished long time ago, maybe limiting the usefulness of the data collected through interviews. Thus, the sixth lesson learned is:

Lesson #6: *Retrospective correction or supplement of historical data may decrease data reliability or even be impossible to perform.*

When surveying existing literature from other more mature research fields (e.g. management sciences and pscychology) to define the theoretical framework of our investigation, we identified that learning and performance have been treated as independent constructs by researchers who investigate group learning and its impact on team performance. Based on our experiences and related literature we also believe that it is important to study learning and performance as distinct constructs, because:

- Performance can be affected by many factors, such as project or organizational environment changes [28], task complexity, personnel attrition, fatigue and boredom [4], or by inclusion of multiple teams as in our case, while these factors might have no influence on the team learning process.
- A team or individual may have learned something that would not necessarily lead to performance improvement [5], [12], or a team or individuals may not have the opportunity to employ what was learned [28].

Hence, the seventh and last lesson learned is:

Lesson #7: *Data collection shall focus on eliciting learning-related factors and team performance data, while data analysis shall recognize them as distinct constructs.*

6. CONCLUSIONS AND FUTURE WORK

In this paper, we report the results and lessons learned of an investigation related to group learning and performance in large-scale software development teams. We investigated two years of work performed by offshore teams recently onboarded in a large-scale project in Ericsson.

Our results show that four out of five teams demonstrated steady productivity improvements in the few first tasks, then stopped improving when confronted with a large complex work item, and in what followed showed random performance. We determined that the large complex task required the participation of a large number of teams distributed across multiple sites, which introduced performance challenges. This was partially supported by the results of a regression analysis that we conducted. However, our attempts to understand other reasons for performance variance were unsuccessful.

We believe that more research must be conducted to clarify the relationship between team stability, cross-team cooperation, psychological safety and learning behaviors in large-scale software development teams.

In continuation of our study, we intend to complement the research presented herein by:

- Clarifying whether teams are really improving performance over time by extending the number of data points through incorporation of work items to be finished until the end of 2016;

- Separating temporary task team and formal team performance by "filtering" the contribution of each team in terms of actual effort and the size of respective work expressed in complexity points;
- Investigating how team stability and continuity of cross-team cooperation relates to psychological safety of teams and thus to learning behaviors and team performance.

As a result of our preliminary analysis we also proposed seven lessons learned that may help other researchers when conducting new research related to group learning, and especially when collecting data for such research studies from large-scale software development teams.

ACKNOWLEDGMENTS

This research was partially funded by CNPq, UFPI, INES and KKS within TEDD project, grant no. 20120200. We are very thankful to all the company employees involved and being sincerely interested in our research.

REFERENCES

[1] N. B. Moe, T. Dingsøyr, and T. Dybå, "Overcoming barriers to self-management in software teams," *IEEE Softw.*, vol. 26, no. 6, pp. 20–26, 2009.

[2] A. C. Edmondson and I. M. Nembhard, "Product development and learning in project teams: The challenges are the benefits," *J. Prod. Innov. Manag.*, vol. 26, no. 2, pp. 123–138, 2009.

[3] A. Edmondson, "Psychological safety and learning behavior in work teams," *Adm. Sci. Q.*, vol. 44, no. 2, pp. 350–383, 1999.

[4] A. C. Edmondson, J. R. Dillon, and K. S. Roloff, "Three Perspectives on Team Learning Outcome Improvement, Task Mastery, and Group Process," *Acad. Manag. Ann.*, vol. 1, pp. 269–314, 2007.

[5] A. C. Edmondson, R. M. Bohmer, and G. Pisano, "Speeding Up Team Learning," *Harv. Bus. Rev.*, vol. 79, no. 9, pp. 125–132, 2001.

[6] M. J. Anzanello and F. S. Fogliatto, "Learning curve models and applications: Literature review and research directions," *Int. J. Ind. Ergon.*, vol. 41, no. 5, pp. 573–583, 2011.

[7] T.-P. Liang, C.-C. Liu, and T.-M. L. Lin, "Effect of team diversity on software project performance," *Ind. Manag. Data Syst.*, vol. 107, no. 5, p. 636653, 2007.

[8] E. Weimar, A. Nugroho, J. Visser, and A. Plaat, "Towards high performance software teamwork," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE'13*, 2013, pp. 212–215.

[9] W. F. Boh, S. Slaughter, and J. A. Espinosa, "Learning from Experience in Software Development: A Multilevel Analysis," *Manage. Sci.*, vol. 53, no. 8, pp. 1315–1331, 2007.

[10] *Oxford English Dictionary*, 7th Editio. OUP Oxford, 2012.

[11] J. Winterton, F. Delamare-Le Deist, and E. Stringfellow, *Typology of knowledge, skills and competences: clarification of the concept and prototype*. Office for Official Publications of the European Communities, 2006.

[12] P. S. Goodman and L. a. Dabbish, "Methodological Issues in Measuring Group Learning," *Small Gr. Res.*, vol. 42, no. 4, pp. 379–404, 2011.

[13] T. P. Wright, "Factors Affecting the Cost of Airplanes," *J. Aeronaut. Sci.*, vol. 3, pp. 122–128, 1936.

[14] E. Tüzün and B. Tekinerdogan, "Impact of Experience Curve on ROI in Software Product Line Engineering," *Inf. Softw. Technol.*, vol. 59, no. C, pp. 136–148, 2015.

[15] K. Spohrer, B. Gholami, and A. Heinzl, "Team Learning in Information Systems Development-A Literature Review," in *European Conference on information Systems - ECIS'12*, 2012.

[16] C. L. Huntley, "Organizational learning in open-source software projects: an analysis of debugging data," *Eng. Manag. IEEE Trans.*, vol. 50, no. 4, pp. 485–493, 2003.

[17] R. S. Huckman, B. R. Staats, and D. M. Upton, "Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services," *Manage. Sci.*, vol. 55, no. 1, pp. 85–100, 2009.

[18] S. Narayanan, S. Balasubramanian, and J. M. Swaminathan, "A Matter of Balance: Specialization, Task Variety, and Individual Learning in a Software Maintenance Environment," *Manage. Sci.*, vol. 55, no. 11, pp. 1861–1876, 2009.

[19] Y. Zorgios, O. Vlismas, and G. Venieris, "A learning curve explanatory theory for team learning valuation," *Vine*, vol. 39, no. 1, pp. 20–39, 2009.

[20] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[21] S. Darja, F. Calefato, and C. Wohlin, "Cost Savings in Global Software Engineering Where's the Evidence?," *IEEE Softw.*, vol. 32, no. 4, pp. 26–32, 2015.

[22] C. Wohlin, D. Smite, and N. B. Moe, "A general theory of software engineering: Balancing human, social and organizational capitals," *J. Syst. Softw.*, 2015.

[23] J. R. Hackman, "The design of work teams," in *Handbook of Organizational Behavior*, Prentice-Hall, 1987.

[24] J. R. Katzenbach and D. K. Smith, "The Discipline of Teams," *Harvard Bus. Rev. Bus. Rev.*, vol. 71, no. 2, pp. 111–120, 2005.

[25] J. M. Dutton and a. Thomas, "Treating Progress Functions as a Managerial Opportunity.," *Acad. Manag. Rev.*, vol. 9, no. 2, pp. 235–247, 1984.

[26] S. B. Green, "How many subjects does it take to do a regression analysis," *Multivariate Behav. Res.*, vol. 26, no. 3, pp. 499–510, 1991.

[27] S. E. Maxwell, "Sample size and multiple regression analysis.," *Psychol. Methods*, vol. 5, no. 4, pp. 434–458, 2000.

[28] J. M. Wilson, P. S. Goodman, and M. A. Cronin, "Group learning," *Acad. Manag. Rev.*, vol. 32, no. 4, pp. 1041–1059, 2007.