Introducing SCRUM into a Distributed Software Development Course

Ivana Bosnić
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
ivana.bosnic@fer.hr

Federico Ciccozzi
Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
federico.ciccozzi@mdh.se

Igor Čavrak
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
igor.cavrak@fer.hr

Elisabetta Di Nitto
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Milano, Italy
elisabetta.dinitto@polimi.it

Juraj Feljan
Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
juraj.feljan@mdh.se

Raffaela Mirandola
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Milano, Italy
raffaela.mirandola@polimi.it

ABSTRACT

The growing enactment of Global Software Engineering in industry has triggered educational institutions to perceive the importance of preparing students for distributed software development. During the last twelve years we have disclosed advantages and pitfalls of GSE to our students through our Distributed Software Development course. After running the projects according to the iterative process model for eleven years, we decided to shift to an agile development model, SCRUM. This decision was due to the growing industrial adoption of agile methods, but more importantly to increase proactiveness, sense of responsibility, and to balance the workload among the project team members. In this paper we describe the process and outcomes of our first attempt at introducing SCRUM in our distributed course.

Categories and Subject Descriptors

K.3 [**Computers and education**]: Collaborative learning, Distance learning, Computer science education

General Terms

Theory, Experimentation, Human Factors

Keywords

Global Software Engineering, distributed software development, education, agile methods, SCRUM

1. INTRODUCTION

ECSAW '15, September 07 - 11, 2015, Dubrovnik/Cavtat, Croatia

DOI: http://dx.doi.org/10.1145/2797433.2797469

Global Software Engineering (GSE) is an established practice in industry. Although GSE education was at first lagging somewhat compared to its industrial counterpart, recently an increasing number of universities have recognised the importance of preparing students for the prospect of distributed software development [1–6]. On the one hand, lower cost of development, shorter time to market, a 24-hour development cycle, and being close to the customer are among the potential benefits of employing a distributed development approach. On the other hand, unclear responsibilities and cultural clashes are some of the pitfalls. We have been conveying both good and bad sides of GSE to our students for twelve years through our Distributed Software Development course (DSD) [7]. After running the projects according to the iterative process model, we decided to align the course with a recent trend that is becoming more and more established in industry, namely agile development [8]. In general, agile methods promote a flexible software development process with focus on iterative development, continuous delivery and rapid response to changes in requirements. Agile development often stresses the importance of face-to-face communication, which, despite the existence of various communication and collaboration tools, is not as seamless and practical in a distributed setting. However, communication in general is a crucial aspect of distributed software development, and agile methods can be tailored to embrace distribution, even when it comes to communication.

Since industry provides a growing number of examples of adopting agile methods in a distributed development setting, it is important that this fact is reflected in the software engineering education. Although rising, there are not many efforts yet, which try to combine both agile methods and distributed software development in education [9]. Experiments with distributed pair programming in education were performed in [10], showing that it is feasible to develop software using distributed pair programming, and that results are comparable to those achieved with colocated or virtual teams. [11] reports the experience of a capstone course for three virtual teams, but does not provide a quantita-

tive analysis. In [12], students from three sites used agile methods and SCRUM in an experimental setting, in conjunction with a specific end-to-end tooling solution. A similar approach, but using distributed SCRUM, was presented in [13]. The same research group extended that work into a course where team members were mixed in the middle of the course, thus having the experience of both local and distributed settings [14]. Their results showed that there are no significant differences between distributed and non-distributed work, and this would suggest that SCRUM can help in mitigating several GSE issues.

In this paper we present the initial results of introducing agile methods, in particular SCRUM [15] as one of the most widely used agile practices, to our DSD course. We compare the performance of SCRUM teams and teams following the iterative process and we show that the adoption of SCRUM has introduced advantages in the quality of decisions and collaboration and, partly, in the distribution of workload. The paper is organized as follows. In Section 2 we describe the course organization and present the motivation for introducing SCRUM. In Section 3 we present the comparison of key project aspects in the old and new framework. In Section 4 we shortly discuss the results, before concluding the paper in Section 5.

2. DSD COURSE DESIGN

2.1 Overview and Original Organisation

DSD is a project-based Software Engineering course, run since 2003/04. It is carried out in a distributed manner, currently among University of Zagreb (FER), Mälardalen University (MdH) and Politecnico di Milano (POLIMI). The course goal is providing students with real-life experience of distributed work, including all phases of software development, such as defining project requirements with customers, designing system's architecture, developing, testing, documenting and presenting the project on several occasions.

Distributed student teams are usually comprised from two sides, each having 3-4 students as a small local team, being assigned to a preferred project, if possible [16]. In addition to all development-related issues, teams face several *soft-skills* challenges, like working with team members whom they never met, gaining trust, showing responsibility, using technology to communicate on a distance, communicating in a foreign language, or overcoming cultural differences (international students come from a number of countries, not just Croatia, Sweden or Italy).

Throughout the project, students and staff play several roles from work environment, such as *customers* (either teaching staff or external stakeholders - SE companies [17]) who have the general idea of the project goal, *Supervisors* (teaching staff) who monitor the project more closely, communicating with their team and helping to solve potential team issues; *Project Leaders* and *Team Leaders* (students chosen by their team members) who are "in charge" of organizing the team. *Project Leader* is located on the site where the customer resides, while *Team Leader* is organizing and leading team's remote site. There should be a strong and frequent collaboration between *Project Leader* and *Team Leader* and they should communicate well with their team members - other students. This type of organization is presented in Figure 1a. The project relies heavily on collaboration, where students use several means of communication. To document their decisions, they are required to publish Minutes-of-meeting documents after each important discussion.

At the end of the course, the evaluation is made based on several aspects. Teaching staff grades the whole project on 60 criteria grouped into *documentation quality, timeliness, presentations, product quality* and *process quality*. Project points are sent to the *Project Leader* who decides how to distribute them to team members. Such points distribution can reflect possible team problems and is only a proposal to the staff. Another part of the evaluation is a final questionnaire, which every student has to fill-in, containing several questions regarding the process, team collaboration, methods used, etc. These answers, both quantitative and qualitative, can show students' understanding of GSD.

2.2 Issues

As mentioned above the original DSD student team organization heavily relies on the mandatory roles of *Project Leader* and *Team Leader*. We have observed some negative aspects of such organization, primarily concerning the role of *Project Leader*. Those aspects are: (*i*) overload for students acting as *Project Leaders*; (*ii*) lack of initiative / proactiveness of other project members, relying on explicit leadership of *Team Leaders* and *Project Leaders*; (*iii*) dependence of project performance on the leadership /coordination skills of just one person.

Overload of *Project Leaders* can be attributed to several causes. *Project Leaders* and *Team Leaders* are, in general, more involved than the other team members in the project they are steering and feel obliged to do whatever it is needed for the project to succeed. Since *Project Leaders* are acting as communication point to the external customers, they are putting additional effort to compensate the effect of frequent requirements changes on other team members.

Lack of initiative of other team members can have two reasons: i) when team members agree with the established hierarchical organisation, they tend to act as followers and to wait for inputs from their *Project Leaders* without realising that they are sometimes unable to actually steer them; ii) in the cases the other team members do not agree with the hierarchical organisation (or would have liked to act as *Project Leaders* themselves), they tend to create problems to the *Project Leader* by not accomplishing the assigned tasks and not actively participating in common decision processes.

While the process of *Project Leader* and *Team Leader* selection is supposedly a democratic decision of local teams, in various cases it has lead to a non-optimal selection, typically because in the specific team there was no member having the right maturity to act as a real leader. Dependence of the project on just one (inexperienced) student in the role of *Project Leader* has exhibited the following downsides: *i)* the authoritative approach taken by the *Project Leader* where he/she takes decisions alone, suppresses the initiative of other team members and lowers their enthusiasm and commitment; *ii)* the failure of *Project Leader* to properly/timely assign tasks to the others leads to inefficiencies in the development process; *iii)* the failure of *Project Leader* to convey sufficient information to the remote team, effectively isolates the remote team and results in delegating to the remote team only minor tasks, while keeping main effort on her/his side.

The current DSD project model considers role assign-

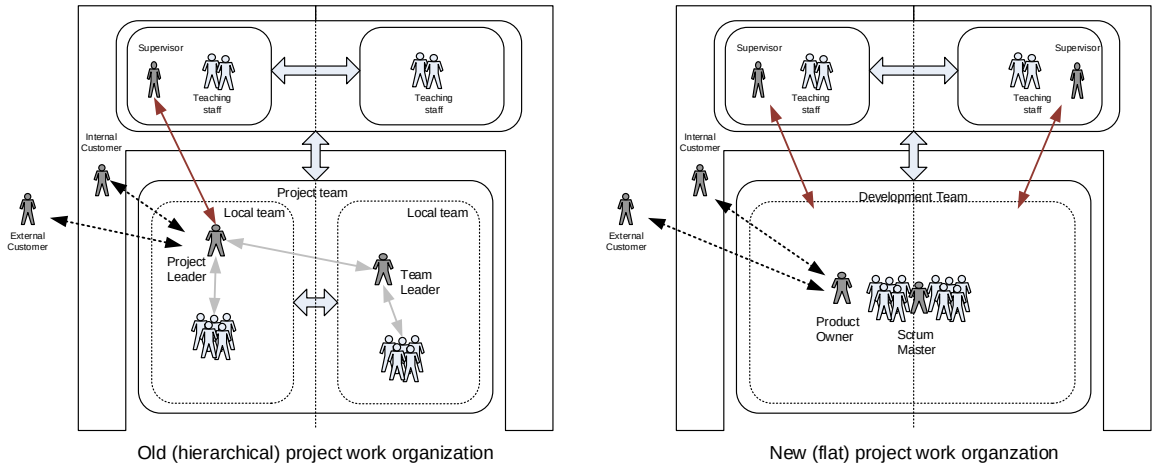

Figure 1: (a) old project organization + iterative development (b) new project organization + SCRUM

ment to be permanent. *Project Leader* and *Team Leader* roles have been reassigned on a relatively small number of projects, and approximately half of those cases required intervention of the teaching staff. In other cases, informal re-assignment occurred where more capable students took over the coordination of development, leaving the *Project Leader* taking care of administrative tasks only.

2.3 Improvements

To overcome these limitations, some changes have been considered in the project organization with the goals of (*i*) reducing the burden on students playing a role of *Project Leader* and (*ii*) involving all students in decision making as much as possible, while keeping a high quality of project work. Different possibilities have been analysed such as:

SCRUM-like project organization: SCRUM promotes self-organized teams and relatively weak team roles, such as the *SCRUM Master* and *Product Owner*. *SCRUM Master*is in charge of monitoring the process and its progress, and concerned mostly with organizational issues. *Product Owner* acts as a bridge between customers and the team, in product-related questions.

Role reassignment: during project work, the students could agree with the others to change their roles. This reassignment could encourage students to be involved in various aspects of the project work rather than focusing on tasks dedicated to a single role.

Peer-to-peer project groups: students in the team work as peers, there is neither a prescribed structure nor hierarchically defined roles, students must coordinate the effort in a democratic decision process.

Matrix organization: differentiate development roles from managerial tasks. Each student is assigned a development role and works on one or more modules, but is also a member of a "project level body" that addresses certain managerial aspects of the project (documentation, version control, testing etc.). Such roles are also responsible for defining and enforcing project-wide policies, but not for implementing specific aspects of the project. In this way, no student can be assigned only to managerial or trivial roles and all students contribute to project management thus lowering the load of the *Project Leader*.

After discussing the pro and cons of the different approaches, we have decided to experiment, in the academic year 2014 / 15, with a SCRUM-like approach for the project organization, to encourage proactiveness, awareness and communication among all team members. Figure 1b shows the new organisation. The term SCRUM-like implies the absence of prescribed SCRUM elements to be used by students, apart from sprints and two – *SCRUM Master* and Project owner – roles, but requires students to justify their decisions regarding the SCRUM elements used / not used. The students have been encouraged to assign the *SCRUM Master* role to different persons during individual sprints to avoid overload on single students and to promote the active participation of all team members.

To better observe the local teams and help them with problems related to new organization, mostly flat hierarchy and self-organization, the role of supervisor has been introduced to each local site involved in the project - thus each local team had a local supervisor as well.

3. RESULTS

To assess the impact of introduced process and organizational changes, data from last three course iterations (academic years 2012/13, 2013/14 and 2014/15) have been collected, analysed and compared (Table 1). Sources of collected and summarized data include published summary week reports, final project report documents, students final questionnaires and evaluation scores of each distributed project. Overall, we studied the material produced by 96 students.

Out of a total of 14 projects studies, 10 were conducted according to the old organizational schema and following iterative process, while the last course iteration hosted 4 projects employing new organizational schema and agile development process. The number of team members varied from five to ten, and each project involved two locations.

Besides basic project data, the table contains the project grading outcomes (overall evaulation score, as well as evaluation scores for development process and final product). The number of overall work hours is taken as a measure of invested team effort and is included in the table as well.

We identified and listed four major decision schemas used by project teams to reach important, project-level decisions:

Table 1: Analyzed DSD projects

Year	Pro–ject	Process	stu–dents	Team locations	Evaluation score project(%)	process(%)	product(%)	Work hours	Decison schema
2012/13	A	iterative	6	MdH, FER	67,7	65,1	74,4	1102	small group
2012/13	B	iterative	6	MdH, FER	68,0	68,3	71,4	1065	voting
2012/13	C	iterative	7	POLIMI, FER	94,6	93,5	93,9	1401	consensus
2012/13	D	iterative	10	MdH, FER	78,1	66,9	79,5	2150	consensus
2012/13	E	iterative	6	FER, MdH	69,0	64,8	63,0	1288	consensus
2012/13	F	iterative	6	FER, POLIMI	88,5	88,8	86,5	1708	PL decides
2012/13	G	iterative	8	POLIMI, FER	97,3	98,5	96,4	1086	voting
2013/14	H	iterative	8	MdH, FER	90,7	74,1	100,0	1822	voting
2013/14	I	iterative	7	POLIMI, MdH	95,2	94,0	96,3	2532	consensus
2013/14	J	iterative	6	FER, POLIMI	77,6	72,1	96,1	1096	small group
2014/15	K	SCRUM	5	POLIMI, MdH	85,5	87,4	91,6	566	voting
2014/15	L	SCRUM	9	MdH, POLIMI	96,4	94,5	96,6	1297	voting
2014/15	M	SCRUM	6	MdH, FER	91,7	93,1	90,8	801	consensus
2014/15	N	SCRUM	6	FER, MdH	85,0	78,4	87,5	827	consensus

Consensus type of schema required the project team to unanimously support the decision, while resorting to voting only in rare cases where consensus could not be reached within a sensible time frame. While being the most democratic and recognized by students as the most beneficial for building team cohesion, several drawbacks were identified such as inefficient meetings, increased time to reach decisions and high level of technical expertise required by all team members.

Voting was a decision schema recognized by students as more efficient than consensus but with the danger of lowering team cohesion if some team members or subgroups had been consistently outvoted. Most of the studied projects that formally employed voting as the decision schema tended to reach consensus, but without it being the primary process goal.

In *small group* decision schema, project–level decisions were made among a smaller number of project team members. We have identified two sources of such a process: (a) lack of technical expertise within a significant subset of team members, preventing them from active participation in the decision process and (b) asymmetry in project team size, where a larger number of team members is located at one location, effectively taking power and enforcing decisions on team members at other location. Both types of small group decision schema can appear as consensus or voting type at first sight, but fundamentally do not provide equal opportunities for all team members to participate in decision process.

Project Leader decides decision schema was occurring in the case were, based on argumentation of proposed solutions and discussion among project team members, the final decision was left to the *Project Leader* alone.

3.1 SCRUM Adoption

Table 2 shows how SCRUM has been adopted by the teams in the year 2014/15. Two teams organised the work in four sprints of two weeks each while the other two adopted a five sprint approach, one of those had a variable sprint duration between 1 and 3 weeks.

All teams used a product backlog where they had been including all features to develop. While SCRUM differentiates between product and sprint backlog, three out of four DSD project teams did not make a clear distinction between the two. This was mainly due to the fact that they did not have enough experience to split the backlog in clear sprint level backlogs and because they did not have a clear sprint planning phase to accomplish this splitting. One of the teams, instead, was able to focus on all sprint-related activities, from planning to retrospective, and was therefore able to build a sprint-specific backlog.

Given the distributed nature of the teams and the fact that all team members were not working at the project full-time, they did not run a daily SCRUM meeting but replaced it with meetings with various frequencies, ranging from once a week to twice a week. In one case, there was no fixed frequency for such meetings, but they were called in an ad-hoc manner. In all cases, however, students were using some asynchronous communication means to share information about the status of their work and of the backlog.

All teams adopted both the *Product Owner* and the *SCRUM Master* role. Two of them decided to rotate the roles. In one case this was due to the fact that the *SCRUM Master* felt that he was not able to properly coordinate the process. In the other case, a conflict between distributed teams occurred early in the project, requiring involvement of teaching staff and *SCRUM Master* role reassignment. Two teams also decided to assign additional fixed roles, such as specific development roles, testing role, documentation role etc., partly inheriting the old organizational style and limiting the involvement of team members to just a few project tasks.

3.2 Work Effort

We analysed the effort (declared work hours) invested by the key roles in the iterative and SCRUM organizational schemas compared to the average effort invested by their project teams. The results show (Figure 2) that the *Project Leader* role in the old organizational schema had invested a significantly larger (120% of the average team effort) amount than other team members. The *Team Leader* role also invested more time (108% on average), proving that those two key roles in the old schema posed a significant burden on the involved students. The data for the new organizational schema reveal that the role of *SCRUM Master* does have a lower average declared work hours (112% on average) compared to the *Project Leader* role, but still requires more

Table 2: Adoption of key SCRUM elements by distributed projects in 2014/15

	Sprints		SCRUM roles			SCRUM events				SCRUM artefacts	
pro–ject	sprints	dura–tion	product owner	SCRUM master	rotated SM	sprint planning	daily SCRUM	sprint review	sprint re–trospective	product backlog	sprint backlog
K	5	2 w	y	y	y	n	2-3/w	n	n	y	n
L	4	2 w	y	y	n	y	2/w	y	y	y	y
M	5	1-3 w	y	y	n	n	ad-hoc	n	n	y	n
N	4	2 w	y	y	y	n	1/w	n	n	y	n

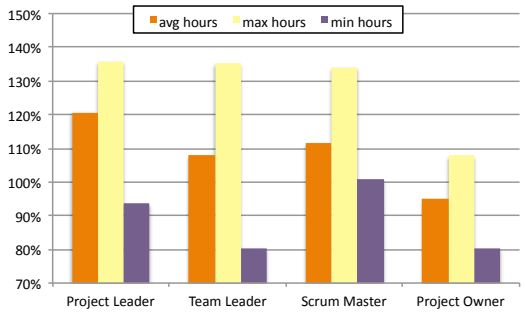

Figure 2: Average work effort of key project roles

effort than basic Development Team member role. On the other hand, the role of Project Owner does not seem comparable to *Team Leader* role – the average declared effort is even lower than the average project team effort (95%).

Observing the total effort required by the two key roles in each organizational schema, the older schema clearly required more above–the–average effort, finding that is consistent with our empirical observations. In the new schema, the task of team coordination is formally delegated to just one role, but due to the self-organizing nature of teams and the employed decision process – spreading much of the effort to all team members – the required *SCRUM Master* effort and the overall key role effort is lower. It is interesting to note, however, that the maximum effort by three roles (PL–136%, TL–135%, SM–134%) is almost the same. While understandable in the old organizational schema, such a high required effort in the new schema might indicate problems in self-organization of the specific teams, requiring the *SCRUM Master* to effectively play the role of *Project Leader*.

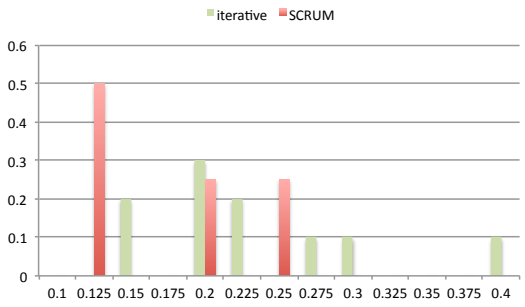

Figure 3: Variation of project team effort

We further analysed the variation of invested effort among project teams by observing the standard deviations of averaged work hours within each team. The distribution of deviations for projects using iterative and SCRUM process is presented on the histogram in Figure 3. Although the number of projects analysed precludes drawing firm conclusions based on statistically significant results, a clear trend can be observed: projects employing SCRUM process tend to yield lower variation of work hours (X-axis) among team members (in percentage, Y-axis).

3.3 Decision Process

In the old organizational model and iterative development process, all four decision schemas were present; *consensus* was the most frequently occurring one (4), followed by *voting* (3), *small group* (2) and *PL decides* (1). In the new model, equally represented *consensus* and *voting* schemas (2) were identified. It is reasonable to conclude that the new project organisation favoured more democratic decision processes, preventing explicit occurrence of *PL decides* schema. However, it generally does not prevent transformation of those two schemas into a *small group* schema, driven by different negative properties of project teams.

In order to gain more understanding of students' view on the decision process type and its properties, we have analysed the answers to the following questions in the Final Questionnaire, defining five process quality perception indicators:

Decision process: rate if the process of reaching decisions in your project was a good one or not (from 1-bad to 5-excellent).

Influence: rate and describe how much you could affect the project decisions (from 1-no influence to 5-high influence).

Role: rate and describe satisfaction with your role in the project and tasks assigned during the project (from 1-very dissatisfied to 5-very satisfied).

Insight: rate and describe awareness on other people roles and their work status (from 1-had no idea to 5-completely aware).

Overview: rate and describe information flow in your project team, awareness of the overall project status, important information and events (from 1-kept in the dark to 5-completely aware).

Figure 4 presents the averaged indicator values for old and new project organisation / development processes. All the indicator values are higher for the new project organization, and the largest progress can be observed with the insight indicator. Thus we can conclude that self-organization and lack of a priori work division have made team members more familiar with each other tasks and results. Influence and role indicators also show significant improvements; individual influence on decisions was increased and the lack of firmly defined roles allowed team members to proactively choose the work items they were more motivated for.

Figure 5 presents the analysis results with regards to the

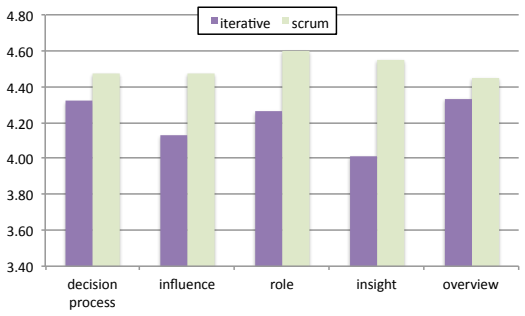

Figure 4: Influence of development process on process quality perception indicators

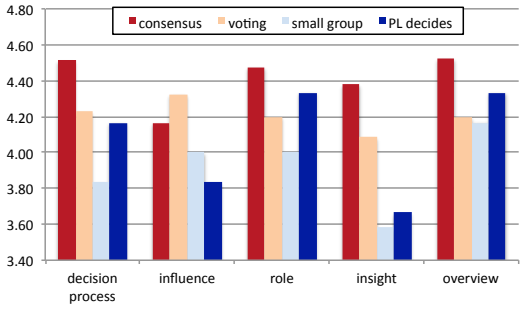

Figure 5: Influence of decision process on process quality perception indicators

adopted decision schema. *Consensus* schema has the highest valuation for all but one indicator. The students' opinion of the decision process is highest for the *Consensus* schema, lowest for the *Small group* schema, and, surprisingly, comparable for the other two schemas. Individual's influence on the decisions made is, as expected, lower for the two schemas not involving all team members. Similar ratings can be observed for the insight indicator, where those two decision schemas do not promote strong team cohesion.

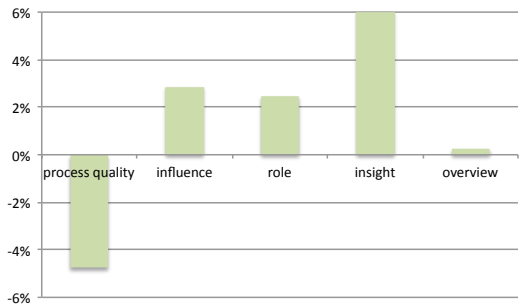

Figure 6: Influence of SCRUM on process quality perception indicators

To evaluate the impact of development process alone on process quality perception, we compared the average values of indicators between SCRUM projects and iterative projects using only *voting* and *consensus* decision schemas. Figure 6 reveals a slight decrease in the perception of decision process quality by around 5%, while influence, role satisfaction and insight indicators show slight increase. The drop in process quality perception can be attributed to inexperience of students in applying SCRUM in practice and more effort needed to self-organize the teamwork, lacking the structure the old organizational schema provided. Positive influence on other perception indicators can be explained by the encouraging effects of self-organization and the increased proactiveness required from students in such an organizational environment.

3.4 Communication and Collaboration

To assess the impact of new project organization and process on communication and collaboration, we analysed the quantitative observations of cooperation among local and distributed teams, where each student rated local and remote cooperation with a score 1–10.

Table 3: Ratings of local and remote cooperation

process	Location	
	local	remote
iterative	8,16	7,26
SCRUM	8,35	7,42
iterative without *PL decides*, *small group*	8,43	7,65

Table 3 shows slightly increased scores of both cooperation within local teams and cooperation between distributed teams for SCRUM development process, compared to iterative one. However, if only iterative projects using *voting* and *consensus* decision schema are compared to SCRUM projects, their score is even higher.

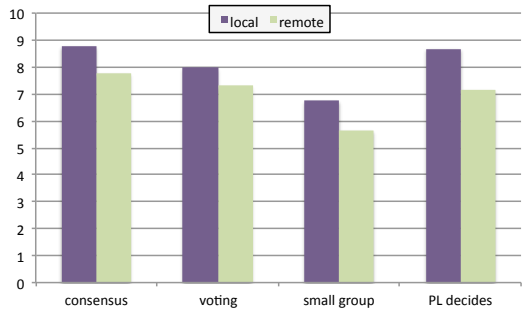

Figure 7: Impact of decision schema on local and remote cooperation

The highest cooperation ratings, concerning the decision schema used (Figure 7), were assigned to *consensus* and, surprisingly, *PL decides* schemas, while *Small group* schema was the worst one, hampering overall cooperation.

The relative difference between average ratings of local and remote cooperation tends to be the same regardless of the development process (Table 4). However, there is a significant difference between two ratings with regards to the decision schema used in the project; while *voting* schema has a slightly more uniform valuation of local and remote cooperation than *consensus*, *small group* and *PL decides* schemas exhibit larger differences, pointing towards systematic problems in collaboration.

Table 4: Difference between local and remote cooperation

process/decision schema	difference (%)
iterative	11
SCRUM	11
iterative without *PL decides*, *small group*	9
consensus	11
voting	8
small group	16
PL decides	17

From the data presented, it can be concluded that the quality of local and remote cooperation can depend on the decision schema used, but not directly on the development process itself. Both self-organizing and authoritative decision schemas can prove as efficient coordinators, however they can have different effects on the quality of local and remote cooperation: some schemas tend to even local and remote cooperation (*consensus, voting*), while others tend to emphasize related differences (*PL decides, small group*). The difference in observed quality between local and remote cooperation can be influenced up to a certain point, as the ease of local communication among local team members over remote communication with distant team members will always prevail in well–functioning distributed teams.

3.5 Project Evaluation

Another research question to answer is whether development process used and decision schema have influenced the project evaluation scores, observing the overall project score and scores related to development process and product quality.

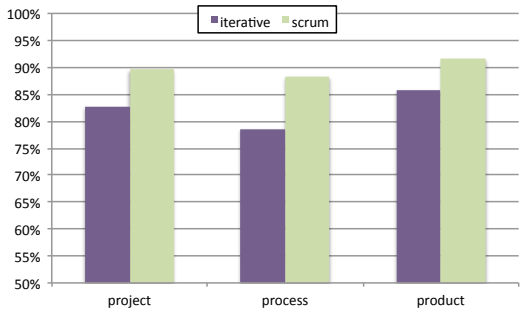

Figure 8: Impact of development process on evaluation scores

The analysis reveals that the overall project evaluation scores tend to be higher for projects employing SCRUM as the development process (Figure 8). The difference between iterative and SCRUM is even more evident in process–related scores. However, *voting* and *PL decides* decision schemas outperform *consensus* schema for all three outcomes observed (Figure 9). Those results suggest that, in spite of being the highest rated by students, *consensus* schema is not the most efficient one concerning the project and process end results. Other two schemas, while modestly or significantly reducing cohesion within project teams, gain better end results in two fundamentally different ways; *voting* schema promotes efficiency by a majority of competent team members, while *PL decides* enforces efficiency by placing most of the power in hands of one (or two) project team members. Both approaches however bring along significant risks. *Voting* schema, in case of a small number of competent students tends to transform into a *small group* schema with serious consequences for team cohesion, while performance of projects employing *PL decides* schema strongly depend on personality and competence of just one or two students.

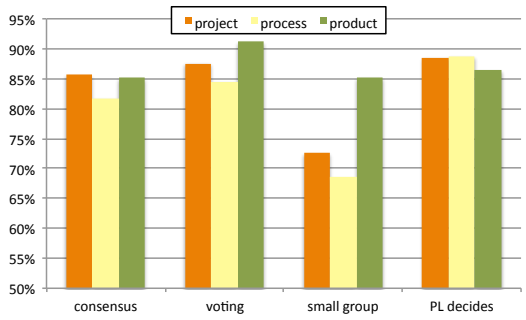

Figure 9: Impact of decision schema on evaluation scores

3.6 Threats to Validity

The analysis we have presented is clearly based on a very limited number of cases. This means that our observations do not have, by any means, a statistical validity. We have no objective way at the moment to mitigate such threat to validity, but we do think that the fact data and feedback from students reflect the expectations of instructors can be seen as a first and preliminary confirmation of results.

4. DISCUSSION

After 11 years of iterative development with hierarchical team organization, due to several reasons mostly related to the role of the Project Leader such as, e.g., overload for leading students and lack of proactiveness of other group members, we decided to introduce a SCRUM-like approach with flatter team organization. The role of "leader" together with the lack of self-(re)organization possibilities had triggered several unwanted plights in the previous course editions. With a SCRUM-like approach, our hope was that, flattening the roles by avoiding leading roles, by giving the teams the "power" of self-(re)organization with generally weaker fixed roles, would naturally lead to more democratic decisions as well as enhanced proactiveness and sense of responsibility from the whole team. So, are we there yet?

One of the reasons for flattening the team organization was to mitigate the likelihood of witnessing unfair overload for project leaders who drag the rest of the team. While experiencing a reduction of maximum effort for the somehow corresponding role of SCRUM master, on average it is still way above the rest of the team members. The decision of having a "masked" project leader was (not entirely but still) unexpected and seems to indicate the need of a clear guidance for most students. This could be either due to the fact that students with a weaker background tend to put themselves in the hands of the more skilled colleagues, but it can also be a consequence of unwillingness to proactively par-

ticipate in decision making. In either case, we need to find instruments to mitigate the tendency, since the sole introduction of a more agile development process does not seem to help all the way as we wished.

Another crucial reason for introducing a flatter organization was to trigger more democratic decision making, proactiveness, and strengthen team cohesion and awareness. We have indeed noticed a general improvement of team cohesion and awareness, especially in those cases where reorganization was not needed. In fact, while flexibility of reassigning roles seemed to be one of the keys for improvements, we noticed that it actually led to organizational problems when exploited. When it comes to decision making, not having a project leader triggered more democratic approaches in some cases. In others, the SCRUM master became, as aforementioned, a de-facto project leader, and consequently decision making was delegated to one individual again. While project and process could slightly profit from a central decision source, quality of the final product seems to grow when more democratic decisions (often through voting driven by the group of more skilled team members) are taken. This means that we need to come up with more effective ways to achieve a more equally distributed workload without undermining project and process end results, and keeping in mind the strong signal that more democratic decisions lead to better products.

5. CONCLUSIONS

In this paper we have presented a comparative analysis of the performance of student teams developing software engineering projects in a distributed setting following two different processes, iterative and SCRUM-based. The iterative process is the one we have applied in our Distributed Software Development course for eleven years, while the SCRUM-based is the one we have adopted in this academic year for the first time. The veer towards SCRUM was partly motivated by the growing industrial adoption of agile methods in software engineering, but mainly by the observation of an unbalanced workload, in the initial iterative approach, between project leaders and the rest of the teams. Our assumption was that the adoption of SCRUM would have resulted in a more even distribution of work in the teams. Moreover, we were hoping that less fixed roles would have led to increased proactiveness and sense of responsibility from the whole team. Our analysis has produced the following findings: i) we have observed a positively improved team cohesion and awareness as well as more democratic decision models leading to better final products; ii) even though the SCRUM masters showed, on average, lower declared work hours compared to project leaders, still they have been working more than the other team members. In the future, we plan to continue using SCRUM, trying to enable the students in achieving a flat organizational structure with an equal distribution of work and responsibilities.

6. ACKNOWLEDGMENTS

We express our gratitude to professors Ivica Crnković (Mälardalen University / Chalmers University) and Mario Žagar (University of Zagreb) for their pioneering work in establishing and running of the DSD course.

7. REFERENCES

[1] M. Nordio, C. Ghezzi, B. Meyer, E. Di Nitto, G. Tamburrelli, J. Tschannen, N. Aguirre, and V. Kulkarni. Teaching Software Engineering Using Globally Distributed Projects: the DOSE Course. In *Procs of CTGDSD*, Waikiki, Honolulu, HI, USA, 2011.

[2] M. Paasivaara, C. Lassenius, D. Damian, P. Räty, and A. Schröter. Teaching Students Global Software Engineering Skills Using Distributed Scrum. In *Procs of ICSE*, San Francisco, CA, USA, 2013.

[3] E. Stroulia, K. Bauer, M. Craig, K. Reid, and G. Wilson. Teaching Distributed Software Engineering with UCOSP: The Undergraduate Capstone Open-source Project. In *Procs of CTGDSD*, New York, NY, USA, 2011.

[4] S. Case, S.K. Schneider, L.J. White, S.J. Kass, K. Manning, and N. Wilde. Integrating Globally Distributed Team Projects into Software Engineering Courses. In *Procs of CTGDSD*, San Francisco, CA, USA, 2013.

[5] F. Fagerholm, N. Oza, and J. Munch. A Platform for Teaching Applied Distributed Software Development: The Ongoing Journey of the Helsinki Software Factory. In *Procs of CTGDSD*, San Francisco, CA, USA, 2013.

[6] E. Almeida, Li Dali, S. Faulk, C. Lima, Zhang Rui, D. Weiss, Jin Ying, M. Young, and Lian Yu. Teaching Globally Distributed Software Development: An Experience Report. In *Procs of CSEE&T*, Nanjing, Jiangsu, China, 2012.

[7] I. Crnković, I. Bosnić, and M. Žagar. Ten Tips to Succeed in Global Software Engineering Education. In *Procs of ICSE*, Zurich, Switzerland, 2012.

[8] R. C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, 2003.

[9] L.L. Fortaleza, T. Conte, S. Marczak, and R. Prikladnicki. Towards a GSE international teaching network: Mapping Global Software Engineering courses. In *Procs of CTGDSD*, pages 1–5, Zurich, Switzerland, 2012. IEEE.

[10] P. Baheti, L. Williams, E. Gehringer, D. Stotts, and J.M. Smith. Distributed Pair Programming: Empirical Studies and Supporting Environments. *TR02-010. University of North Carolina at Chapel Hill Dept. of Computer Science*, 2002.

[11] D.F. Rico and H.H. Sayani. Use of Agile Methods in Software Engineering Education. In *Procs of AGILE*, Chicago, IL, USA, 2009.

[12] C. Scharff, O. Gotel, and V. Kulkarni. Transitioning to Distributed Development in Students' Global Software Development Projects: The Role of Agile Methodologies and End-to-End Tooling. In *Procs of ICSEA*, Nice, France, 2010.

[13] M. Paasivaara, C. Lassenius, D. Damian, P. Räty, and A. Schröter. Teaching Students Global Software Engineering Skills Using Distributed Scrum. In *Procs of ICSE*, San Francisco, CA, USA, 2013.

[14] M. Paasivaara, K. Blincoe, C. Lassenius, D. Damian, J. Sheoran, F. Harrison, P. Chhabra, A. Yussuf, and V. Isotalo. Learning Global Agile Software Engineering Using Same-Site and Cross-Site Teams. In *Procs of ICSE JSEET*, Florence, Italy, 2015.

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

[16] I. Bosnić, I. Čavrak, M. Orlić, and M. Žagar. Picking the Right Project: Assigning Student Teams in a GSD Course. In *Procs of CSEE&T*, San Francisco, CA, USA, 2013.

[17] I. Bosnić, I. Čavrak, M. Žagar, R. Land, and I. Crnković. Customers' Role in Teaching Distributed Software Development. In *Procs of CSEE&T*, Pittsburgh, PA, USA, 2010.