In this paper, we compare two interrelated, process-centered courses in software engineering at the curriculum interface between Bachelor’s and Master’s degree. Our goal is to find out what benefits and disadvantages emerge regarding education and what could be improved.

The first of these one-semester courses (Bachelor level) makes students familiar with a software engineering project using teamwork while the latter (Master level) builds on this achieved experience and focuses on personal process improvement. Thus, these courses use different methods to achieve the objectives.

We conducted a postmortem analysis from both courses. We also integrated the results of our earlier, independent studies on software engineering projects into this analysis. For the analysis, we used a quantitative-qualitative approach.

We started by analysing working-hour metrics and grading information quantitatively. After that, we conducted a qualitative analysis on the documents to fill in the gaps, and the separated studies.

The comparison revealed anticipations as expected but also striking, interesting issues as well. It was evident that teamwork leads to good planning, probably because all information needs to be communicated to other members. On the other hand, we noted that prior work experience in software development does not correlate positively with process effectiveness, rather vice versa: those students who were eager to report their high work experience, were less capable of developing their personal work process.

In addition, notable differences in working methods exist between these two courses. Moreover, the yield value (hours per credits) differed notably. The percentage of work hours per work phase on average also varied substantially between the courses.

In the sense of improving software engineering education, we found that the ways in which students met the objectives and experienced the processes were important. We will discuss the results and educational improvements below.

Index terms: Software engineering education, Software process, Capstone software project

I. INTRODUCTION

Software engineering education needs to focus on the competencies needed in real software development. These competencies include the knowledge of the working process, teamwork, and resource estimation [6]. Modern software engineering education focuses on these skills in many ways. In fact, such software project is typically included in a software engineering curriculum [5]. In addition to teamwork-based courses, some courses focus on individual learning in so-called personal software project (PSP) instead. In PSPs, the competencies (excluding teamwork) are achieved by process metrics, self-estimation, analysis and process improvement. Learning experiences differ according to the course.

Both types of courses are available at the Department of Computer Science, University of Helsinki. This study is part of a larger study on software engineering education at the Department. In the study, we were interested to find out what makes the quality of teamwork projects vary so much. On the other hand, students get remarkably high grades on the personal software process course, in contrast to students’ and instructors’ negative feedback to the course.

II. COURSE CHARACTERISTICS

We chose two process-centered courses for our study. These courses are usually consecutive for the Department’s software engineering students. Instructors use different approaches to teach process-related issues. The Software engineering project is usually the last course of the Bachelor’s degree while the Software processes and software quality is the first or second course of the Master’s studies.

1E-mail: kimmo.simola@cs.helsinki.fi
A. The project course

Software engineering project is a practical teamwork course (a capstone project) on software engineering. It is preceded by a theory course covering a wide range of project-related issues. This way students already know the theory when they start working on the project, minimizing the problem of applying techniques while still learning them, reported by Towhidnejad and Hilburn [6]. In the rest of this paper we call this course the project course.

Typically, students take the project course in their 3rd year. They have practical knowledge of programming, documentation and small-scale design. Their theoretical backgrounds include project management and software architecture.

The project imitates a real-world software project although students cannot influence the group size, the duration of the project, or the competence of their groupmates (e.g. they cannot “buy” a capable project manager). The supervisor of the project course furnishes customers and aligns the grades for group members based on suggestions from group instructors and customers.

The course lasts 14 weeks. Although the amount of documentation is predetermined and there is only one process model available, students may decide the number of iterations (1 to 3). The process with only one iteration is in principle the classic waterfall model. Students plan their process, timetable and risk strategy at the beginning of the project and have weekly follow-up meetings. They are also recommended to prioritize the requirements.

The customer may be from the Department as well as an outside partner or a commercial company. Every team has an instructor (Faculty member of the Department) who monitors the team throughout the project but helps the team only on a general level (i.e. support in planning or implementing is not included) and gives practical tips to the project manager. Group members choose a project manager among their own group. Project managers are usually unexperienced in the sense that they have not been managers in any real-world projects. Students at the Department do not receive much formal education on conducting teamwork but they have experienced working in small groups during earlier courses.

The project course is in a sense very straightforward. However, real-world topics and customers add a lot of challenge to it. Although some versions in different institutions include some advanced organization-level tinkering (see “dirty tricks” in Dawson’s paper [2]), the Department’s course cannot be expanded in this respect with given resources.

B. The personal process course

Software processes and software quality consists of two parts: first theory and then practice. Process metrics and process improvement are important factors in both parts. In the first part students use information given in lectures and literature to prepare for their weekly exercises and the final exam. Exercises in this part do not involve programming. In the second part of the course students are given six programming tasks that are done individually. The idea is to gather metrics on the tasks and use the information to improve one’s own software development process. The first three tasks are done using the student’s current process. After these tasks students analyze their process and metrics, and come up with a process improvement, which they implement in the last three tasks. This process improvement is described in the midway report, handed in simultaneously with the third assignment. When all tasks have been accomplished, students make their final reports including the analyze of the improvement success and feedback on the course.

Although the setting is similar to the PSP (Personal software process) exercise setting introduced by Humphrey [3], significant differences exist: for example, the tasks are to be done in 7 weeks’ time and tasks themselves have been redesigned.

Each task is to be completed before moving on to the following task and the fourth task may only be started when the process improvement step is decided. Each student submits metrics from each tasks, including minutes spent on each predetermined work category, defect count, and lines of code. The work categories are planning of the task, planning of the software, programming, testing, and analysis of the work. In the rest of the paper we call this course the personal process course.

III. Study setting

For the study, we made two kind of analysis. One was based on interviews during the project course while the other was a postmortem analysis gathering data from statistics and reports the students submitted.

The purpose of the project course (Section II-A) is not to satisfy neither “serve” customers but learning experiences, teamwork, processes of producing apparently big software. In order to take this into account, we
categorized the success of the project course projects according to how well the students as a group reached the objectives. To unify our estimation process we observed a variety of issues affecting the results. In this way we got the groups apparently closer to the same “line” for comparison. For example, the complexity and size of the product interact with personal workloads. These workloads are dependent on the amount of persons in the project. Thus, a failure from the viewpoint of a customer, for example, did not affect our estimation. Among others, we recognized success factors although a small group (4 persons) could try to make software that was too big (thus often resulting in malfunction or just a partial product).

After the midpoint of the projects, we clarified group competences with a semi-structured, personal interview by the second author [4]. The goal was to find out whether the groups have worked appropriately in e.g. coordinating, teamworking, or interacting, or what kind of internal or external problems they have encountered. Such an appropriate act could be, among others, a certain focus on a problem, direction, or cooperation skills. A problem could be, for example, lack of motivation, discouraging team spirit, or inefficient process or project coordination.

IV. Results

We examined a total of 26 students from five groups of the project course: two six-person groups, two five-person groups, and one four-person group. On average, students spent 196.5 hours each on the course. Actual hours were between 126.5 and 303.5 hours, while 240 hours is the recommended maximum.

The categories were rearranged to only five: requirements engineering, design, programming, testing, and process overhead. Most hours were spent on process overhead by a clear margin (Figure 1).

We calculated a yield metric by dividing the average total hours by 10, the number of credits awarded from the course. The average project yield was 19.65 hours per credit. Project-specific yield values are shown in Figure 2 where bars 1 to 5 represent individual project group yields while the horizontal line is the average project yield.

We also devised an hours-per-grade metric by dividing students’ total work hours by grade (Figure 3). Grades range from 0 (failed) to 5 (excellent) but only grades 3, 4 and 5 were given to the students in the projects we examined. The hours-per-grade indicates how much work is needed to obtain grades. The higher the value, the more work was needed to obtain a good grade. The average value was 47.7 but the most important finding was how the value varied between groups. The value was significantly larger with small groups compared to larger groups, although grades did not differ notably.

In the personal process course we examined 29 students individually. The actual time used for each phase varied greatly among students, most significantly in programming, where the standard deviation was 15.23.
The programming was the most time-consuming phase with its 48%, which is more than double the time spent for any other phase. Figure 1 visualizes this difference.

The distribution of individual hours did not differ remarkably in relation to the average hours, even after the process improvement step was introduced. Some students had noticed the code-centered nature of their process and aimed to improve it. Only few of them succeeded in narrowing the gap between programming and other phases.

We calculated a yield metric by dividing the average total hours by 4. The divider is not the total 8 credits obtained from the course because only half of the course consists of individual programming tasks. The yield value is 9.42, notably less than half of the yields of the project course. The personal process course yield is shown in the ‘PSP’ bar in Figure 2.

While the numerical data reveal interesting issues, we explained these findings from the more qualitative-like study presented below. In the following we present the results of the interviews and analysis of students’ final reports.

A. The project course

Although interviews by the second author were done individually with the group members, in the rest of this subsection we use the term group to generalize opinions and estimations of single members.

No group had explicitly defined the criteria for success in their project. Nevertheless, the more successful the group, the more they shared opinions of the group’s common goals, when explicitly asked. In other words, members of a less successful group had more such personal goals that differed from the shared goals of the group. As expected, those groups reached the most successful results whose members were motivated and enthusiastic to learn for the future and were willing to experience new things.

Every group used a waterfall-based model with one to three iterations. Most of the interviewed groups did not have to change their requirement documents although they made some additions during design and implementation phases. Compared to the original requirements, the most typical change was reduced requirements because of the lack of time resources or underestimated workloads of projects. This requirement reduction could occur both in successful and unsuccessful groups.

Successful groups considered requirement documents to be helpful for the design phase. Unsuccessful groups, however, were unsure how to move forward. Indeed, this transition from the requirement phase to the design phase was revealed as one critical point of projects: if a group spent weeks designing general-level matters, designing details afterwards crucially delayed the schedule. Customers of some projects demanded additional requirements even after freezing and accepting the requirement document. In these cases, the timetables of too “customer-servicing” groups were delayed. Some groups were able to appeal to the acceptation of the requirement document and told their customers to stop demanding more.

![Fig. 2. Project yield compared to average yield on the process course.](image)
When asked about the driving and restraining forces for the project advance, one driving force emerged in every group among every single member: that the member relationships in the team work well. This force was independent of motivation, success, and differences in personal workloads. However, successful groups stated group pressure as a driving force: committed people wanted to do their best, especially because their peers did their best as well. In contrast, unsuccessful groups mainly thought that the problems they encountered did not cause stress because the members in these groups concentrated on saving their project from failure, not on reaching superior results. Experience from real software engineering projects was a driving force although this experience was not seen as crucial for the projects: some successful as well as some unsuccessful groups reported either experience or inexperience. The more open the atmosphere in a group, the more free the members felt to ask for help and talk about problems in their group. Communication was one crucial factor for projects. The highly communicative groups had less coordination problems and less overlapping work. Only motivated and enthusiastic groups had a tendency for open and effective communication.

Restraining forces differed substantially among members. The activity of some customers was seen as a restraining force when they delayed projects by delaying acceptance of documents or by delivering promised specifications late to the groups. If more than one person represented a customer to a project group, problems may have occurred. For example, one person could accept everything while another, without knowing decisions had been made, was critical toward every suggestion.

One critical point for the schedule was a groups’ inability to adapt in changing situations. For example, responsibility areas (such as project manager, document manager, and testing manager), committed to at the beginning of the projects, did not work well in every project. This malfunctioning was mostly the result of the inability to clarify each member’s experience or appropriate skills. Besides, tasks were given, particularly at the beginning, to persons who wanted or were interested in those tasks, not those who might have the potential to perform them. Moreover, in some groups project meetings were held regularly throughout the projects despite the amount of relevant concerns. Thus, irrelevant meetings with no relevant issues were seen as unmotivating and frustrating. Successful groups were more capable to change their methods, manners, and roles if needed. Afterwards, some groups estimated that meetings should have been held more at the beginning and less toward the end of their projects. Additionally, successful groups had a tendency to challenge their working methods and were more situational-aware than unsuccessful groups.
Inexperience from real software engineering projects was felt to be one reason for making poor design and implementation solutions. These solutions had to be re-implemented, which took a time. At the beginning of projects, some groups wanted to make too big and brilliant products. Often this willingness led to decrease of motivation during the project when it turned out everything could not be reached in the project.

The product subject as a motivator was independent of project success: groups find their source of motivation from other issues, such as personal goals or group sociality. One problem regarding unawareness of the big picture was that not everyone in groups read their design document carefully, or the documentation itself was unclear. Also, getting too extensive individual tasks in the design and implementation phases was reported only in unsuccessful groups.

B. Personal process course

We divided the final reports of the process course to the categories poor, medium and good, paying less attention to the medium one since the most valuable information appeared to be at both ends of the grading scale. The reports revealed the following weak points.

Students who scored few points had problems with the exercise assignments. They generally expected more straightforward assignments. Some reported that their personal process was not suited to the nature of the tasks. Some decided to move to another task while previous tasks contained errors and run into problems later.

One of the most prominent reason for poor performance was the lack of adaptibility of the students. Those who boasted years of work experience were not ready to adapt to new challenges. In fact, those with lesser work experience and a more pure academic background judged the assignments to be fairly easy. Experienced programmers who fared poorly were eager to note superficial flaws in the exercise setting and theoretical background.

We noted that students who did poorly did not concern themselves with their working process and tended to play down its significance. They also reported quite low motivation and did not achieve notable improvement with their process improvement step. In contrast, the good students were able to base their process improvement step on the collected metrics and reported improvement.

Students had several positive comments about the course. Generally, measuring one’s own process was judged to be a good idea and most students noted that they had in fact got to know their own process better than before. Some achieved experience in programming, something they did not have because they had only completed the mandatory programming courses.

V. Discussion

Judging by the results both courses could be improved. We came up with a list of suggestions and the most important ones are discussed here. We hoped to use the information we had to combine the strengths of both courses. Since students are only learning the development process in the project course, we judged that integrating more overhead to it would not be suitable. However, as every student is familiar with the project course when they attend the personal process course, some good practices could be inherited from the project course.

An overview of positive and negative issues in both courses are gathered into Table I. The skills of group members complemented each other. Work experience and high yields were also advances for the project course groups. Disadvantages were disconnected goals of the members, problems in communication, and small teams.

Regarding the personal process course, students’ willingness to monitor and improve their own processes typically related to motivation and the degree of method. Acquired routines were a disadvantage because experienced coders had a tendency to use only familiar methods. On the other hand, lack of coding skills was a disadvantage too.

Next, we discuss the results in more detail.

A. Project course

The personal, separate performances of members advanced projects. Unfortunately, this was undesirable because one goal of the project course was to learn to use the working methods taught and to see how efficiently these methods could work in practice. Anyway, in the following we discuss observations regarding the results (Section IV-A) of groups as a whole. Moreover, we make some recommendations on how project success could be improved and what should be taken into account in education.
TABLE I
AN OVERALL VIEW OF OUR FINDINGS IN BOTH COURSES.

<table>
<thead>
<tr>
<th>Positive issues</th>
<th>Personal process course</th>
<th>Negative issues</th>
<th>Project course</th>
<th>Personal process course</th>
</tr>
</thead>
<tbody>
<tr>
<td>peer support</td>
<td>monitoring and learning</td>
<td>disconnected goals</td>
<td></td>
<td>using only familiar methods</td>
</tr>
<tr>
<td>experience</td>
<td>motivation</td>
<td>communication</td>
<td></td>
<td>lack of coding skills</td>
</tr>
<tr>
<td>high yield</td>
<td>degree of method</td>
<td>much work in small teams</td>
<td></td>
<td>acquired routines</td>
</tr>
</tbody>
</table>

Naturally, highly motivated groups reached better results than unmotivated ones. This problem is hard to solve if students have no time or they do not care to focus on the project. On the other hand, a person’s motivation may decrease during the project if one is given too much to do alone. Learning to know the suitable size of individual tasks is essential for motivation and the success of that task. Work break structuring should be practiced before the course because too extensive tasks have a tendency to make the whole project fail. Such a damaging failure can not be repaired on time once it happens. After it has happened, motivation decreases and students’ learning experiences suffer in the course. In the name of more cohesive groups, instructors could create some group pressure inside unmotivated groups. Nevertheless, this motivation action should be done carefully because the result can be a more unmotivated group whose members blame each other and concentrate their energy on fighting instead of working together. If all the members participate in the project goals, commitment and agreement can be accomplished better. Furthermore, students should practice effective communication and coordination skills before the project. Then, during the project, students could more easily compare different working strategies and practices.

Unexperienced groups run a great risk for delayed schedules. Such delays could be avoided. Despite a groups’ awareness of the risk of underestimating workloads, these underestimates still occurred during the projects. Obviously, students would need more experience in using work estimation models or tools, or, as an easier solution, make the requirements less extensive. Also, project damages from delays should be emphasized to the groups even more. This awareness could encourage groups even to pressurize customers not to be late.

In addition, working and preventative actions in unsure circumstances should be taught more. Students should also be taught to say no whenever fundamentals are adequate for this although, in real life, customers sometimes prefer extra functioning instead of non-delayed schedules. However, in the current project course, denying to implement extra functioning is well-grounded because of pedagogic considerations. Besides, customers should be committed to the projects well enough because some serious delays were the consequence of customer actions.

Members had more shared goals in successful groups than unsuccessful ones although no group has discussed or planned any goals explicitly except the requirements. According to this observation, the results of the groups could be estimated before the end of the projects. Thus, instructors of potentially unsuccessful groups could help them to focus on relevant issues when formulating the big picture during the project. Then, the learning experiences of the students could be improved.

Obviously, more teaching is also needed for creating an open atmosphere among group members. In this way, trust between members would increase and students would be encouraged to ask for help when needed. Then, the skills of the group could be exploited more efficiently. Many restraining project forces could be eliminated with better awareness of changing circumstances. Students should be taught more dynamicity and more active monitoring and reacting in the case of change. In other words, students should be able to make readjustments whenever necessary. In real-world projects, many factors (either internal or external) change continually and failures occur if no one is able to react. However, students should practice these more complex models before the project starts because they risk to fail a project if this is the first process model they use.

As hours per grade metric indicated, students need to do more hours to get a good grade in small groups compared to bigger groups. It was clear that the group size should be at least five. Otherwise the smaller groups are at a clear disadvantage to bigger groups.

B. Personal process course
The personal process course emphasized differences in students’ personal attitudes toward learning objectives. Students were free to use whatever process they wanted. The actual data indicates that almost
everyone used a process that may be described as agile or no process at all. Initially, almost everyone thought that the tasks were straightforward and started programming early without any planning. Usually this led to problems. It was not easy to compensate for timetable problems since there were only 7 days between individual task deadlines. Unfortunately, due to a strict schedule, compensation was impossible. Some students started working on the tasks before the actual teaching period started, but encountered the same problems. In this sense the problem of using unsuitable process does not seem to be related to the actual time to work on single task. The real problem is the individual estimate on exercise difficulty.

It was soon evident that high grades from the personal process course were only a result of a simple misjudgement in the grading. Too many points were awarded for just filling out the metrics correctly. After one or two mistakes students learnt to fill the forms correctly and received full points from following forms. One could actually get more than half the total points by just submitting metrics in the requested form, which does not serve the purpose of the course. One instructor told about a student’s off-the-record comment that it was actually quite frustrating that points were not given for the actual work, which includes the development and process improvement.

The process improvements themselves appeared sometimes to miss the point, judging by the feedback from the students themselves. As we discussed in a previous section, the ones who reported to have lots of work experience were the biggest disappointments. They were most likely expecting to score easy points. They blamed the course organization for their problems, instead of finding adept ways to improve their process.

Because deadlines of the tasks were only a week apart from each other, students had only a little time to work on each assignment and plan the process improvement step. Due to this, we suggest that a certain peer support form should be implemented: the exercises themselves should be made individually, but in study groups peers could help each other to improve their processes. Some of the students had in fact hoped for such support. Even more students hoped for peer support in programming. Peer support worked well in the project course and it should be trialed in the process course as well.

Many students had problems with getting the tasks completed on time. Deadlines for the tasks were only a week apart. In addition, the midway report had to be handed in at the same time as task 3. Therefore students had to work on both assignments at the same time, which probably lead to problems in finding the real problems of the process. Students have one extra week to complete the final report. We recommend that this week should be used for planning the process improvement instead. This should give students more time to analyze their process.

The main problem on the personal process course appears to be the lack of experience on process matters. This is reflected in the quality of process improvements and students’ own comments. Although students have work experience, only few had in fact enough knowledge to assess their own working process in efficient manner. This could be a real problem with the PSP in general, since teaching process metrics and process improvement should not require a lot of work experience. There is clear evidence that the process model education at the Department does not have enough emphasis on agile models. Students are unwilling to use too complicated process models in their individual work. There is clearly need for a tested and evaluated agile process model for academic purposes.

VI. CONCLUSION

Software projects and personal software process courses give a practical base for students. The learning experiences from these courses are essential for the students in future when they interact with real-world software engineering projects.

In this study we found interesting issues on which to focus in the name of educational improvements. With these improvements, the learning experiences of students could be increased. According to our findings, the main suggestions in an educational sense to support better learning experiences are as follows.

For the software project course (called the project course in this paper), a kick-off session of forthcoming groups could be worth a trial. During the workshop, the students could practice by examining methods they have learnt and learning to know each others as persons as well as workers with useful skills. Once the actual project starts, students would have some experience and ability to make crucial decisions regarding the project: from now on, mistakes can damage the project and one has to know what to do. Some time resources are saved because of trust, openness and less uncertainty have already been created in the project group. If necessary, some credit points might be moved from the actual project to the preliminary kick-off session. This session time is taken from the actual project but this strategy could enable better learning.
experiences, one objective of the course. After all, the idea of this kind of kick-off session has similarities with the kick-off sessions of real-world projects.

For the personal software process course (called personal process course in this paper) a certain peer support should be trialed. Although students would still make the tasks individually, help could be available in peer groups. Moreover, this kind of group action would increase motivation of those who have difficulties concentrating on the course and help the students to stay on schedule. Such an informal “group pressure” was found efficient in the case of the project course whenever it existed.

These suggestions made for the two courses are based on the study presented in this paper. As they are still just suggestions, success cannot be guaranteed. Regardless, education could clearly be improved by focusing on, among others, issues that emerged in this study. After all, more research is needed to confirm the effects of the suggestions made here. However, the findings in this paper encourage us to carry on.

REFERENCES