AN ANALYSIS OF STUDENT APP IDEAS FOR IMPROVING UNIVERSITY EDUCATION

Guttorm Sindre
Department of Computer Science (IDI)
Norwegian University of Science and Technology (NTNU)
guttorm.sindre at ntnu.no, +47 7359 4479

Abstract: *In the course TDT4140 Software Engineering at NTNU in 2017 students worked in groups of 4 to propose ideas and make limited prototypes for software apps that could revolutionize university education. This paper analyses what kind of apps the students proposed, and what this may tell us about the student view of shortcomings in current university education. The main findings are that the students' ideas mainly gathered on a few types of apps: calendar and progress tracking (e.g., keeping track of lecture and seminar times, exercise deadlines, etc.), Q&A / discussion forums, lecture interaction apps, and quiz apps either for the teacher to test students or for student self-testing. The analysis indicates that many students see lack of student-teacher interaction as an important challenge. Somewhat related, many students are uncertain of their progress during the semester and want more frequent feedback. Technology is seen as a possible enabler for solving both these challenges. Towards the end, the paper also speculates how the project might have been done differently if it had been important that the students come up with more revolutionary ideas, as the resulting ideas in this project were mainly for incremental improvements of existing ways of education.*

Keywords: E-learning, apps, project-based education, software engineering

1. INTRODUCTION

The course TDT4140 Software Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, has the size of 7,5 ECTS credits. It is given to students in their fourth semester of Bachelor in Informatics (3 years), Integrated Master of Computer Science (5 years), and some related degree programs. A number of students from other degree programs take it as an elective course, then often later in the study, since most degree programs have few elective slots in the 4th semester. The class size of the course has been growing from less than 200 ten years ago to more than 400 for the Spring 2017 offering of the course (433 enrolled, 413 passed). The course has covered a typical introductory software engineering curriculum, most often using Sommerville's textbook (Sommerville, 2004) as reading material. Teaching has combined lectures and project work. In the early 2000's, the project followed a waterfall process and spanned all the four courses that the students were taking in the same semester (Sindre, Stalhane, Brataas, & Conradi, 2003). Diverse needs among study programs made this difficult to maintain, so gradually the project has shifted to being local to the Software Engineering course, as well as shifting from waterfall to more agile processes. Another major change in 2017 was to have no exam, the entire grade instead based on the project, which the students performed in teams of four persons.

Another recent change in the course is to have more open project assignments, to encourage student creativity. Previously, the idea was that the project in this course should focus on design, coding, and testing (i.e., later and more detailed development stages), as well as project management and estimation, but not

on concept inception and requirements analysis. Hence, the projects in earlier years, such as reported in (Sindre, Stalhane, et al., 2003) started out with given requirements specifications, same for all groups, which they then had to comply with through design and coding. The key argument for this was that our students have several other projects during their course of study, most notably a larger project with external stakeholders (customers), in the 6th semester for Informatics students and 7th semester for CS students. The presence of external customers make these projects ideal for focusing on early tasks such as product scoping and requirements engineering, but on the other hand, they are not able to make complete prototypes for the proposed functionality so there is relatively less focus on coding and testing. Thus, placing the 4th semester project with a focus on design, coding and testing could make sense to have these different projects address different issues. On the other hand, starting with a given requirements specification is less stimulating for creativity and increases the risk of student teams copying from each other, thus reducing motivation and learning. Allowing students to invent their own requirements could also go better with an agile process, with formulation and estimation of user stories, etc. Hence, it was decided to change to a very open project assignment. In 2016 the theme was internet of cars, and student teams could come up with any kind of app within that area. In 2017 the theme was apps / bots to revolutionize university education, again leaving to each student group to find out whatever problem in education they would like to target and whatever kind of app or bot they would propose. The department has some experience with open project assignments also before, for instance in an introductory OO programing course (Sindre, Line, & Valvåg, 2003), where the only requirements was that the delivered software be a game written in Java.

The focus of this paper is **not** on the student satisfaction or learning outcome of the course, this will be explored in a paper by other authors. Instead, this paper will analyze what kind of apps the students proposed in the 2017 offering of the project. This is interesting from an IT didactics perspective because the theme was apps / bots to revolutionize university education. Hence, proposals can both give an indication what students feel missing in today's education, and how they think e-learning might help improving things. The theme of the project in 2016 (apps for cars) is not similarly interesting from this perspective, so the analysis will only deal with student projects proposed in 2017. We have the following research questions:

- RQ1: What types of software apps and bots did the students propose?
- RQ2: What does this tell us about the students' view of shortcomings in today's university education?
- RQ3: To what extent did the project succeed in eliciting proposals that might really revolutionize education? What were the possible reasons for success / failure at getting revolutionary ideas?

The rest of the paper is structures as follows: Section 2 describes the research method, and section 3 presents the findings. Section 4 then discusses the results, whereupon section 5 provides some concluding thoughts.

2. RESEARCH METHOD

The author attended a presentation session that the students had at the end of the course, containing a stream of videos (2 min per group) with each group's pitch of their product idea. However, the pace of this session made it difficult to get a good impression, so notes from the session was not used in this research. Another motivation for not using those notes, was that the author was present in the auditorium not for the purpose of research but as one of three referees who would award a price for the best product (or in practice: best pitch), as well as give honorable mention to 10 other projects. Hence, notes taken during the pitch session was not geared towards the above research questions but rather on giving points or noting down particular details for a few projects that stood out from the rest.

Instead, the classification of student proposals for RQ1 was based on the book that was printed for the project, where each group had two pages available to present their project (Nguyen-Duc & Abrahamsson, 2017). For each group, these two pages typically contained the following information:

- Name / logo of the proposed product
- Picture of the team, with names and roles
- A persona / imagined user for the app, with picture, name and a quote indicating what the person wanted from an app. For some groups, this was a fictitious persona, for others a real lecturer or student at the NTNU.
- A couple of motivating bullet items with problems that students or teachers experienced
- A backlog of typically five user stories that the product was intending to support
- Screenshots for the app (in some cases real but more often sketched)

Grading in the course was not just made on the very brief videos and 2 page executive summaries in this book, but to a higher extent based on more detailed documentation that the students delivered during the course, such as user stories, estimates, designs, code, tests, test reports, etc. So of course, the analysis in this paper could also have inspected all these additional documents, as well as tried out the code prototypes that the groups had developed for their app ideas. However, it was chosen not to do this for the following reasons:

- With 400+ students, and thus 100+ project teams of 4 students each, the more detailed documentation would amount to thousands and thousands of pages, which would take a lot of time to analyze.
- Similarly, running prototypes would take a lot of time, especially since some of them might require platforms that the researcher did not have readily available.
- Anyway, the key to answering our research questions is not to look into all details of the students' work, nor to check the quality of their work, but rather to capture their essential ideas.

Hence, although there is a danger that the 2 page pitch in the book (Nguyen-Duc & Abrahamsson, 2017) might be inaccurate for some projects, so they might have been placed in a different category if inspecting the full documentation, it is assumed that for an overall impression it should be enough to look at the 2 page descriptions. The researcher listed these in a spreadsheet, then noted down in the next column the key functionality offered. From this analysis emerged some broader categories, and each app was then put in the category that best matched its main idea. In some cases, it was tricky to classify projects in one category or the other since the proposal included functionality from several of the identified categories (for instance, some included both lecture interaction and quiz). In such cases, apps were put in the category that appeared most heavily flagged by the students themselves in the two-page pitch description.

3. FINDINGS

3.1 Categories of app types

Many student teams had proposed apps that resembled each other, not so much in surface appearance (e.g., colours, screenshots), but in the proposed features and underlying assumptions about what to improve in education. Proposals were classified in the broad categories given in Table 1. Some proposals contained elements of several categories. For instance, some proposals combined lecture interaction and quiz, or time management and progress tracking. In such cases where a proposal did not fit neatly into one of the listed categories, it was put in the category which appeared most prominent in the student group's two page pitch of their product.

Table 1: Broad categories of proposed app types

Type of app	# Projects
Student-driven lecture interaction	28
Time-management	22
Quizzes, exercises, progress tracking	18
Questions and answers	14
Feedback (for quality assurance)	7
Other	10

The "Other" category includes various types of apps proposed by three or fewer groups each. In the following, we describe in more detail what was seen within each of the six categories indicated in Table 1.

Student-driven lecture interaction. Apps in this category mainly concerned themselves with supporting live interaction between students and teacher during a lecture. A notable distinguishing factor for the focus of these apps is interaction for which the students would take the initiative, e.g., asking questions to the teacher or expressing opinions about pace and difficulty level. Interaction where the teacher would take the initiative, such as quizzes and polls during a lecture, have instead been put in the Quiz category.

Although it might not be clearly stated, it seemed most of the student groups intuitively envisioned lectures in huge auditoria (e.g., quoting student persona who were shy to raise their hand and ask questions), not online lectures. However, much of the proposed functionality might work equally well in the context of streamed lectures. The exact functionality proposed varied from group to group but typically included some subset of the following:

- Enabling students to ask questions during the lecture.
- Enabling students to up-vote questions already asked by others.
- Enabling the teacher to choose whether to see questions live, and perhaps answer them immediately, or whether to analyse them afterwards and e.g. deal with them after the break or in the next lecture.
- Enabling the student to indicate on the fly (for instance per slide in a lecture), whether the pace of the lecture was appropriate (or too slow / too fast), whether the content was understood, etc.
- Similarly enabling the teacher to see the aggregate of this, both live in the auditorium (e.g., to be able to adjust the pace of the lecture on the fly, or take more time to explain some details that few understood), and after the lecture, for instance to improve for next year parts of the presentation that seemed to have shortcomings.

Some proposals in the category lecture interaction partly also included functionality central to other types of applications in Table 1, primarily quiz functionality (e.g., check student understanding before or during a lecture) or feedback functionality (e.g., give comments on quality of lecture). Proposals were placed in the Lecture interaction category if their main idea was live interaction during the lecture, unlike proposals where quizzes or feedback on quality after lectures were the main ideas, which would be placed in those categories.

Time-management. Apparent from persona quotes, many students find it challenging to manage their time and tasks, e.g., what lectures / seminars / labs am I supposed to attend today, what deadlines for exercise deliveries are soon coming up? Obviously this becomes increasingly complex if also including off-university activities like part-time jobs, sports, roles in student organizations, etc. Again, the exact functionality proposed varied between projects. The top 3-4 items were typically supported by most apps in this category, while the last ones were more ambitious, only to be mentioned by a few.

- Showing time and place of e.g. lectures, seminars and labs of courses the student is taking
- Possibly also including part-time jobs and spare-time activities in the calendar
- Reminding students of deadlines for compulsory work such as exercises, essays and lab reports.
- Enabling students to check off whether a lecture was actually attended, an exercise actually delivered, etc.
- Including some gamification aspects, such as giving badges or scores for performing the scheduled activities.
- Showing retrospective aggregated reports of e.g. degree of lecture attendance or exercise delivery in various courses.
- Measuring time spent on activities (e.g., doing a lecture, reading a textbook chapter), and possibly basing the scheduling of future reminders on how long time similar activities have taken you previously.

Quizzes, exercises, progress tracking. This category of apps intended to support work related to official course exercises / quizzes provided by the teacher (which could be either compulsory or voluntary to respond to), or more informal / unofficial quizzes that students could do whenever they liked just to test their progress in a course. Some apps also indicated functionality for progress tracking, e.g. estimated whether a student is on track with regards to the learning goals of the course compared to the time elapsed. Progress tracking here focuses on achievement (what has the student learnt, as indicated for instance by score on quizzes), whereas the last bullets of the time-management category rather focus on effort (i.e., did the student actually do various activities, and how much time was spent). Still, it must be admitted that some project proposals were combining time-management and progress tracking ideas, and thus tough to classify as purely one or the other. About half of these were put as time management, and half as progress tracking, depending on which of these features was most heavily flagged in the students' pitch description. Typical functionality included in this category:

- Enabling teacher (and in some cases also students) to make quizzes for the students to answer
- Making quizzes and exercises available to students in an easy way
- Supporting communication about exercises, for instance getting help from the teaching staff
- Supporting student self-testing and measurement of progress
- Automatically proposing learning resources (e.g. book chapters, videos, other exercises) to the student based on performance on a quiz / exercise

Questions and answers. From the name this could be assumed overlapping with the quiz category. However, the Q & A category contain apps that did not focus on course knowledge questions like a quiz would do, rather dealing with more generic administrative questions (e.g., Can I use the previous edition of the textbook? How do I register for the exam?). Main ideas for functionality in this category:

- Allowing students to ask questions
- Having the bot automatically answer questions for which it already knows the answer (thus giving quicker response time, and relieving the teacher of many questions)
- Forwarding to the teaching staff questions that the bot does not know the answer for
- Some: Allowing students to answer questions

Feedback. Apps in this category focused on feedback from students to teachers. Several of these apps focused on feedback about lectures, but were different from the "Student-driven lecture interaction" category in that the focus was not on interaction during the lecture, but after the lecture – and primarily on the students' quality assessment of the lecture as a whole, rather than immediate impressions or clarification questions. Typical functionality:

- Enabling students to give feedback on lectures (single lectures, or a whole series)
- Enabling students to give feedback on exercises and other learning resources
- Enabling students to give feedback on whole courses, or on combinations of courses in study programs
- Enabling the teacher to see, analyse and respond to the feedback

Some of this feedback would normally otherwise be communicated to the teacher via reference groups for the courses. Advantages of an app could be that it would be possible for more students to give feedback, that feedback would be more instant, and it would be easier for the teacher to aggregate and analyse the feedback to improve the teaching.

Other proposals. There were some project proposals that did not fit into any of the main categories discussed above, namely the 10 proposals classified as "Other" in Table 1. Of these:

- Three proposals were related to videos. One wanted to make it easier to capture and organize video of ordinary lectures. Another was more in the direction of supporting short videos (for instance programming screencasts) and the combination of these and quizzes, e.g., that different videos could be recommended for a student depending on the performance in a quiz, and each video again followed up by a new quiz. A third proposal did not concern itself with videos to be produced at the NTNU, but rather a system for supporting crowdsourced student rating of already available videos within a subject of interest. For instance, a teacher in a programming course could point to various series of videos available on YouTube dealing with how to program in that language, and then students could rate them according to their learning experience. This could then make it easier for the next students to find the best videos for learning effectiveness, and easier for the teacher to know what to recommend. Of course, video platforms like YouTube already have mechanisms for crowdsourced rating in terms of views and likes, but the proposed app was meant to be exclusive for students in a certain course, so that the rating would be isolated from the ratings of outsiders who perhaps had a different learning context.
- Three proposals were for apps to support counselling. One of these considered counselling for high school students for choosing a university degree program, i.e., filling in some preferences, strengths and weaknesses of yourself, and then getting proposals for programs matching this. Another proposal supported counselling for potential change of degree program, targeting students who might be in doubt if their current degree program was the right choice. A third proposal was for an app to support counselling in selecting a university for a half or full year stay abroad.
- Two proposals were for apps to allow students to queue for help from TA's. In some courses, for instance the huge freshman programming classes, there are sometimes problems with long waiting times in the lab to get help from a TA, or to get the TA to check and register that you have completed a compulsory exercise. The old fashioned solution with raised hands does not work well if you have to keep your hand raised for a long time, and sometimes newcomers might get help before people who had waited longer simply because they had "longer arms".
- One proposal was for collection and management of digital learning resources, and one was for a room reservation app.

Finally, it might be mentioned that one group did not propose an app at all, rather an educational board game.

3.2 Revolutionary or incremental?

Looking through the proposals it is hard to find something revolutionary. The biggest category of proposals was about lecture interaction. This is not a brand new idea, and there are systems for this already. Some

local examples are Sembly[1] and TettPå[2]. Both these have been tried in courses at the NTNU, though not necessarily in classes that the Software Engineering students of 2017 took. Some of the student apps may have proposed more advanced functionality than the abovementioned systems currently offer, but anyway the proposals in this category appear as an attempt at incremental improvement of an existing learning activity – the lecture – not as a revolution. Juhani Risku, ex-innovation director at Nokia, made a quick critical analysis of the proposed student projects (Risku, 2017) and reflected that instead of revolution a lot of cozy apps emerged, suggesting that revolutionary would have been for instance to suggest getting rid of mass lectures (with an app somehow supporting this), rather than apps for incrementally improving the lecture situation. The fact that so many student groups proposed lecture apps could have several explanations:

- Lectures is the teaching approach that many of the students have been most exposed to, although our courses have other learning activities, too. Hence, it may have been hard for them to imagine a university without lectures.
- Although lectures draw criticism for low learning effect and little interaction, students might not believe that things would be better without them. Probably, they learn at least something from lectures and would like to learn more from them. If nothing else, it is also a meeting place – you do not get so much together with your classmates just by sitting somewhere reading the textbook.

The second biggest functionality was related to time-management, where much of the proposed functionality resemble what can be found in calendar and to-do apps on smart phones, or in Outlook which the students can use via an NTNU license. A key assumption underlying most of the time-management proposals, however, is that students should not have to enter the times, places, and deadlines into the calendar themselves, rather such information should be automatically available (except for spare time activities). The natural choice, rather than a separate app, might then be the calendar functionality that comes with the LMS, as both It's Learning[3] and Blackboard[4] do offer such functionality[5]. However, it can be noted that:

- The calendar in the LMS will only give a good overview if the teachers of all courses that the student is taking in the same semester, are using the calendar function of the LMS in a consistent manner. Often, this is not the case.
- The calendar may show teacher-generated events (e.g., lectures) and deadlines (e.g. exercises), but it may be more difficult for students to add own events and deadlines, such as group-work agreed with other students, own intermediate deadlines, or spare time activities that they want to plan and see in the same calendar as study activities.
- The LMS may not have good support for mobile phones, which would be the preferred device for many students to view calendar events and reminders. Both It's Learning and Blackboard support export of the calendar to formats usable by typical smart phone apps, but for this to be effective, the student would then have to export again every time a teacher updates the LMS calendar.

The fact that more than 20% of the groups proposed apps related to time-management is a clear symptom that they are not satisfied with the LMS calendars, or the way these are currently used by teaching staff.

The other categories are not very revolutionary either. Quiz apps abound also in the commercial marketplace, though it could be noted that the proposed quiz apps were markedly different from Kahoot!,

[1] https://www.sembly.no/
[2] https://www.ntnu.no/toppundervisning/tettpaa
[3] http://files.itslearning.com/help/nb-NO/Content/Calendar/calendar.htm
[4] https://help.blackboard.com/Learn/Instructor/Courses/Calendar
[5] NTNU used It's Learning as its LMS up until the summer of 2017, then transitioning to Blackboard. Hence, the students at the point of doing the project would primarily have had experiences with It's Learning, not Blackboard.

not proposing exactly the same types of questions or time pressured competition mode, rather focusing on other types of questions and often for student self-test rather than auditorium settings. This could reflect that student were well familiar with Kahoot! and thus aware that it would not appear original to propose something very similar, while they were obviously less familiar with lecture interaction apps such as Sembly and TettPå discussed earlier. Question and answer functionality is often supported by LMS'es in the form of discussion forums, and the same goes for various types of course feedback, especially if the teacher facilitates it by eliciting such feedback. Again, proposals in this direction could be a symptom that this functionality does not appear satisfactory in the LMS'es, indeed in our big classes at the NTNU it has become more popular among teachers and students to use Piazza[6] as a discussion forum, rather than using built-in LMS functionality, partly due to scalability issues of large classes.

4. DISCUSSION

4.1 Why did the students "fail" at being revolutionary?

As seen in the previous section, it must be admitted that the proposals coming out of this project were not revolutionary, rather ideas for incremental improvement of the way teaching and learning is currently done at the NTNU. Risku concludes: (Risku, 2017)(p.6): "Students were offered a brilliant possibility to make a difference in academic learning and education, and as a result we got several cozy apps. … Revolutionizing was misunderstood as gradual, incremental, nice, and polite. Revolution rocks the boat; something has to be forgotten" His observations are quite to the point, yet it seems unfair to be very critical of the students for not being revolutionary. Of course, to some extent there could be lack of bravery or imagination among students, too, but there are several other explanations that might be more justifiable:

- The students may have adapted to the learning goals of the course, rather than to the revolutionary goal signalled in the project theme. The main criteria for grading of the project deliverables would have been e.g. quality of user stories (precise formulation, etc.), design (low coupling, high cohesion, …), code, and tests, rather than the degree of revolution. Hence, it is understandable that the students will have optimized their effort on what was assumed to affect the grade most, which was not revolution.
- The project context, with a demand to come up with a software prototype that could be demoed towards the end of the course, may have been an obstacle to revolution. If you need to come up with an app, you will go for ideas for which a prototype is within reach in the given time-frame, with fairly well understood functionality. With an idea like "improve interaction during lectures" is it fairly go get ideas for viable functionality. If the idea was instead "get rid of mass lectures" it is quite hard to determine what kind of app that should be, and what functionality it should contain. Indeed, revolutionizing education would primarily have to be about organizational and pedagogical change, and then maybe pondering what apps might support that change, rather than starting with the app first. In that light, a project to come up with ideas for revolutionizing education might have been a better fit for a course in pedagogy or organizational development, where students were not bound by the requirement to make an app to support their idea.
- The students were encouraged to seek input for product ideas, requirements and user stories from other stakeholders, typically teaching staff. Like students, teachers would also tend to think first about ways to improve their immediate day-to-day working situation, such as getting better interaction in lectures or more closely monitoring student progress through quiz apps – and less about really disruptive ideas that would radically change their jobs or in the worst case even make them obsolete.

[6] https://piazza.com/

The fact that ideas were not very revolutionary might not be a big worry, though. After all, the purpose of the course is to learn principles and practices of software engineering, which many of the students did successfully, given that most made good effort at describing their ideas and designing their prototypes. Almost all passed the course (413 of 433), and many of them with good grades.

4.2 What to do to elicit more revolutionary proposals?

If trying again to engage students in coming up with proposals for revolutionizing education, the analysis of the 2017 project indicates the following possible measures:

- *Make it more explicit that the target is radical improvements, not just incremental improvements of existing practice*, and possibly give some concrete examples of both to clarify the difference (but maybe from another domain than education, to avoid biasing the students in one certain direction).
- *Make creativity and originality the key criteria for assessment, rather than technical quality of the work.* Students will often adapt to what they believe is key to getting a good grade. However, there are also potential challenges with such an approach. In an education which has a technical focus, it would feel weird if tech quality means nothing, and being forced to implement a partial prototype of a proposal could also help the students reflect more on the proposal than if simply being allowed to come up with wild ideas without having any responsibility for realizing them.
- *Do not require that the students must implement their proposals as a software prototype?* Having to implement means that students might tend to reduce risk by only suggesting solutions that they know they can handle, thus go for something less revolutionary. Moreover, educational improvement need not be primarily about IT solutions, but rather the pedagogy or way the courses and studies are organized. Hence, assuming that the solution is IT means that some possible educational improvements will be missed. On the other hand, in an IT study program it might appear weird to have projects where IT may not have a role.
- *Do not recommend teaching staff as the primary source of input?* Teaching staff, alike students, will often think first about incremental improvement of existing practice, causing most suggestions to be on that level. Maybe international "gurus" and their writing about innovative / disruptive education might be better input in such a case.

Anyway, it is to be expected that the context and framing of a task have a huge impact on the outcome. Students asked to improve education will intuitively think about their education, courses they are currently taking or just took, and how these can be improved – thus most likely proposing something for incremental improvement. The paper (Gonzalvo, Dinh, Nguyen, Fernandez, & Zimmerman, 2016), titled "Youth re-envisioning the future of education" lets the teenagers themselves state what they would think innovative. The focus is much on game-based learning, which was natural there since that was the context of the workshop those youngsters had participated in.

4.3 What could have been revolutionary?

However, would the student projects necessarily have been more revolutionizing if they had searched international literature for ideas, rather than asking their teachers? We made a literature search with terms disruptive education, innovative education, revolutionizing education, to get some hints about what the students might have found by such a search. Some likely hits:

- MOOCs. For instance, both (Hyman, 2012) and (Gallagher & Garrett, 2013) have "disruptive" in the title, which was a common belief about MOOCs around 2012, but now seem somewhat over-hyped, at least in the short and medium term. Moreover, our students are enrolled in on-campus programs, so MOOCs might not appear of particular interest to them.
- The report (Christensen, Horn, Caldera, & Soares, 2011) also flags online learning as a potential disruptive force in education, though giving other types of advice, too. For instance, they recommend: "Not focus on degree attainment as the sole measure of success. Degrees are a proxy

for skill attainment, but they are far from a perfect one, as seen in the amount of retraining that employers do as well as the current unemployment figures. Real outcomes and real mastery—as often shown in work portfolios for example—are more important." (Christensen et al., 2011) (p.5)

- Learning analytics is also popular in research publications now, e.g. (Papamitsiou & Economides, 2014; Siemens & Long, 2011). However, these systems are large and sophisticated and need huge amounts of data – not something that would be tractable for a one-term project for a team of four students.
- Lean education (Alves, Flumerfelt, & Kahlen, 2016) could mean two things, either education *about* lean production principles, or the usage of lean principles *in education* itself, whereof the latter would be most interesting in a discussion of disrupting education. A typical example is engineering studies, which traditionally may have started with a full year of more of foundational maths and physics before students got to the more specific engineering courses where the theory would be put to use. Lean would instead recommend to teach such theory just-in-time, i.e., when it is needed in an engineering course.
- Agile education (Moreira, 2017) could also mean two different things, either education about agile processes and practices – such as agile software engineering – or usage of agile principles in education itself, for instance pertaining to design of study programs, courses, and learning objects.

Some of the above ideas are not altogether new. Distance education has been around for a long time, the point about MOOCs is just that IT has gradually enabled it to scale much more than before. Portfolios have also been around for a long time, and have been commonplace when e.g. architects, photographers and graphical designers apply for jobs. However, the most revolutionizing potential of IT in education does not necessarily lie in a current trend or hype. As an example, Keller's Personalized System of Instruction (PSI) was proposed in the 60's (Fred S Keller, 1967, 1968) and had a lot of followers in the 70's, with a number of empirical studies indicating both better learning outcomes and higher student satisfaction (Kulik, Kulik, & Cohen, 1979). Still, PSI failed to take off, instead seemingly losing popularity in the 80's, and as we all can see, the traditional "same pace, different learning" approach is still dominant in today's university courses (as well as in lower education. Several explanations have been given for this, main ones being that PSI was a too big deviation from traditional practices (Buskist, Cush, & DeGrandpre, 1991), in particular the "different pace, same learning" approach – meaning that students would finish courses at different times, made it hard to fit into universities otherwise driven by fixed semesters, fixed weekly schedules for lectures, etc. Moreover, frequent testing and feedback as required by PSI often meant heavier workloads on teaching staff. However, it has more recently been speculated that PSI could see a revival (Eyre, 2007), given the progress in learning technology. Automated solutions for frequent testing and feedback would reduce the burden of teachers otherwise resulting from PSI. IT could also mitigate the increased administrative complexity of student self-pacing.

4. CONCLUSION

Although the students did not propose anything revolutionary – e.g., no brave move away from grades to portfolios, no idea of reviving PSI or going into lean or agile education – there are several things we can learn from their proposed projects. First of all, the high number of proposals for lecture interaction systems suggest that students think that there are a lot of shortcomings with the currently dominant lecturing style, but perhaps also that they somehow feel they have some use for the lectures; otherwise they should not have suggested to improve them. **Lesson for teachers:** Whether or not taking up IT for supporting questions and answers in large lecture theatres, we as teachers should certainly think about ways of improving the interactivity of the lectures we give.

Proposals for Q&A systems also addressed the problem of interaction, though outside the lecture context. Many students feel that they receive slow response to queries, e.g., if emailing the teacher or asking

questions through the LMS. From the teacher's point of view that it is hard to respond quickly if you get a lot of inquiries, plus it is annoying to respond to things that you have already responded to before, plus maybe said in lectures and informed about on the course web page or LMS. **Lesson for teachers:** Make consistent use of information channels, within and across courses, so that students know where to find the information and thus more seldom have to ask questions.

The many proposals for time-management systems including simple calendar and reminder functionality could be somewhat surprising, as such apps already abound. The LMS (whether it would be its learning, Blackboard, Canvas or something else) also has such functionality. The conclusion to draw from this might be that existing solutions are not satisfactory for the students. The standard calendar and to-do apps on smart phones, while often offering good usability, have the problem that the student must enter all lectures, deadlines etc. manually. The LMS apps may have poorer usability, plus the challenge that the necessary information will only be there if the teachers of all courses the student is taking are using the calendar consistently. **Lesson for teachers**: If using the LMS calendar function, make consistent use of it across all courses the students are taking – otherwise the calendar they see will be incomplete and have little value.

Quiz apps is another example of a large group of proposals that is not especially revolutionary as such, although the groups proposing such apps took care to make them quite different from e.g. *Kahoot!* both in style and purpose. In many of the proposals for quiz apps, a central motivation for the students appeared to be uncertainty about where they stand. Many students are worried that their progress in a course is worse than that of their classmates or worse than recommended at that point in time. Many of the proposed quiz apps were apps for student self-testing rather than apps for lecture-related competition. For instance, some envisioned the possibility of having a readily available quiz for every small unit of learning material, so that they could immediately test whether they had learned the material or not. **Lessons for teachers:** Although it may be demanding especially with big classes, look for ways to give students frequent feedback on their progress. For learning outcomes that can be auto-tested, quiz apps may be a solution, though there is an obvious up front challenge in developing enough good questions. For topics taught in many different universities and colleges it would be an interesting idea for many teachers to collaborate to develop a joint question base, rather than each having to make a large number of questions alone. Students could also be encouraged to contribute questions, since this would be a useful learning experience in itself, and give an increased feeling of ownership to the assessment.

Although the proposed quiz apps were not particularly revolutionary, they might be one component of a bigger movement that could be more revolutionary. For instance, learning analytics (Baker & Inventado, 2014) requires frequent testing to collect the data it needs, and PSI (Fred Simmons Keller & Sherman, 1974) requires frequent and flexible testing to check if students pass the module mastery requirements. Lean (Hines & Lethbridge, 2008) and agile education (Whalen, Freeman, & Jaeger, 2008) would require frequent testing for quick, iterative incremental learning. Just like agile software development would have the programmer demonstrate running code to the customer representative in small increments, to get feedback and make corrections, the agile learner would have to demonstrate his / her competencies ever so often, and then be guided to resources either to improve the currently demonstrated competence, or to the next learning goal in the backlog. Although it is not evident from the brief project pitches, we cannot exclude that some student groups have had quite revolutionary ideas behind their proposals. On the other hand, when you know that you have to develop a partial, running prototype for something, it is risky to be very revolutionary, especially when the project counts for the full grade of the course. Hence, the context of the task made it understandable that students would rather go for incremental improvements to existing, well-understood learning activities.

REFERENCES

Alves, A. C., Flumerfelt, S., & Kahlen, F.-J. (2016). *Lean Education: An Overview of Current Issues*: Springer.

Baker, R. S., & Inventado, P. S. (2014). Educational data mining and learning analytics. In *Learning analytics* (pp. 61-75): Springer.

Buskist, W., Cush, D., & DeGrandpre, R. J. (1991). The life and times of PSI. *Journal of Behavioral Education, 1*(2), 215-234.

Christensen, C. M., Horn, M. B., Caldera, L., & Soares, L. (2011). Disrupting College: How Disruptive Innovation Can Deliver Quality and Affordability to Postsecondary Education. *Innosight Institute*.

Eyre, H. L. (2007). Keller's Personalized System of Instruction: Was it a Fleeting Fancy or is there a Revival on the Horizon? *The Behavior Analyst Today, 8*(3), 317.

Gallagher, S., & Garrett, G. (2013). Disruptive education: Technology-enabled universities. *The United States Studies Centre at the University of Sydney*.

Gonzalvo, K., Dinh, T., Nguyen, S., Fernandez, J., & Zimmerman, M. (2016). Youth re-envisioning the future of education. In *Revolutionizing Education with Digital Ink* (pp. 365-385): Springer.

Hines, P., & Lethbridge, S. (2008). New Development: Creating a Lean University. *Public Money & Management, 28*(1), 53-56. doi:10.1111/j.1467-9302.2008.00619.x

Hyman, P. (2012). In the year of disruptive education. *Communications of the ACM, 55*(12), 20-22.

Keller, F. S. (1967). Engineering personalized instruction in the classroom. *Revista Interamericana de Psicologia/Interamerican Journal of Psychology, 1*(3).

Keller, F. S. (1968). Good-bye, teacher…. *Journal of applied behavior analysis, 1*(1), 79-89.

Keller, F. S., & Sherman, J. G. (1974). *PSI, the Keller Plan Handbook: Essays on a personalized system of instruction*: WA Benjamin Advanced Book Program.

Kulik, J. A., Kulik, C.-L. C., & Cohen, P. A. (1979). A meta-analysis of outcome studies of Keller's personalized system of instruction. *American Psychologist, 34*(4), 307.

Moreira, M. E. (2017). Building a Learning Enterprise. In *The Agile Enterprise* (pp. 97-108): Springer.

Nguyen-Duc, A., & Abrahamsson, P. (Eds.). (2017). *100 Software Robots for Tomorrow's Education. Revolutionizing the learning experience with bot technologies*. Trondheim: NTNU.

Papamitsiou, Z., & Economides, A. A. (2014). Learning analytics and educational data mining in practice: A systematic literature review of empirical evidence. *Journal of Educational Technology & Society, 17*(4), 49.

Risku, J. (2017). Voice of Critique. In A. Nguyen-Duc & P. Abrahamsson (Eds.), *100 Software Robots for Tomorrow's Education* (pp. 3-6). Trondheim: NTNU.

Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. *Educause Review, 46*(5), 30-32.

Sindre, G., Line, S., & Valvåg, O. V. (2003). *Positive experiences with an open project assignment in an introductory programming course.* Paper presented at the Proceedings of the 25th International Conference on Software Engineering.

Sindre, G., Stalhane, T., Brataas, G., & Conradi, R. (2003). *The cross-course software engineering project at the NTNU: four years of experience.* Paper presented at the Software Engineering Education and Training, 2003.(CSEE&T 2003). Proceedings. 16th Conference on.

Sommerville, I. (2004). *Software Engineering*: Addison Wesley.

Whalen, R., Freeman, S., & Jaeger, B. (2008). Agile education: What we thought we knew about our classes, what we learned, and what we did about it. *Proceedings of the American Society for Engineering Education, Pittsburg, PA*.