Open Resources as the Educational Basis for a Bachelor-level Project-Based Course

Ville Isomöttönen and Tommi Kärkkäinen
Department of Mathematical Information Technology, P.O.Box 35, FI-40014, University of Jyväskylä, Jyväskylä, Finland

Keywords: Open Data, Open API, Project-Based, Innovation Ability, Creativity.

Abstract: This article presents an innovation-based course concept for project-based learning. In this course, student groups are asked to ideate and implement a software product based on Open Data and Open API releases. By emphasizing students' own product ideation, the course requires and enhances self-directed learning skills and prompts the students to see the unlimited possibilities in becoming and being a practitioner of the computing discipline. Relatedly, the course provides a tool to improve student self-efficacy, as the students, coached through challenges, come to know that they are able to produce software using various open interfaces.

1 INTRODUCTION

Along with the decades-long project education in computer science and software engineering (Tomayko, 1998), various options for project-based courses have been conceptualized (Fincher et al., 2001; Clear et al., 2001; Burge and Gannod, 2009). These taxonomic works discuss aspects such as individual vs group projects, the problem scale and curricular levels of projects, and levels of realism, just to name a few. Recent course conceptualizations also include team work in multi-cultural (Pears and Daniels, 2010) and multi-disciplinary (Burnell et al., 2003) settings. In general, project education reflects educational thinking that 'function drives the form' (Fincher et al., 2001), i.e., the educational goals for a project can be reached by selecting applicable course attributes.

The present paper describes and reviews a project course concept where *students innovate* a software product. The theme of this bachelor level course is 'Open Data and Open APIs (Application Programming Interfaces)', which are on the increase among societal interfaces. This theme was initially regarded as topical and thereby potentially interesting to the students. Later, the course coordinators began to pay attention to the innovation ability of the students, and the course is now considered to be a means to illustrate to the students their potential as practitioners of computer science and software engineering. The course under study is located at the University of Jyväskylä, Finland.

Several scholars agree that there is a need to include innovation ability into higher education curricula, and research on this topic is ongoing. Yunfei and Qin (2009) posit that innovation ability is a natural attribute of every person and that it can be cultivated by education. Fila et al. (2012) discovered that students associated innovation mostly with creativity, while their study also calls attention to other dimensions such as desirability and feasibility. The study by Yang and Cheng (2010) in turn suggests that interaction between individuals across project teams enhances creativity. In the present paper, the specific research aim is to identify and discuss challenges that students, who are expected to be inspired by the opportunity for creative software development in groups, arrive at a course fully based on the students' own ideation in a setting where the work is not done for the teacher or an external client or sponsor.

The present work is part of the diagnostic phase of an action research project. Action research, coined by Lewin (1946), addresses local concerns to induce a social change and improved understanding of the social situation. Educational action research is typically conducted by or with authentic participants; through such insider roles, action research is considered critical and practical (Carr and Kemmis, 1986). The need to add a project course at the bachelor level was identified by educators who observed a multitude of learning challenges among students during a master's level authentic customer project in the local curriculum. Adding the second/third-year practical course was an attempt to smooth the students' study path to

Isomöttönen V. and Kärkkäinen T..
Open Resources as the Educational Basis for a Bachelor-level Project-Based Course.
DOI: 10.5220/0005432100460056
In *Proceedings of the 7th International Conference on Computer Supported Education* (CSEDU-2015), pages 46-56
ISBN: 978-989-758-108-3

the customer collaboration project. The Open Data and APIs provided a realistic setting for this early curriculum course, and thereby introduced a fruitful educational environment. So far we have run the course twice and are identifying the main educational achievements and issues to be addressed. This paper is presented in a lessons learned-style, much like a self-ethnography wherein authentic participants investigate observations that have raised their research interest (Kaihlavirta et al., 2015)—the action research contextualization will be emphasized in future work.

2 BEING 'OPEN' IN THE DIGITAL ERA

When a resource is open, it is available for others to use. It is freely accessible, usable, modifiable, and redistributable as-is or as a derived work. Similarly, an open activity or process can be followed and affected by others during its execution. However, when the attribute of openness is attached to different things, we end up with multifaceted and varying conceptualizations, such as open data, open source software, open service, open innovation, open education, open information society, open government, open science and research, etc. (see, e.g., (Jaakkola et al., 2014b)). To be known as open, a resource or outcome of a process is typically attached with an open license. For a piece of software, this can mean complete freedom, even to close the derivative works, as with the permissive MIT license. Another well-known alternative is to enforce the derivative works to be similarly available as the original source code, as with the copyleft GPL licenses (see Tuunanen et al. (2009) and references therein).

Open *data* is accessible and usable for further processing and refinement. According to Domingo et al. (2013): 'Open data is the concept that defines the publication of government or private company data without copyright restrictions. The data should be formatted so that citizens can reuse it at their discretion to create new, innovative services or applications.' However, the overall rights for even a raw data resource should be governed by a content license, such as one from the creative commons family[1]. Moreover, open (and sometimes big) data is often not enough, but one should be able to turn it into interesting patterns, models, and visualizations (Han et al., 2011) that provide answers to some specific questions of true value (Hand, 2013). In the Finnish national context, the development of the open data movement has been summarized by Jaakkola et al. (2014a).

Recently, many tools and methods for utilizing open data have been proposed: visual exploration of open data (Otjacques et al., 2012), a flexible environment for Web data integration (Castanier et al., 2013), a domain-specific language (DSL) for open data visualizations over the Web (Morales-Chaparro et al., 2014), and a high level library to help developers build safe mashups over APIs in HTML5 (Telikicherla and Choppella, 2014), to mention a few. Open data can also be utilized in existing or novel games, as suggested by Friberger and Togelius (2012). Concerning the Semantic Web tools, the integrated knowledge base of open data is referred to as the Linked Open Data (LOD) (see Auer et al. (2014) and articles therein). Arguably, such massive data sources increase complexity of applications but also their potential benefits.

Activities to create, publish, and utilize open data are becoming more and more popular. In many cases, availability of open data can be linked to an open innovation process fostering collaboration between private companies (especially in the creative industry), governmental or public actors, citizens, and academia (scholars and students), once again to create new products and services through purposive inflows and outflows of existing or newly created knowledge (e.g., (Huizingh, 2011; Conradie et al., 2012)). The existence of a local co-creative platform can be an important precondition for publishing some existing data, even if to reach the actual innovative outcomes requires continuous efforts and allocation of time from all the participants.

It should be noted that to be able to be innovative may require a basic understanding of the actual term innovation, which, for undergraduate students, could be challenging (Fila et al., 2012). Similarly, as argued by Dahlander and Gann (2010), scientific inquiries related to open innovation reveal a lack of clarity regarding the term's meaning, especially in the business context. However, the paradigm of open innovation has been proposed by Chesbrough (2003), with the encouragement of knowledge inflow and outflow to ensure that tasks are completed efficiently and effectively. He sees open innovation as the key to developing novel products and services in the modern business world. He proposes six principles to achieve open innovation, two of which are relevant here: 1) work with smart people inside and outside the company (university students of CS should qualify), and 2) a firm (or a student team) does not have to originate the research (or data and software) to profit from it.

The innovation process and outcome can be closed or open. For instance, an open source software com-

[1] https://creativecommons.org/licenses/

ponent can come from a private software company taking part in the open source development. Similarly, the user-centric design process of, for example, texture for a new curtain model, might end up as a new product sold under a (closed) trademark. As educators in the digital era, we should encourage students to make the outcomes of their Open Data/API projects open (Huizingh, 2011). So far, the main focus of the course studied in this paper has been inbound open innovation (i.e., utilization of existing open resources to produce an own software deliverable).

The present section contextualized our study in reference to the use of open resources, while we are aware of the other kind of literature describing various options for implementing a project course (e.g. (Fincher et al., 2001; Clear et al., 2001)). The peculiarity of the course studied here is the high degree of student responsibility expected in the early curriculum. Comparison between this aspect and early curriculum project models in the literature merits a separate study.

3 THE COURSE

3.1 Workload, Learning Objectives, and Facilities

The project course spans 12 weeks and students are rewarded with 4 to 6 ECTS credits: the intention is that each student will earn 5 credits. The course is intensive and challenging; the software product prototypes illustrating at least a proof of concept are ideated and implemented in the given time frame through newly formed groups.

The initial learning objectives were to introduce and conceptualize group processes and software process issues to the students through realistic project work at the bachelor level. After two course iterations, these objectives have been complemented with the ones emerging from the innovation-based course concept; that is, prompting self-directed study processes among the students and improving student self-efficacy. These latter kinds of learning objectives are the main focus in the present paper.

Related to the possibility of improving self-efficacy, ideating and producing software from scratch sets a technical learning goal for the students. Pre-requisite courses include CS1 and CS2, and all the students are thus expected to have basic programming skills. In our CS1 course, students program a graphical game using a particular library. A few students have taken a web programming course and our CS2 has an elective continuation part briefly introducing web programming. The introductory programming courses (CS1 and CS2) are the minimum programming background for the project, however. From this premise, technical support by which students' problem situations are reviewed and resolved, often by directing students to useful learning resources and software component resources, is an important facilitating element of the course (see Section 3.2).

Each group is provided with a lockable work room equipped with personal computers for each student to support students' realistic and autonomous work. The course thus differs from studio-based learning environments where students' work is guided through fixed practice sessions (see, e.g. Suri (2007)). The faculty's PC support is available to the students, and they are granted local administrative rights to install and configure software independently. Typically, programming code is managed in a version control system, and each student pulling from the remote repository can run the needed server applications and so on in the local computer (localhost). The intended group size is four students, though, due to course population, a few groups have also comprised 3 or 5 students.

3.2 Teaching Resources

The course is taught by two supervisors. A departmental teacher (the first author) supports students with group work issues and software process issues, while a senior student works as a technical supervisor. The senior student recruitment is based on a strong personal interest in the course topic (creative software development based on the open theme).

Absorbing the overall architectural idea of how to work with APIs and integrate data and software components into new products is important for the students. It is this competence that the technical supervisor of the course needs to possess. Many API releases are supported with an open source wrapper code. Then, what remains as a task of the programmer, in order to receive the data needed, is studying, integrating, and potentially modifying the wrapper. Similarly, parsing various data formats used in open data and API context, such as JSON, is often an effortless task, while open source components also tend to exist for parsing less popular data formats such as PC-Axis. In light of these examples, students are to be informed of the *conventions* and existing *technological possibilities* of small-scale open source (web) development. The technical supervisor, attending all the per group supervision sessions, importantly facilitates the students' technology adoption process.

The aim is to develop a course climate where the personnel reflect strong interests in their specific expertise areas, to inspire students to develop professional interest in the course topics. In our experience, the course has been manageable for two persons with up to 7–8 groups and a maximum of four students in each.

3.3 Joint Course Events

The course structure is displayed in Figure 1. The top line describes events concerning all the students. The course begins with a start-up meeting during which the course idea, course events, and documentation are explained to the students. This meeting also provides students an introduction to Open Data and APIs, by giving examples of the related applications found on the Web and a comprehensive list open resources. Both national and international (EU) open data and API links are listed, and known services with API releases, such as Spotify, Twitter, Bibsonomy, and Trello, are noted. To enable students to picture about the scope of their project, exemplary resources are discussed through speculative examples ("Trello could be complemented with a component that first exports users' work hours markings and then imports a sum of them with visualization"). Students are also guided to independently seek other fresh data releases based on their own interests. They are also told that HTML scraping could be needed to receive data within the interesting topic. Students are grouped by the teacher at the start-up meeting. Because the course pedagogically focuses on group work issues, the aim is to group student who have not worked together previously. This way each group's dynamic has a fresh and equal start.

The next shared event is the project lecture where group work concepts, including norms, statuses, roles (Brown, 1988), and justice in group work (Isomöttönen, 2014), are discussed. The topic of software processes is also included. Here, discipline in the software development is carefully explained and linked to project safety. A particular emphasis is also given to iterative software process, which has been found to fit realistic student projects (Brown, 2000; Isomöttönen, 2011). On the third week, a shared session is arranged where the groups describe the project topics they have ideated. This session is intended to reinforce the innovation theme across groups and share considerations on the use of data resources and the designs of software products. Near mid-course, an expert lecture on open source software licenses is provided to increase student awareness on the licensing issues and enable them to agree on the licensing of the group products.

At the end of the course, groups present their software products during an 'open day' for the department's personnel and other students. In place of traditional presentations, students present in their project rooms and allow the audience to try the software products and discuss design and implementation. This way of presenting fosters dialog among the attendees.

3.4 Minimal but Important Documentation

The mid-line in Figure 1 describes the documentation required during the course, which is minimal. At the beginning of the course, student groups prepare short synopses for 1) what they have agreed as their project topic and 2) how they are going to manage their work. These are intended to help the students be aware and take responsibility. In the synopsis on the project topic, student groups provide an abstract for the topic but also address the important feasibility questions below:

- further ideas and visions,
- target group (utility aspect),
- licenses and terms of conditions of the intended data usage,
- data formats,
- technical environment (programming languages, platforms, libraries, etc.),
- licensing of the product (under what open source license the product is released), and
- boundaries of the project, i.e., what parts of the ideated project are most relevant and to be achieved during the project.

These questions are reviewed in per group supervision sessions. A potential gray zone in data usage is thus avoided since the students are guided to find out and comment on the conditions of their data usage. Should there be any open questions, these will be naturally raised since the synopsis questions are reviewed. Because student work is by default owned by the students, they naturally decide on the licensing of the products. In the synopsis, groups make an initial agreement on an open source license for their product. The licensing question is later supported by an expert lecture (see Figure 1), after which the students are prepared to make an informed decision, taking into account the licenses of the used components. In the course based on the students' own ideation, the university does not require any transfer of rights.

Given the emphasis on the students' own product ideation, the projects are based on the students'

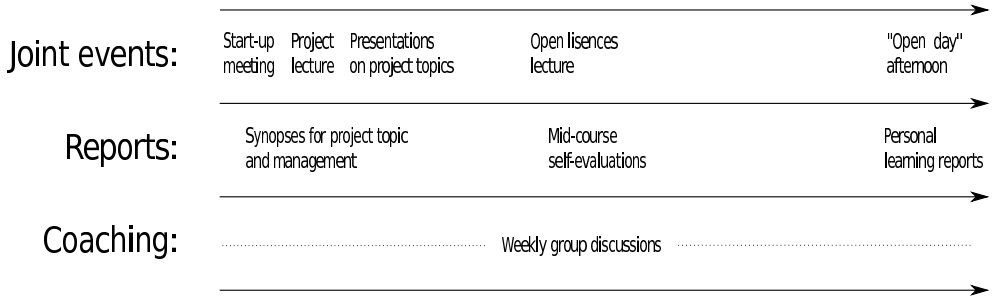

Figure 1: Course events and tasks during 12-week creative software development effort.

proposals with no initial inclusion/exclusion criteria on APIs, data, or application type. For instance, the product can be a web service or a game, as well as a desktop application, and we have also suggested the use of Arduinos and Raspberry Pis in the context of open resources. Students must nevertheless consider a target group, which implies that the projects become realistic as compared to programming exercises with no user interest. The open theme enables small-scale products starting from ones making single data source utilizable through data manipulation and visualization (cf. small tools for data journalism), while often the ideated products are larger. For this reason, student groups prioritize and define boundaries for what they aim to achieve during the project (see synopsis items above). We want to prompt students to be creative, while the synopsis document and weekly supervision discussions help groups set their goals realistically during the project.

At mid-course, self-evaluations on group work and software process are conducted, each student completing a survey form. The teacher inspects the evaluations and raises their main points during a group discussion session; the group situation revealed by the evaluations is openly discussed. When the projects are complete, each student prepares a personal learning report, reflecting on group work and software process issues in light of both lectured theory and conceptualizations that emerged during the project. The teacher gives a written response to each learning report to enhance student learning.

The inclusion of very little documentation means that all tasks are substance tasks that advance the actual ideation and software product development. This makes it almost impossible for students to limit their involvement to completing some secondary tasks; potential 'passenger' roles become visible and can be raised as group issues. For project management, the students use VCSs (with no exceptions, this has been Git) and project management software, such as Trello. It should be noted that limited documentation does not indicate low teacher workload, as sensitive group discussions (Section 3.5) are very challenging for supervisors and require preparation and continuous reflection.

3.5 Emphasis on Dialog Through Group Discussions

The bottom line in Figure 1 illustrates that teaching this course means coaching through group discussions. Thus, a discussion session attended by the course teacher, the technical supervisor, and the student group, is arranged each week for all the groups. The group situation, software process, and various issues in product ideation and implementation are discussed in these sessions. Informal discussions were considered suitable for the creativity-based course, and are a tool to introduce theory in the presence of authentic practical work; emergent problems are contemplated in terms of theory. During a particular session, the written self-evaluations provide the basis of discussion, which aims to guarantee that all the students are heard through personal writings during the project.

In this short course, the main learning objectives are introduced at the beginning through lectures and are then intensively addressed in the group discussion sessions throughout the course in the context of actual individual projects. The main pedagogic principle is based on the realist epistemology (see Bhaskar, 1978; Moore, 2000). Thus, it is based on an assumption that there are important objectifications that can explain to the students their project successes and challenges, and that these objectifications must be raised during the project to foster conceptual understandings among the students. It is important to note that the innovation-based open-themed projects have provided a good forum to introduce truly authentic project work where group issues, for instance, emerge naturally and can be conceptualized to the students realistically. Taking group work as the example case, the pedagogy of the course was described in detail in another study (Isomöttönen, 2014). As mentioned, in-

stead of group processes and software process issues, this paper focuses on educational achievements and challenges arising from the course concept.

3.6 Pass/Fail gGrading

Considering the sensitive issues (e.g., justice in group work) discussed during the course and the aim to promote the students' innovation ability, grading is pass/fail. Without competitive or external pressure of numeric grading, students are prompted to overcome their difficulties in adopting self-directed, creative, learning processes and to fully focus on conceptualizations that explain their group experiences. Positive experiences with a project-based course without numeric grading have been reported by Daniels et al. (2004). Promoting student interest in course content instead of 'just passing the course' is also in line with the Klug's (1976) work. He linked (numeric) grading with a degree system that does not necessarily match with the learner's personal intellectual development.

In our course, students track their work hours. However, passing the course is not based on quantitative inspection of student work hours or amount of programming code. These attributes approximate student role in the group, but they do not explain the effect of group situation on the student's possibility to participate. The passing is based on active participation, which is fairly easy for the teacher to interpret with the selected course arrangements: the frequent meetings between the supervisors and groups and the related discussions on group processes. During these discussional sessions, reasons for low participation are objectively addressed in terms of group work processes, following that the students themselves become aware if their difficulties result from group work or if they are truly not participating in the course. In the first case, various solutions for improving the group situation are sought for. For instance, by improving intra-group communication the division of work could be improved to match the skill levels of group members.

On the basis of the exposition above, 'failing' and 'dropping out' can be said to mean same thing in this course. Accordingly, two dropped out students of the course agreed to drop out for low participation which was due to reasons external to the course. It should also be noted that group difficulties do not directly mean dropping out; rather, through active participation the difficulties undergone can be reflected on for the sake of conceptual learning.

4 LESSONS LEARNED

Based on the experiences collected, Open Data and Open APIs enable both a unitary course theme and full innovation by students. In the context of entrepreneurial learning, Kyrö (2005) has stated that room for possibilities and creativity must be provided by the teacher; in our experience, these are easy to implement with the open theme where software development and creativity easily elide.

4.1 Main Success: Ability to Create a Software Product in a Student Group

All the groups have been able to implement software products that are proofs of concept for the ideated uses. Some products have been mature and ready for deployment. Preliminary analyses of students' learning reports indicate that the students are able to reflect their experiences on the group work concepts and become aware of the meaning and purpose of the software process in software development work. After two course iterations, of the 43 students who have started the course, two have dropped out, as they noticed their low participation. Besides the two dropouts, no-one has failed. As noted in Section 3.6, in this course where active participation is the main criterion for passing, failing and dropping out mean same thing. Other two students were given a small additional assignment to give them a genuine chance to improve on their skills in the presence of different skill levels in the group, the issue that led to uneven participation and could not be satisfactorily resolved in the course of the short project. In our experience, the open theme provides a feasible contextualization for the 12-week project-based course.

Various project topics have been ideated and implemented. One group developed a zombie game using Open Street Map information as the game field (as suggested by Friberger and Togelius (2012)). Another group developed a service where listings for cheap apartments for rent (suitable for a student budget) were collected from several service providers and shown on a map of the local area with links to further information. Another of the location-based application placed elk accidents on the map with various UI controls for filtering the information. One group developed an application that calculated and visualized expenses related to car, train, or bus travel. Another project enabled the user to place various United Nations data sets in a three-dimensional world map where they could visualize differences between countries using height and color.

Of the eleven projects finished so far, nine are web services, one is a desktop application, and the remaining one a game started as a desktop application. Data usage scenarios include scraping, having data locally, and using data through API calls. As for the scope of projects, students typically (in minimum) aim at finishing the main use case where the integration of components that request, parse, manipulate, and show about data is already in place, and by which the product idea can be demonstrated to an audience.

Students overcome considerable technological learning challenges on top of starting the project with a newly formed group. Many of the students are unfamiliar with Linux, which is a straightforward operating system for many frameworks used in small scale Web application development (the typical technological context for the student ideated projects). Students often learn a new programming language, such as Python, while they use C# and Java during their introductory programming courses. They typically write code for running a server and build UIs with HTML5 and JavaScript, with a particular CSS library, such as Bootstrap. Some groups choose to adopt modern frameworks, such as AngularJS. The students learn to seek and select recent tools particularly applicable for their projects. This process is supported by the technical supervisor. Overall, students learn to build useful software products by *integrating* various data sources and technologies, meaning that many of the products reflect the 'mashup' character in recent Web development.

4.2 Challenges and Suggested Improvements

When students begin this course, which requires both creativity and implementation of a 'proof of concept' product in a short time frame, they are undertaking a learning process where issues are bound to emerge. These are reviewed below.

4.2.1 The Challenge of Self-directed Learning

Students tend to struggle with relating to a course requiring self-direction and individual responsibility in a group. After being exposed to the highly structured introductory university courses during their prerequisite studies, they face a course in which they do not work for the teacher at all. Therefore, we must consider system level in terms of schooling background, and, in particular, the first, usually highly structured, university courses. In the present computer science context, for instance, we wish to answer the question

> What is the effect of *evenly educating and highly structured* introductory programming courses on students' perceptions of self-direction required in the actual profession?

Beginner's difficulties with programming are extensively documented. If highly structured and evenly educating introductory courses are needed, then courses such as the one presented here could help prompt self-direction early after the introductory courses. Students' first experiences with university courses are in any event likely to reinforce a 'departmental view' of their positions as learners (Parlett, 1977).

We also put forward the question above on the grounds that, during the project course, students are heavily informed of the active learner position required and the related pedagogic purpose, yet we have observed that a great deal of coaching resources are expended on this concern of self-direction. We are also fully aware of and using the coaching practice that prompts students to adopt concrete project routines early to decrease the known confusion stage at the beginning of projects.

To improve students' awareness of the requirement of self-direction, we plan to better describe the roles of all stakeholders at the beginning of the course. This means that we will increasingly inform students about the role of the teacher, which is not that of a customer (but a facilitator). In this connection, we will also emphasize how the course setting is different from other courses that require coursework through weekly lectures and exercises and provide example programs that can be followed as references for the course assignments.

4.2.2 The Challenge of New Technology

Students may feel overwhelmed by the requirement to ideate, implement, and showcase a software product from scratch in 12 weeks. They often seem unclear on the effort needed to implement a useful software by integrating data resources using technologies that are new to them. We argue that this relates to students' experiences of self-efficacy, such that low self-efficacy hinders students' ability to adopt work in the innovation-based environment. In principal, students are usually not troubled by the task of ideating, whereas their in-progress professional self-efficacy related to adopting new technologies can present severe obstacles in project initiation. We observe that students who are further in their bachelor studies struggle less with this issue.

So far our attention to technical issues during the starting session has been a review of couple of example programs in the sense of promoting student cre-

ativity, showing examples of open resource usage, and providing initial tip list of web development and other frameworks and libraries.

We now plan to concretize more the software development effort for the students at the beginning of the course. This could mean adding a separate crash course lecture that shows example applications together with programming code, and carefully explains the design and the code to the students. This would allow us to provide a correct picture of the course challenge among the students immediately when the course starts, and illustrate the course as a doable effort. At the same time, we would be able to demonstrate scraping and typical uses of APIs early for all the students, and potentially reduce less experienced students dependence on more experienced ones. Further, potential emotional reactions to independent technological learning requirement should also be explicitly discussed at the beginning of the course to help students understand that these feelings are natural.

4.2.3 The Challenge of Variations in Student Characteristics

After two course iterations, we have noticed the need to address differences in skill level in forming the groups. We acknowledge here that differences in skill level can lead to enriched learning experiences during group work (Oakley et al., 2007), which is why we have not given a lot of attention to this aspect in grouping so far. However, the project's goal setting on the basis of similar skill levels in a group would be important in light of how students may be overwhelmed by the requirement of technological learning (cf. Section 4.2.2). With great differences in skill level, the skillful students' high goals may force less skillful students to be dependent on others throughout the project. We want to avoid such phenomena stealing attention from the task of innovation itself. We plan to roughly group students according to study year and hobbyism in the future.

Another challenge relating to student characteristics is how they relate to (in this case, creative) software development work. Regardless of the emphasis on creativity and innovation, the course is one of many required in the degree system. For us, the important question is to what extent we should push students toward passionate work. This concern emerges from students reporting discomfort with a highly passionate group member and a desire to take a more moderate approach to work. Practical courses reflect working life, and the question of how to emotionally and motivationally *relate to* the work in the group context could be raised during the project lectures among other group work topics.

4.2.4 Lack of a Proper Release Challenge

An unfavorable pattern can emerge when students overcome their group work and self-direction challenges and show abilities to ideate and implement a product, only to see the completed project as 'just a course assignment' without big-picture relevance. This perspective can leave the course without purposive outflow of newly created products (Huizingh, 2011). Different from our master's level customer project, these is no final release of project deliverables. In addition, in the context of the local CS curriculum, there is no larger business study component underpinning the course that would inspire students to continue with their projects as the course ends.

Here, our plan is to consult with students about establishing a showcasing site with links to GitHub or other platforms used by students for their product development. Such a site could be connected with a local open data community, meaning that student work would become 'involved'. The open data contests in Finland (see (Eteläaho, 2014)) could be exploited for this purpose as well. The important question here is how to deal with differences in the maturity of the products (i.e., how will students perceive these differences if the products are publicly available).

4.2.5 The Intellectual Property Challenge

When exploring various resources for their work (data, APIs), students encounter a 'jungle of terms of conditions (TOCs)'. This particularly occurs when a student group uses data sources that are 'semi-open' in nature, with some specific and often ambiguous clauses stating the conditions of use. The intention is to act legally. Unclear situations have been currently resolved with the teacher requesting expert comments from the department personnel known to possess the competences needed. Alternatively, where students have scraped web content, the teacher has guided groups to contact the service providers to request permissions for using the data, which has been a successful procedure. The groups have either received permissions or an answer that has helped them to revise their project goals. Altogether the IP issues have introduced a great challenge, kind of a delay element from the teaching perspective, as the interpretation of TOCs and various small clauses can require juridic expertise.

Preferably, those groups who deal with complex license and TOC texts could be provided with a 'consultation hour' from the university jurist. The project

course is one alternative for the mandatory practical study component in the students' bachelor studies, and the responsibility for answering juridic questions that arise during the course naturally rests on the course provider. Having an explicit course arrangement for this purpose would make the course beginning more fluent and underline IP issues as an important learning topic. Another option would be to guide the students to limit their project ideation to resources whose conditions are phrased without any ambiguities. However, in our view, this latter option would limit how the course concept can foster student innovation ability.

In our experience, the use of various data resources and APIs increases both students' and supervisors' useful knowledge on intellectual property rights and licenses, for what reason we recommend that also the course teachers attend the expert consultation hours outlined above. These topics are also similar to recommended topics in CS by the ACM/IEEE Computing Curricula ('Social Issues and Professional Practice/Intellectual Property') (Sahami et al., 2013, p. 24).

4.2.6 Summary

Many of the challenges above reflect our overall experience that the course concept described is an efficient tool for prompting particular transformation and self-related honesty early in the curriculum. That is, students can be coached to overcome their self-efficacy-related challenges and learn how interested he or she is in the computing discipline. In our experience, this transformation does occur, and at different paces for different individuals. The teacher's patience and continuous coaching are keys to the transformation toward self-directed creative work.

5 CONCLUSIONS

This paper described a course concept that builds on student innovation ability, including attributes such as creativity and co-creation. The course is based on open data and programming application interfaces.

Apiola et al. (2012) have recently proposed that the CS higher education in Finland should better support creativity and innovation-friendliness, especially creative problem solving. The purpose of this paper was to describe, as the early stage of an action research project, one approach toward this direction. More precisely, Apiola et al. proposed that components of a creativity-supporting learning environment include competence (i.e., solving challenging problems), autonomy (i.e., self-direction), relatedness (i.e., social interaction and self-adaptive working methods), domain relevant skills (i.e., good computing skills), and cognitive processes and working styles (i.e., creativity-enhancing methods in course sessions). We conclude that, according to such a definition, a creativity-supporting learning environment in the undergraduate project course was established. The open theme provided the framework for implementing a course concept of this kind.

Our specific interest was to report challenges that could inform our subsequent action research and that of other educators designing similar courses. In our view, many of the challenges reviewed relate to self-direction and the curricular position of the course. After highly structured introductory university courses, students may be overwhelmed by the challenge of a creativity-based course. Generally, this concerns how students are able to undertake self-directed study processes, which here signifies the responsibility to advance the project without an external customer. A related challenge concerns the students' self-efficacy, which emerges from the challenge of independent technological learning required by the course. Students may not know that useful, proof of concept open data applications can be created with reasonable effort, and demonstrate low self-efficacy not knowing that this is unnecessary.

We have altogether observed that student groups are able to overcome their challenges to produce an outcome. As reported by Lin and Chen (2012), commitment and self-efficacy are important mediating factors for a student group with diverse competences to grow into a team capable of open innovation. Based on the collected experiences, *continuous coaching and teacher patience* contribute to transformation toward condition that is fruitful for innovative work in the early curriculum. In this regard, our subsequent action research is likely to benefit from the theory of transformative learning (Mezirow, 1981) as a framework for a 'hidden curriculum' designed to improve students' self-efficacy in the computing discipline. More precisely, we need to conceptualize a particular 'perspective transformation' (a term used by Mezirow) where a learner realizes that self-directed work emphasized through creative effort reflects the computing profession and is therefore valuable.

REFERENCES

Apiola, M., Lattu, M., and Pasanen, T. A. (2012). Creativity-supporting learning environment—CSLE.

Trans. Comput. Educ., 12(3):11:1–11:25.

Auer, S., Bryl, V., and Tramp, S., editors (2014). *Linked open data – creating knowledge out of interlinked data: results of the LOD2 Project*. Springer.

Bhaskar, R. (1978). *A Realist Theory of Science*. Harvester Press, Sussex, UK, second edition.

Brown, J. (2000). Bloodshot eyes: Workload issues in computer science project courses. In *Software Engineering Conference. APSEC 2000. Proceedings. Seventh Asia-Pacific*, pages 46–52. IEEE Computer Society.

Brown, R. (1988). *Dynamics within and between Groups*. Basil Blackwell, Oxford, UK.

Burge, J. and Gannod, G. (2009). Dimensions for categorizing capstone projects. In *Software Engineering Education and Training. Proceedings. 22nd Conference on*, pages 166–173, Los Alamitos, CA. IEEE Computer Society.

Burnell, L. J., Priest, J. W., and Durrett, J. R. (2003). Assessment of a resource limited process for multidisciplinary projects. *SIGCSE Bull.*, 35(4):68–71.

Carr, W. and Kemmis, S. (1986). *Becomming Critical: Education, Knowledge and Action Research*. The Falmer Press, London.

Castanier, E., Coletta, R., Valduriez, P., and Frisch, C. (2013). Websmatch: A tool for open data. In *Proceedings of the 2Nd International Workshop on Open Data*, WOD '13, pages 10:1–10:2, New York, NY. ACM.

Chesbrough, H. W., editor (2003). *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press, Boston, MA.

Clear, T., Goldweber, M., Young, F. H., Leidig, P. M., and Scott, K. (2001). Resources for instructors of capstone courses in computing. In *ITiCSE-WGR '01: Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 93–113, New York, NY. ACM.

Conradie, P., Mulder, I., and Choenni, S. (2012). Rotterdam open data: Exploring the release of public sector information through co-creation. In *Engineering, Technology and Innovation (ICE), 18th International ICE Conference on*, pages 1–10.

Dahlander, L. and Gann, D. M. (2010). How open is innovation? *Research Policy*, 39(6):699–709.

Daniels, M., Berglund, A., Pears, A., and Fincher, S. (2004). Five myths of assessment. In *Proceedings of the sixth conference on Australasian computing education - Volume 30*, ACE '04, pages 57–61, Darlinghurst, Australia. Australian Computer Society.

Domingo, A., Bellalta, B., Palacin, M., Oliver, M., and Almirall, E. (2013). Public open sensor data: Revolutionizing smart cities. *Technology and Society Magazine, IEEE*, 32(4):50–56.

Eteläaho, A. (2014). Analysis of the received applications in the open data contests in Finland 2010–2013 (draft.). Technical report, Tampere University of Technology, Department of Pori.

Fila, N., Myers, W., and Purzer, S. (2012). Work in progress: How engineering students define innovation. In *Frontiers in Education Conference (FIE), 2012*, pages 1–6.

Fincher, S., Petre, M., and Clark, M., editors (2001). *Computer Science Project Work: Principles and Pragmatics*. Springer-Verlag, London.

Friberger, M. G. and Togelius, J. (2012). Generating game content from open data. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 290–291, New York, NY. ACM.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 3rd edition.

Hand, D. (2013). Data, not dogma: Big data, open data, and the opportunities ahead. In Tucker, A., Höppner, F., Siebes, A., and Swift, S., editors, *Advances in Intelligent Data Analysis XII*, volume 8207 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg.

Huizingh, E. K. (2011). Open innovation: State of the art and future perspectives. *Technovation*, 31(1):2 – 9. Open Innovation - {ISPIM} Selected Papers.

Isomöttönen, V. (2011). Theorizing a one-semester real customer student software project course. In *Jyväskylä Studies in Computing*, volume 140. University of Jyväskylä. PhD Thesis.

Isomöttönen, V. (2014). Making group processes explicit to student: A case of justice. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, ITiCSE '14, pages 195–200, New York, NY. ACM.

Jaakkola, H., Mäkinen, T., and Eteläaho, A. (2014a). Open data: Opportunities and challenges. In *Proceedings of the 15th International Conference on Computer Systems and Technologies*, CompSysTech '14, pages 25–39, New York, NY. ACM.

Jaakkola, H., Mäkinen, T., Henno, J., and Mäkelä, J. (2014b). Open^n. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 608–615.

Kaihlavirta, A., Isomöttönen, V., and Kärkkäinen, T. (2015). A self-ethnographic investigation of continuing education program in engineering arising from economic structural change. *Studies in Continuing Education*, 37(1):99–114.

Klug, B. (1976). To grade, or not to grade: A dramatic discussion in eleven parts. *Studies in Higher Education*, 1(2):197–207.

Kyrö, P. (2005). Entrepreneurial learning in a cross-cultural context challenges previous learning paradigms? In Kyrö, P. and Carrier, C., editors, *The dynamics of Learning Entrepreneurship in a Cross-Cultural Univeristy Context*, pages 68–103. University of Tampere Research Centre for Vocational and Professional Education.

Lewin, K. (1946). Action research and minority problems. *Journal of social Issues*, 2(4):34–46.

Lin, C. P. and Chen, S.-H. (2012). The role of integration mechanism in open innovation team: An ex-

ploratory study on cross-field student team contests. In *Technology Management for Emerging Technologies (PICMET), Proceedings of PICMET '12*, pages 1953–1960.

Mezirow, J. (1981). A critical theory of adult learning and education. *Adult Education*, 32(1):3–24.

Moore, R. (2000). For knowledge: Tradition, progressivism and progress in education—reconstructing the curriculum debate. *Cambridge Journal of Education*, 30(1):17–36.

Morales-Chaparro, R., Sánchez-Figueroa, F., and Preciado, J. (2014). Engineering open data visualizations over the web. In Luo, Y., editor, *Cooperative Design, Visualization, and Engineering*, volume 8683 of *Lecture Notes in Computer Science*, pages 51–59. Springer International Publishing.

Oakley, B., Hanna, D., Kuzmyn, Z., and Felder, R. (2007). Best practices involving teamwork in the classroom: Results from a survey of 6435 engineering student respondents. *Education, IEEE Transactions on*, 50(3):266–272.

Otjacques, B., Stefas, M., Cornil, M., and Feltz, F. (2012). Open data visualization: Keeping traces of the exploration process. In *Proceedings of the First International Workshop on Open Data*, WOD '12, pages 53–60, New York, NY. ACM.

Parlett, M. (1977). The department as a learing milieu. *Studies in Higher Education*, 2(2).

Pears, A. and Daniels, M. (2010). Developing global teamwork skills: The Runestone project. In *Education Engineering (EDUCON)*, pages 1051–1056. IEEE.

Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthrone, E., Katz, R., LeBlanc, R., Reed, D., Roach, S., Cuadros-Vargas, E., Dodge, R., Kumar, A., Robinson, B., Seker, R., and Thompson, A. (2013). Computer science curricula 2013.

Suri, D. (2007). Providing "real-world" software engineering experience in an academic setting. In *ASEE/IEEE Frontiers in Education Conference, 37th annual*, pages S4E–15–S4E–20. IEEE.

Telikicherla, K. C. and Choppella, V. (2014). Enabling the development of safer mashups for open data. In *Proceedings of the 1st International Workshop on Inclusive Web Programming - Programming on the Web with Open Data for Societal Applications*, IWP 2014, pages 8–15, New York, NY. ACM.

Tomayko, J. E. (1998). Forging a discipline: An outline history of software engineering education. *Annals of Software Engineering*, 6(1):3–18.

Tuunanen, T., Koskinen, J., and Kärkkäinen, T. (2009). Automated software license analysis. *Automated Software Engineering*, 16(3-4):455–490.

Yang, H.-L. and Cheng, H.-H. (2010). Creativity of student information system projects: From the perspective of network embeddedness. *Computers & Education*, 54(1):209–221.

Yunfei, P. and Qin, D. (2009). Cultivating the innovation ability of college students in course teaching. In *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on*, volume 1, pages 828–831.