Infusing Design Thinking Into a Software Engineering Capstone Course

Maria Palacin-Silva, Jayden Khakurel, Ari Happonen, Timo Hynninen and Jari Porras
Lappeenranta University of Technology
name.lastname@lut.fi

***Abstract*—Software engineering (SE) educators are challenged to balance the scope and depth in their courses to train students in skills which will fulfill the ever-evolving industry needs. Capstone courses are a tool for educators to transfer hands-on experience into practical knowledge and skills of SE students. This paper describes the design of a Casptone course, at Lappeenranta University of Technology. The designed course is human-centric SE capstone, infusing design thinking methods and agile practices into the project life-cycle knowhow. The capstone was offered in spring of 2017 as a 16-week course for 29 students. Design thinking was effective to perform requirement elicitation, software design and testing. Also, the applied approach allowed students to be self-directed which increased their motivation, as a result there was 0% dropout rate. Design thinking is a powerful mean of problem solving and effectively supports SE education in bringing a more hands-on and minds-on, problem-based curriculum.**

***Index Terms*—software engineering; education; design thinking; human-centric; capstone; software curricula; course design**

I. Introduction

IN A FAST CHANGING WORLD, Software Engineering (SE) educators face challenges to create courses which prepare students for a professional life in industry - courses with concepts that are applicable and real. At the same time courses need to be technology-agnostic to an extent that the fundamental principles are imparted enabling students to operate effectively in an evnironment of constantly evolving technology . Consequently, educators must create a balance among the quality, scope, depth, applicability, soft and hard skills, individual and collaborative learning in their teachings [1]. Project Based Learning (PBL) is an effective mean to train SE students to face real world challenges [2]. An increasingly popular way to bring PBL into SE education is through capstone courses [3] as these are a tool for educators to engage SE students to engineer a solution from the beginning to the end and therefore expose them to practical project working, collaboration with other team members and project management in a hands-on setting. [4]–[7]. Additionally, capstones have been recommended by the ACM curriculum guidelines to be included already in undergraduate SE education programs [8]. A topic which has grown in relevance as societies and industries need highly usable software systems [9]–[11] is Design Thinking (DT). In industry, companies such as IDEO, NEC, Airbnb, Microsoft, SAP use DT methods to improve their processes, solutions and even working environments. In SE education DT has stared to be applied for teaching safety-critical systems engineering [12], mobile applications design [9] and games development [13]. However, DT has been reportedly suggested to have the potential to be used for project-based learning [14], requirements elicitation [15] and cybersecurity [11], [16] as well. As a result, there seems to be a gap between the application of DT into software development in industry and the knowledge being spread in SE education. This motivated us to report our course design, separating our solution from standard running a software project, as our solution is a human-centric capstone including design thinking methods and agile practices into the project life-cycle.

Our goal is to fill this gap between DT in industry and in SE education, trough the analysis of our course results and practical knowhow for other researchers and educators benefit. This paper presents the methods, results and experiences on how we designed the course as a human-centric software engineering capstone by infusing design thinking methods and agile practices into the SE project life-cycle. This article represents a practical example about the importance of project-based courses (such as capstones) in software engineering education. Students were given real-world challenges based on the Urban Solutions for a Living Planet Report [17] and they planned, designed, developed and tested six solutions within 16 weeks. The idea of this is to emphasize complex real-world challenges.

This article presents in section II the related work regarding capstones and design thinking in software engineering, followed by the course design and projects' stages in section III. Section IV, focuses on the analysis of the teams and summarizes the projects developed. Overall student lessons learned and student perceptions are presented in section V. The conclusions of the study are provided in section VI.

II. Related Work

A. Capstones and Software Engineering

Software engineering as an application area is particularly challenging in providing students with the experience that they will find useful after graduating [18]. Project Based Learning (PBL) is an effective mean to train SE students to face real world challenges [2]. A contemporary and increasingly popular way to bring PBL into SE education is through capstone courses [3], [4] as they are a tool for educators to transfer hands-on experience and have been reported to be an effective mean to prepare students for a profession in industry [5]–[7]. Practical project working, collaboration with other team members and reflection of the task at hand

2377-570X/17 $31.00 © 2017 IEEE
DOI 10.1109/CSEET.2017.41

have been established as effective tools to enhance learning and increase students' self-efficacy [5]. Capstone courses also enhance student learning, participation and satisfaction [19]–[21].

According to ACM curriculum guidelines for software engineering, capstone projects should be included already in undergraduate SE programs, because curricula should be situated firmly into the work life needs [8]. For SE education to be effective, students should be given an opportunity to apply their skills. Significant SE projects accomplish this by allowing the students to deepen their skills in many SE knowledge areas [8].

Capstone courses at undergraduate level have been used to teach agile methodologies [22], for example scrum and the roles of software development teams [23]. Senior-level capstone courses, particularly in design and engineering, gained popularity already before the millenia [4]. On senior-level courses students have been able to complete large and impressive projects, such as game development projects [24], [25], [26].

B. Design Thinking and Software Engineering

Design as practice evolved from making products attractive to nest innovative ideas as a way of thinking [27]. As is problem solving, design is also a natural and universal human activity [10]. This is why for long time, design has been a process applied to make physical functional objects look good (from fashion and transport, to construction). Along the time, it became clear that a good design could lead to great success, with examples ranging from post-it notes to apple products [28]. Design grew organically since 1920s and started to be applied into more than just products (from business strategies, social innovation to technology design). As a result, the discipline of design thinking (DT) appeared into the academic literature and business media in early 2000s [29], [30]. However, already in 1969, Nobel laureate Herbert Simon had seen *"design as a way of thinking rather than a physical process* [31].

DT is an approach to solve complex problems by using creative methods. In DT, the empathy between the designer and the users is highly regarded and the exploration of problems and solutions is a shared process between these two [29], [32]. This is aimed to go beyond assumptions that can affect negatively the solutions. As a result of applying DT, the best possible solution to a complex problem should be found [33]. DT has been applied in multiple fields to solve complex and wicked real world problems (e.g. creating corporate strategies, technology designs, security and education and social innovation) [9]–[11]. In recent software engineering education, DT has been used to teach safety-critical systems engineering [12], mobile applications design [9] and games development [13]. However, DT has been reportedly suggested to have the potential to be used as well for project-based learning [14], requirements elicitation [15] and cybersecurity [11], [16]. Furthermore, [34], [35] reported that agile methods could be combined with design thinking in order to improve software development. As both, agile development and design

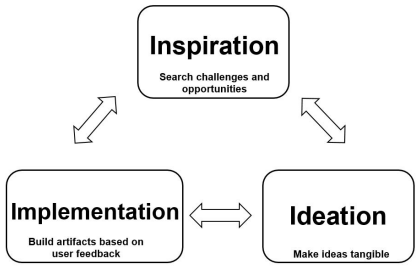

Fig. 1.

thinking emphasis on people over processes, these two seem to integrate naturally. Meanwhile in industry, DT in software development has been regarded as a post-agile approach which is already in place in many companies and has been key in the success of big tech companies such as IDEO and NEC [34], [36], [37]. As a result, there seems to be a gap between the knowledge about design thinking being taught in software engineering education and the application of design thinking in industry. Therefore, a human-centric capstone which infuses design thinking methods and agile practices into the project life-cycle has been considered in the effort to bridge the research gap.

Design thinking has three iterative stages: inspiration, ideation and implementation (Figure **??**) [27], [33]. DT Projects loop between the stages. The first stage (inspiration) is aimed at generating an empathy with the stakeholders that boost the search for solutions some methods used in this stage are personas and story boards. In the following stage (ideation) various solution ideas are developed and tested some methods used are journey maps and prototyping. The final stage (implementation), focus on the development and run of the selected solution idea [9], [38].

The use of DT methods such as personas, story boarding, journey maps, prototype and usability testing is a way of incubating ideas and creating innovative solutions within teams. As *"these techniques of design thinking methods has ability to enhance communication within multidisciplinary teams, but also in their simplicity in use by non-experts"* [39]. Below some of the techniques of design thinking methods have been presented.

- Personas: Personas can be used as the foundation and inspirational approach to develop the application ideas that are initially discussed. Students can define the fictitious, specific, and concrete representations [40] of a group of target application users who share some common characteristics, needs, and goals [41] so that each student can initially understand and address the target users needs and preferences, rather than depending on their own needs and preferences.
- Story Boards: Storyboards can also identify potential consequences that users might face with the application in the future, based on the personas created in the previous stage. Storyboards can be used as effective media to capture and explore the user experience by translating the story and script into scenes through who, what, when,

where, and how using images and text [42], which could later serve as the constant reminder of the usage and task context [43]. Further, [43] states that vision, idea, board context, background, scenario development, design, and the storyboard session are seven steps that are essential in creating the storyboard.

- Journey maps: This technique has been used extensively in the service design industry under several names, such as customer journey, customer journey map, and experience journey, [44] for many years. In the application design context, the user journey map can be defined as a visual illustration of a series of steps or interconnected touch points, which users will experience while engaging with an application. They are depicted chronologically and often accompanied by emotional indicators [45].
- Prototyping and usability testing: Prototyping is the pre-verification technique and can be defined as a quick and effective way to optimize and facilitate innovative ideas through iterative conceptual design work, a process that lends itself well to early user experience research [46], which reduces the risk of failure [47] and enhances time to market (TTM) of application.

III. Course Design

The course *Running a Software Project* was spread in 16 weeks, with 8 sessions of 3 academic hours and 4 mentoring sessions (teams could request more mentoring sessions if needed). The course was designed to require on average 7,5 hours of student effort weekly. A total of 29 students took the course from beginning to end (the dropout rate was 0%). The Course outline is presented in Table I. Course stages are aligned with specific software project iterations aligned with the Rational Unified Process (RUP) [48], with each stage emphasizing the particular software project phase within an iteration. At the same time, those stages are aligned with the DT stages. The lectures had a flipped classroom like setup, as students had to read materials posted on their Moodle before coming to the lecture. Similar approaches have been applied to teach SE foundations by [1] and to teach programming by [49].

The learning objectives in this course were defined as follows:

- Plan, estimate and monitor a software project in an iterative and incremental manner;
- Understand design thinking methods and usability techniques and, their practical application into a project's life-cycle
- Implement as software system from scratch
- Familiarize with contemporary development tools and environments;
- Use modern software engineering practices to plan, monitor and deliver a realistic software project that meets stakeholders needs, is highly usable and sustainable.
- Understand the new business challenges and opportunities for their software projects

It can be highlighted that the list of learning objectives gives an emphasis to soft and hard skills.

TABLE I
Course Structure

Week	SE Stage	DT Stage	Project Outcome
1-2	Project Plan and Business Modeling	Inspiration	Scenario selection Project plan
3-4	Requirements Analysis	Inspiration	Personas Story Boards Journey Maps
5	Analysis and Design	Ideation	Prototypes
6-10	Development	Implementation	Artifact development
11	Testing	Implementation	Validation and Verification Testing
12	Testing	Implementation Ideation	Usability Testing Public Testing
13-15	Deployment and Maintenance	All	Maintenance Report
16	Presenting Project Solutions		

A. Project Stages

The course followed a collaborative learning approach where students worked together to achieve a positive goal - which was to engineer a software project- , with the lecturer as a facilitator. This approach has shown to develop critical thinking and improve the understanding of concepts [50]–[53]. Also, collaborative learning has become popular in SE for providing instruction in a group context [54]. The course consisted of four components: 1) flipped-classroom lectures; 2) in-class exercises; 3) weekly progress reports and, 4) individual project work.

The theme of the course was *Urban Solutions for Sustainable Development*. It was inspired by a report with the same name by WWF [17]. Students had to develop a project that would address one of the themes from this report.

The course aligned SE project stages with DT stages (see Table I). This was done by having a SE life-cycle infused with DT activities, this process can be seen from the DT perspective in figure 2 and from the SE perspective in figure 3. As a result, students learned how to move from the inspiration, ideation to implementation stages while following the SE stages. Each stage had a expert's lecture, progress presentations by teams and hands-on exercises. For example, requirements elicitation was done using design thinking methods such as, personas, story boards, journey maps and prototypes (paper and digital) instead of regular SE techniques. In order to verify the efficacy of this approach, in the testing stage we asked students to identify functional and non functional requirements from their initial project plans and User Experience (UX) design portfolio. The use of design thinking for requirements elicitation was successful as these techniques had improved the understanding of user needs. Also, students increased their interaction and empathy with stakeholders from early stages of the projects this turned into better user experiences and higher student motivation.

Following a description of those stages goals and activities:

- Project Planning: The main goal of this stage was to allow students explore problems and estimate possible solutions. Students chose a scenario where their solutions would be developed, then they elaborated an initial project plan. They presented their motivation and project plan at class on week 2.
- Requirements: This stage lasted three weeks and started with a lecture about DT and usability methods, which was complemented by hands-on exercises such as paper

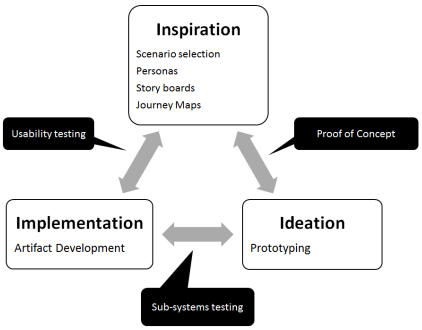

Fig. 2.

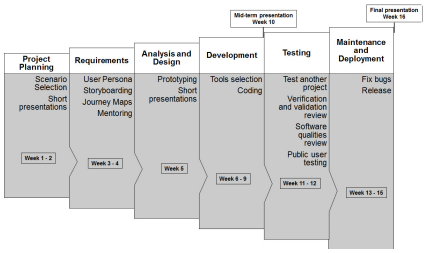

Fig. 3.

prototyping. The second week was used to draw their personas and story boards, the third week was used to perform prototyping. The main aim of this stage was requirements elicitation. Students had to understand users needs, emotions and journeys by using different DT techniques. They started talking with potential users from this stage, they showed them their early prototypes and gathered their comments. During this stage they had closely mentoring sessions - *specially because the students did not know DT methods beforehand* -.Following sub-section describes how personas, storyboards, journey maps and prototype were used in this course to collect the requirements.

- Personas: In the course, students created three personas presenting the users they had planned the project for. Personas includes the name, demographic characteristics, current frustrations, needs, and goals.
- Storyboards: As next step, students created a digital storyboard as an inspirational approach to explore the user experience and further brainstorm their own application ideas, with the focus on the storytelling rather than the technical details.
- User Journey Map: Students did applied user journey map technique in design process for an inspiration so that they could gain a holistic overview of the sequence of the touch points and actions that users should have with their application.
- Prototyping and usability testing:The prototyping technique was conducted in two phases, with usability testing as the ideation approach. Prototyping can be useful as a pre-verification technique to understand the users needs and solve design problems in the early phase of the development cycle. In the first phase, each team of students was asked to design the paper prototype and conduct usability testing with the users outside their design team to gather usability issues (i.e., before and during the mentoring session). The paper prototype was refined based on user feedback and followed by an interactive prototype, on which similar usability testing was conducted as in the previous session. In the second phase, each team of students included the feedback from the first phase and developed the actual fit for the users purpose and the prototype application. Each team was asked to conduct usability testing of the prototype application in the open environment with the public in order to understand what users think and how they interact. The phase concluded with the importance of external users in testing the prototype designed by students during the software development project. In addition, they can potentially understand the usability requirements, such as text size, color, and terminologies.

- Development: The building of the full software artifact took place in this stage. The stage lasted four weeks (with a break week in the middle). It started with a lecture about development practices and an introduction to frameworks and environments available for use. As this was a master's course, each team was given significant freedom to evaluate and to select frameworks, environments and programming languages that they wished to use. At the end of this stage, they had to give a mid-term presentation with a working demo of their solutions. Other students and stakeholders were invited to those presentations to give feedback to the teams.
- Testing: This stage lasted two weeks and was divided into two core activities 1) software testing and quality and 2) public testing. During the first activity, they received a lecture about software qualities based on [55], quality in use based on [56] and verification and validation testing. This lecture was followed by a inter-teams roles jigsaw exercise. Some team members were moved to a different team (to play the role of testers) and some were kept (to observe user interactions with their solutions). They were asked to a) identify functional and non-functional requirements from their project plan and UX design portfolio and b) Evaluate how well the solution meets the quality in use standard. In second activity public testing, booths were set up for each team in a popular area of university. They had to show their solutions to anyone passing by including, students and university staff. Each user had to fill a survey after testing a solution (They had to rate statements in a five point scale from strongly disagree to strongly agree). Those surveys were designed based on the ten statements from the systems usability scale (see table II) from [57], [58]. Finally students processed these surveys and used the results as inputs for the maintenance stage.

TABLE II
SURVEY STATEMENTS FROM THE SYSTEMS USABILITY SCALE [57], [58]

Systems Usability Scale Statements
I think that I would like to use this system frequently
I found the system unnecessarily complex
I thought the system was easy to use
I think that I would need the support of a technical person to be able to use this system
I found the various functions in this system were well integrated
I thought there was too much inconsistency in this system
I would imagine that most people would learn to use this system very quickly
I found the system very cumbersome to use
I felt very confident using the system
I needed to learn a lot of things before I could get going with this system

- Maintenance: This stage aimed at wrapping-up the stages students had completed before, it lasted three weeks. This, stage started with a lecture about software maintenance practices and workshop to review teams performance, practices, success stories and current issues. Students had to improve their solutions using the feedback gathered in the public testing day. They went through all DT stages to solve certain issues they did not expect.
- Presenting Project Solutions: Industry stakeholders and the local government were invited as jury for the final projects presentation. They evaluated the content of the final reports, the ideas and the project implementation. The feedback was positive and as detailed in section VI, they will play the clients role for this course in future.

B. The Teaching Process

The teaching team was composed by the authors of this article. The course was led by MSc. Maria Palacin-Silva, special lectures and mentoring - in particular topics- were given by MSc. Jayden Khakurel (DT techniques and usability design), DSc. Ari Happonen (Team dynamics) and MSc. Timo Hynninen (software quality testing).

The course had one session per project stage (Fig 3). Each session started with a 5 to 10 minutes progress presentation by each team. This would be followed by one hour lecture on a particular topic (e.g software quality testing). Then teams would have an in-class practical exercise to complete (e.g paper prototyping, in figure 4). Afterwards,students would receive an assignment to apply the session teachings onto their own projects and generate a deliverable (which would get presented during the following session). In addition, two sessions were fully dedicated to evaluate the teams detailed progress (a mid-term and a final presentation). Finally, one mentoring session per project stage was organized for the teams.

This course used moodle as a platform for sharing the reading and lecture materials, exercises, deliverables and discussion forums (e.g. a forum was set for students to brainstorm ideas and join teams).

IV. THE TEAMS

Project teams of four to five students were formed in the early stage of the course (via the forum tool of Moodle). In order to promote inter-cultural diversity, students were advised to team up with students of different nationalities.Students posted their interests and skills and searched for students

Fig. 4.

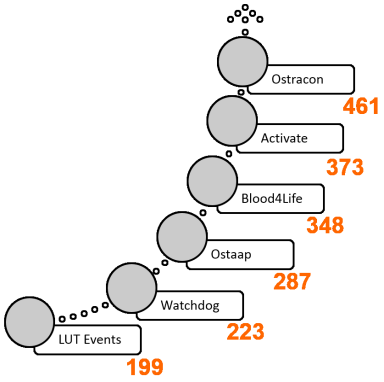

Fig. 5.

matching their interests. Teams were formed among students with closest interests. Each team had to appoint clear roles for each team member and divide responsibilities from the beginning of the project. Students had to record their project work hours and report them on a weekly basis. Figure 5 presents a summary of cumulative work hours per team (this figure includes all team members).

Teams had a total of nine deliverables consisting project plan, UX design portfolio, code repository, architecture, verification and validation review, usability testing results, weekly reports, belbin tests [59] and final report. Teams also had progress presentations by the end of each project stage. The two main presentations, a mid-term and a final presentation, were evaluated and rated by other teams and invited panel of external experts. Teams also had a possibility to receive bonus points for active participation in class and mentoring sessions.

A. Teams Analysis

In this course Belbin team analysis [59] was used for analysing the team structures. Belbin created a theoretical framework for team roles, as he saw that each individual could bring a special contribution to team work and, as such, enhance the team efforts [60]. The point of understanding the team structure is explained by [61], stating that team reflections can make the entire team more aware about the potential challenges and impending threats the team could face in future. This increased understanding can add internal team

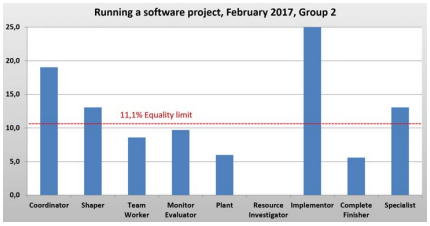

Fig. 6.

communion that can then balance collaboration in the team [62]. In short, the Belbin team inventory analysis divides team structures into nine core team roles [60]. This analysis is done to pinpoint the possible underlying team structures that have impacts to successfull teamwork. In the course, the concept of team analysis was explained to every team and all teams were assesed with belbin analysis tool. Each team were then given their own team structure. For example, the results of the team analysis for one of the student teams is shown in Figure 6.

The visualization of team 2 analysis results (figure 6 clearly shows the missing roles in the team. Team 2 did not have anyone who would prefer to take the role of *resource investigator*. Resource investigator helps the group to find external resources for the team. This could indicate a challenge for this team for example to find outsiders to test their product. On another hand, a lot of interest was given towards the role of implementor, which actually compares to the amount of implementation actions this group did.

The idea behind the use of this analysis tool was that when a team knows what are their individual preferences in team work, they could be better prepared to tackle the project challenges. As a result, some teams did notice later on why their team struggled in certain stages. For example, one team who had weak interest on plant role - also known as idea generator - struggled in the initial stages to find a scenario and idea for their solution. At the end of the course, team members reported, that seeing the Belbin results made them understand the source of their struggle.

This part of the course added an important learning outcome into the course as it helped teams to understand the team dynamics in work life. This is important outcome as skills to work in groups and efficientcollaboration are especially important in software engineering work life. Collaborative group work adds the amount of positive results [63] in software projects.

B. Projects

The projects were defined to follow a value-driven approach [64] as this approach allow students to focus on the actual user needs rather than what the user asks for [19]. A total of six projects were developed and implemented during this course:

1) Blood4Life: As the blood demand in health care around the world is increasing, the will to give and ability to donate blood in right place and time is an integral part of an efficient blood supply. Blood donation does not have a big physical impact on the health of a donor, but it can save lives. As blood donation is currently the only viable solution for blood shortage problems, this project was created to connect the people with the blood donation collectors at the right time and place. The aim of Blood4life is to help saving human lives. It was designed as a web application where individuals register and provide some basic information such as blood type and living area. Blood4life algorithm will then automatically notify the donors when their blood is needed in their area and the donors can book a blood donation time. For the health center the application offers monitoring of all the appointments and provide a channel to notify about the blood need.

2) Activate: Activate is designed to bring like-minded people together into shared activities to improve their mental well-being. It is an application to connect people with shared interest (e.g app allows to use distance / preferred area range as parameter). Activate differentiates itself from a typical social media tool as it not based on know connections. Individuals use the application and create or join in an event (e.g. movies discussion or dog walking) in a set location.

3) LUT Events: team found a gap in the way events are organized at Lappeenranta University of Technology(LUT). Many students leverage on campus events for social life to meet new people and to develop long lasting relationships but the information about events is spread in multiple forums (e-mail, facebook, websites). This causes the students to miss these important events. Therefore, this team decided that a single source of events information was needed. The project's aim was to implement a mobile phone application for accessing all events at LUT. The app uses a socail media approach in which users can easily inform (post), tag and find events as well as follow event organizers. The app provides also a way for organizers to reach the students (e.g. send event notifications). The platform also allows rating functionality in order to rank the organizers and help students to choose which paid event is worth of their money.

4) WatchDog: is a solution for city administration to see a structured view of problems in the area. This solution eases up operational planning and allows actions to be targeted where these are needed (e.g. prioritize deteriorating infrastructure). The web application helps to create a democratic framework for citizens to report any hazards in the city. A web app is used for taking pictures and classifying the problems (e.g. road problem, violation etc.). Application automatically attaches the location coordinates to the picture and the data is sent to the server. The city administration can see a visualization of the emerging problems on the city map. Items can be sorted in order of significance (per citizen consensus) to create more democratic operational plans. Citizens can follow the status of the reports until they are fixed by the city administration.

5) Ostracon: focuses on the problem of food waste as nearly 1/3 of produced food is wasted yearly [17]. The purpose of this project was to minimize food waste in the phase of cooking, serving and consuming in restaurants by increasing customer satisfaction in student restaurants. The application allows users to review and rate the food they had eaten. The feedback is detailed yet simple to give. This information is then processed and restaurants can have a granulated visualization of which items need to be changed or improved and

which ones to keep. This application also provided a feature for meal planning based on the feedback for restaurants. The intended effect is that restaurants can improve the food taste to fit more perfectly to the customers preferences. The projet assumes that this will produce less food waste.

6) Ostaap: was designed to promote the culture of smart consumption. The aim was to build a platform where you can access product reviews and find best deals around a town in order to avoid countless hours of wandering around shops and supermarkets. Users can also track interesting upcoming or ongoing sales and as such affact app users' smart consuming decisions. The design goal for the Ostaapp was to create a platform that gives decision power back in the hands of the consumers by providing crowd-sourced product reviews with the special emphasis on products which are sustainably produced, packaged or transported.

V. Lessons Learned and Perceptions

Our Running a software project course was designed to combine design thinking to iterative software development in a capstone manner. The lessons learned from the course can be looked from various perspectives.

A. Design Thinking Perspective

Different stages of the design thinking process infused into the iterative software development process all revealed some insights. These insights reflect how teachers perceive the learning of the students during the course due to DT:

- Personas: We discovered that creating personas inspired engineering students to look at problems from the perspective of a group of target application users, rather than from a technological and own perspective.
- Storyboards: We discovered that expressing ideas with real behaviors and actions of users through storyboarding helped students to further understand who, what, when, where, and how the application is used and identify the social and ethical implications. For example with storyboard classroom presentation, one team found out that their idea could invade the privacy of users. As a result, the idea was improved.
- User journey map: We found that the user journey map helped students to improve and enhance their design thinking by helping them understand the emotions of the application users in the early stages. In mentoring session, students pointed out that through the user journey map they managed to understand the deeper feelings of the users of the application, which might have been difficult otherwise.
- Prototyping: We discovered that students found the prototype design to be very helpful in improving their design process. For example, one of the teams pointed out during the mentoring session that having feedback from users during the prototype process was useful in outlining the user interface. Similarly, other teams pointed out that usability testing with external users helped them to understand the importance of the color and placement of a button. Introduction to the prototype testing with the public in an open environment helped engineering students to understand beyond design and development, and it also helped them with marketing. For example, one of the students pointed out that this not only helped students to understand users needs but it also helped them to market the application indirectly.

Design thinking approach fitted well together with the iterative development process of software. Also, The use of DT methods such as personas, story boarding, journey maps, prototype and usability testing was an effective way of incubating ideas and creating innovative solutions.

B. Teachers Perspective

Teachers reflected the implementation of the course in relation to the results and approached of their other course (with similar or different learning objectives). As a result the following lessons were recorded:

- The projects were based on real world challenges presented in the *Urban solutions of the sustainable development* but the real industry and society linkage remained weak. The industry stakeholders and the local municipality representatives were invited as jury for the final project presentations but in future they should be used for defining the local challenges. Teams did engage external users for the testing but they did not have a clear client. However, teams explored business opportunities for their solutions, as a jury of their mid-term presentations pointed out that their solutions could have real value for society. Teams participated in an entrepreneurship event with their solutions and two out of six projects won a local entrepreneurship competition (with some financial compensations) and three are continuing with their projects.
- The capstone course design was effective to motivate students to participate actively in the entire course. Teams took ownership of their projects and their level of motivation was high. Dropout rate was zero, which is not common in work intensive courses like this. Regardless of this seemingly successful approach some enhancements, like gamification [65] approaches should be considered. For this the methods should be like defined in [66], where game like mechanisms area applied in non-game environment. This is extremely important for course like this, as based on [67], models that have applied some sort of gamification mechanism have been able to increase collaboration [68]. In addition to that, it has been show that gamification is adding into motivation to achieve the set goals [69].
- The capstone course design was rather work intensive both for teachers and students. The active interaction with the teams kept the motivation high but at the same time required active communication and mentoring. Each team could have be assigned a separate teaching assistant to monitor closely their progress, dynamics and support them. [1] shows that this approach can be effective in a SE project-based environment. Teachers wanted to keep holistic perception and all teams we co-supervised.

Teachers were not reporting their hours used to this course while students needed to document and report their work. This course was worth 5 ECTS, which means 116 hours of work per student (including the teaching and mentoring sessions). As it can be seen in figure 5, none of the teams (composed by 5 team members) reported to have worked 582 hours or more. However, students perceived this course as a heavy load course for the credits it gives. This might be due to students underreporting their worked hours. Some studies [70]–[72] have shown that perceived workload is not synonymous of time spent studying but, rather a result of the learning environment (which includes context, difficulty, type of assessment, and human interactions). Thus, the measurement of worked hours as such is not a useful probe of learning behavior and outcomes [70], [73]. Also, PBL has been reported to increase the load perception on students [74] and, therefore it should be carefully monitored. In future, we will present additional material and tools to support their time management skills. Also, we will use additional measures - than worked hours - to understand and assess the student workload.

C. Students Perspective

Students of the capstone course continuously reflected self and team learning through reports to help the teachers to measure the teaching and learning efficiency in different stages of the course. All 26 students and 6 teams (response rate for the reflections was 100%) performed a set of reflections about what they did learn in the course. 49 different learning related reflections were collected, recorded and analyzed in a systematic manner. The results show that students were somewhat surprised on how much they actually learned about different details related in developing of a complex software solution. As a result of analysis, 6 most repeated learning outcomes has been pointed out in the following list:

- *"We did learn a lot about team and project management dynamics (including documentation, project time tables, skills and work division, collaboration and so on).".* Many students stated that Belbin test opened their eyes about their own and their teams strengths and weaknesses and helped them to realize that they might need outside help to succeed in their project. In addition, students also noticed that the efficiency difference between participants in teams and that they needed to use the skills people have in vice manner. E.g. students wrote that *We did learn team skills and we noticed that knowing others specialties and skills is important Step-by-step course approach but everyone as responsible person in some part of the course.*
- Students found out the importance of prototyping and developing more ideas / solution and implementation path prototypes than what is needed for minimum effort. For example students wrote that *the course taught valuable lessons in how to conceptualize an idea to a prototype and how to improve the original idea based on the prototype feedback.*
- Students learned the value and meaning of effort put into a project. The extra work actually really differentiates their results from other teams. For example students wrote: *Level of details considered largely affected the end quality.*
- Students recognized the value of the input given by non-engineer back grounded people. Public testing of the solution gave the students the possibility to learn this valuable lesson. Students stated that *in future they might need to listen more what outsiders have to say about their solutions.*
- Students really learned teamwork. Length of the course is one important factor while teaching the students how to work as a team. In shorter courses students can pretend to work as a team, but in capstone course they truly have to divide the work and trust on each other. This course revealed the real challenges in team dynamics and as such was highly praised by the students by stating that this was *an eye opener to aspects or details that we did not think of as a team.*
- Students learned the big picture of application projects in capstone course. This just like teamwork is understood when the problem is complex enough. Some students stated that they never realized how many steps and details are needed to cover when an application is designed for bigger crowds This was reflected as *a project that has given me the opportunity to work on a group to develop a project from a simple idea in a professional manner.*

As final conclusion on this analysis the following reflection truly explains what this course was about. *When you are a normal user of different applications you are not conscious about the real steps that were taken to develop it. You assume that to develop an application someone needs just to code. What we learned from this course is that there are certain processes that should be carried out, as much important as the coding itself. UX Design is one of them. Storyboards, Personas, Scenarios are all crucial to understand what user really needs... of the general message, most of the course reflection did pin point*. Students stated that they were surprised how (as engineering students) they sometimes have somewhat narrow view on what a professional application development truly is. Overall students reflected that this sort of a course is highly efficient professional work life simulator. If you really want to succeed you need to understand your customer, listen the potential customers, and be able to step into customer shoes. Students did believe that after the course, they are more ready to go into work life.

VI. Conclusion

This paper reported the design and implementaton of a software engineering capstone course that infuses design thinking methods and agile practices into the SE project life-cycle. This design was done in order to address the existing gap between the application of design thinking into software development in industry and the skills and competences being taught in software engineering education. This paper presents the methods, results and experiences from this capstone offered in

spring of 2017 as a 16-week course for 29 students. This article illustrates a practical example of the impact of project-based courses (such as capstones) in software engineering education

Design thinking methods were proven to be a hands-and minds-on method to effectively engage students and enhance their experience and improve soft skills (e.g. team working) important in industry. In this course, we used design thinking to prompt project-based learning, requirements elicitation, software design and development. The applied approach allowed students to be self-directed which increased their motivation and, as a result, reduced dropouts to minimum. All student projects surpassed the expectations. Two projects are being continued as entrepreneurial initiatives and one as an open-source project.

In future the course will be improved by 1) inviting industry and local municipality representatives to provide project challenges and participate actively in the client role and 2) improving the link between design thinking and agile software development practices.

REFERENCES

[1] H. Erdogmus and C. Péraire, "Flipping a Graduate-Level Software Engineering Foundations Course," in *Proceedings of the 39th international conference on Software engineering - ICSE '17*, 2017, pp. 23–32.

[2] J. R. Savery, "Overview of problem-based learning: Definitions and distinctions," *Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows*, pp. 5–15, 2015.

[3] S. Howe, "Where are we now? statistics on capstone courses nationwide," *Advances in Engineering Education*, vol. 2, no. 1, 2010.

[4] A. J. Dutson, R. H. Todd, S. P. Magleby, and C. D. Sorensen, "A review of literature on teaching engineering design through project-oriented capstone courses," *Journal of Engineering Education*, vol. 86, no. 1, pp. 17–28, 1997.

[5] J. C. Dunlap, "Problem-based learning and self-efficacy: How a capstone course prepares students for a profession," *Educational Technology Research and Development*, vol. 53, no. 1, pp. 65–83, 2005.

[6] ——, "Changes in students' use of lifelong learning skills during a problem-based learning project," *Performance Improvement Quarterly*, vol. 18, no. 1, pp. 5–33, 2005.

[7] L. Wijnia, S. M. Loyens, and E. Derous, "Investigating effects of problem-based versus lecture-based learning environments on student motivation," *Contemporary Educational Psychology*, vol. 36, no. 2, pp. 101–113, 2011.

[8] T. J. T. F. on Computing Curricula, "Curriculum guidelines for undergraduate degree programs in software engineering," New York, NY, USA, Tech. Rep., 2015.

[9] N. Costa Valentim, W. Silva, and T. Conte, "The Students ' Perspectives on Applying Design Thinking for the Design of Mobile Applications," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. IEEE Press, 2017, pp. 77–86.

[10] R. Razzouk and V. Shute, "What Is Design Thinking and Why Is It Important ?" *Review of Educational Research*, vol. 82, no. 3, pp. 330–348, 2012.

[11] J. J. Winnefeld, C. Kirchhoff, and D. Upton, "Cybersecurity's Human Factor: Lessons from the Pentagon," *Harvard Business Review*, 2015. [Online]. Available: https://hbr.org/2015/09/cybersecuritys-human-factor-lessons-from-the-pentagon

[12] J. Cleland-Huang and M. Rahimi, "A Case Study: Injecting Safety-Critical Thinking into Graduate Software Engineering Projects," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. IEEE Press, 2017, pp. 67–76.

[13] E. Hayes and I. Games, "Making Computer Games and Design Thinking: A Review of Current Software and Strategies," *Games and Culture*, vol. 3, no. 3-4, 2008. [Online]. Available: http://journals.sagepub.com/doi/pdf/10.1177/1555412008317312

[14] W. Akili, "Perspectives on engineering design learning: Realities, challenges, and recommendations," in *Frontiers in Education Conference*. IEEE, 2015, pp. 1–7.

[15] S. T. Acuña, J. W. Castro, and N. Juristo, "A HCI technique for improving requirements elicitation," *Information and Software Technology journal*, vol. 54, pp. 1357–1375, 2012.

[16] J. N. Haack, G. A. Fink, W. M. Maiden, D. McKinnon, and E. W. Fulp., "Mixed-Initiative Cyber Security: Putting humans in the right loop," in *In The First International Workshop on Mixed-Initiative Multiagent Systems (MIMS) at AAMAS*, 2009.

[17] WWF (World Wide Fund for Nature), "Urban solutions for a living planet - Learning cases," Tech. Rep., 2013. [Online]. Available: http://awsassets.panda.org/downloads/urban{_}solutions{_}lc{_}summary{_}web.pdf

[18] A. Chamillard and K. A. Braun, "The software engineering capstone: structure and tradeoffs," *ACM SIGCSE Bulletin*, vol. 34, no. 1, pp. 227–231, 2002.

[19] C. Bastarrica, D. Perovich, and M. Marques Samary, "What can Students Get from a Software Engineering Capstone Course?" 2017.

[20] C. Dean, T. D. Lynch, and R. Ramnath, "Student perspectives on learning through developing software for the real world," in *Frontiers in Education Conference (FIE), 2011*. IEEE, 2011, pp. T3F–1.

[21] H. A. Wayment and K. L. Dickson, "Increasing student participation in undergraduate research benefits students, faculty, and department," *Teaching of Psychology*, vol. 35, no. 3, pp. 194–197, 2008.

[22] B. Lu and T. DeClue, "Teaching agile methodology in a software engineering capstone course," *Journal of Computing Sciences in Colleges*, vol. 26, no. 5, pp. 293–299, 2011.

[23] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99–106, 2012.

[24] T. Smith, K. M. Cooper, and C. S. Longstreet, "Software engineering senior design course: experiences with agile game development in a capstone project," in *Proceedings of the 1st International Workshop on Games and Software Engineering*. ACM, 2011, pp. 9–12.

[25] E. Sweedyk and R. M. Keller, "Fun and games: A new software engineering course," *SIGCSE Bull.*, vol. 37, no. 3, pp. 138–142, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1151954.1067485

[26] N. E. Cagiltay, "Teaching software engineering by means of computer-game development: Challenges and opportunities," *British Journal of Educational Technology*, vol. 38, no. 3, pp. 405–415, 2007.

[27] T. Brown, "Design Thinking," *Harvard Business Review*, 2008. [Online]. Available: https://hbr.org/2008/06/design-thinking

[28] T. Brown and M. Roger, "Design for Action," *Harvard Business Review*, 2015. [Online]. Available: https://hbr.org/2015/09/design-for-action

[29] U. Johansson-sköldberg and J. Woodilla, "Design Thinking : Past , Present and Possible Futures," *Creativity an Innovation Management*, vol. 22, no. 2, pp. 121–146, 2013.

[30] R. Buchanan, "Wicked problems in design thinking," *Design issues*, vol. 8, no. 2, pp. 5–21, 1992.

[31] H. A. Simon, *The sciences of the artificial*. MIT press, 1996.

[32] K. Yamazaki, "Design Thinking and Human-Centered Design - Solution-Based Approaches to Innovation and Problem-Solving in Social Environment," *NEC Technical Journal*, vol. 8, no. 3, pp. 15–19, 2014.

[33] T. Brown and J. Wyatt, "Design Thinking for Social Innovation," *Standford Social Innovation Review*, 2010. [Online]. Available: https://ssir.org/articles/entry/design{_}thinking{_}for{_}social{_}innovation

[34] J. Krüger and J. Müller, "How can Design Thinking improve Software Engineering Processes?" 2014. [Online]. Available: https://www.tele-task.de/archive/video/html5/25024/

[35] O. Sohaib and K. Khan, "Integrating usability engineering and agile software development: A literature review," in *Computer design and applications (ICCDA), 2010 international conference on*, vol. 2. IEEE, 2010, pp. V2–32.

[36] T. Roach, "How to combine Design Thinking and Agile in practice," 2015. [Online]. Available: https://medium.com/startup-study-group/how-to-combine-design-thinking-and-agile-in-practice-36c9fc75c6e6

[37] M. Kern, ""Post-Agile" Software Development," 2016. [Online]. Available: https://www.linkedin.com/pulse/post-agile-software-development-matthew

[38] S. S. Erzurumlu and Y. O. Erzurumlu, "Sustainable mining development with community using design thinking and multi-criteria decision analysis," *Resources Policy*, vol. 46, pp. 6–14, 2015.

[39] D. Chasanidou, A. A. Gasparini, and E. Lee, "Design thinking methods and tools for innovation," in *International Conference of Design, User Experience, and Usability*. Springer, 2015, pp. 12–23.

[40] J. Pruitt and T. Adlin, *The persona lifecycle: keeping people in mind throughout product design*. Morgan Kaufmann, 2010.

[41] J. M. Maness, T. Miaskiewicz, and T. Sumner, "Using personas to understand the needs and goals of institutional repository users," *D-Lib Magazine*, vol. 14, no. 9/10, pp. 1082–9873, 2008.

[42] M. Shin, B.-s. Kim, and J. Park, "Ar storyboard: an augmented reality based interactive storyboard authoring tool," in *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*. IEEE, 2005, pp. 198–199.

[43] J. Arnowitz, M. Arent, and N. Berger, *Effective prototyping for software makers*. Elsevier, 2010.

[44] S. G. Storaas, "Service blueprints and user journeys," Trondheim, pp. 1–16, 2010. [Online]. Available: https://www.ntnu.no/documents/10401/1264435841/Synneva+Storaas+PD9+Artikkel.pdf/6fcfdfd2-75d8-4c03-9a4f-396065445b0e

[45] R. Halvorsrud, R. Halvorsrud, K. Kvale, K. Kvale, A. Følstad, and A. Følstad, "Improving service quality through customer journey analysis," *Journal of service theory and practice*, vol. 26, no. 6, pp. 840–867, 2016.

[46] M. C. Martini, M. A. Smith, and R. J. Youmans, "A comparison of prototyping on paper (pop) software and traditional paper prototyping for developing mobile products with optimal user experience," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 58, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2014, pp. 1849–1853.

[47] J. Purtilo, A. Larson, and J. Clark, "A methodology for prototyping-in-the-large," in *Software Engineering, 1991. Proceedings., 13th International Conference on*. IEEE, 1991, pp. 2–12.

[48] P. Kruchten, *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.

[49] A. Herala, E. Vanhala, A. Knutas, and J. Ikonen, "Teaching programming with flipped classroom method: a study from two programming courses," in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. ACM, 2015, pp. 165–166.

[50] P. Dillenbourg, "The Evolution of Research on Computer-Supported Collaborative Learning: From Design to Orchestration," *Technology-Enhanced Learning*, no. June, pp. 3–19, 2009. [Online]. Available: http://link.springer.com/10.1007/978-1-4020-9827-7

[51] A. A. Gokhale, "Collaborative learning enhances critical thinking," 1995.

[52] A. Knutas, "Increasing Beneficial Interactions in a Computer-Supported Collaborative Environment," Lappeenranta, Tech. Rep., 2016. [Online]. Available: http://www.doria.fi/handle/10024/125665

[53] T. Okamoto, "Collaborative technology and new e-Pedagogy," *Proceedings - IEEE International Conference on Advanced Learning Technologies, ICALT 2004*, pp. 1046–1047, 2004.

[54] G. C. Gannod, J. E. Burge, and M. T. Helmick, "Using the inverted classroom to teach software engineering," *Proceedings of the 13th international conference on Software engineering - ICSE '08*, p. 777, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1368088.1368198

[55] D. Hoyle, "Iso 9000: quality systems handbook," 2001.

[56] ISO, "ISO/IEC 25010," 2011. [Online]. Available: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[57] Usability.gov, "System Usability Scale (SUS)." [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[58] J. Brooke, "SUS - A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[59] R. M. Belbin, *Team roles at work*. Routledge, 2012.

[60] ——, "Team roles at work: A strategy for human resource management," *zitiert in: Teamarbeit und Teamentwicklung*, p. 321, 1993.

[61] G. Gibbs, *Learning by doing: A guide to teaching and learning methods*. Further Education Unit, 1988.

[62] L. J. Mullins, *Management and organisational behaviour*. Pearson education, 2007.

[63] M. Csikszentmihalyi, *Toward a psychology of optimal experience*. Springer, 2014.

[64] B. W. Boehm, "Value-based software engineering: Overview and agenda," in *Value-based software engineering*. Springer, 2006, pp. 3–14.

[65] I. Glover, "Play as you learn: gamification as a technique for motivating learners," 2013.

[66] F. Groh, "Gamification: State of the art definition and utilization," *Institute of Media Informatics Ulm University*, vol. 39, 2012.

[67] F. S. Din, J. Calao, K. Ward, C. Chiong, and C. Shuler, "The effects of playing educational video games on kindergarden achievement," *Child Study Journal*, vol. 31, no. 2, pp. 95–102, 2001.

[68] M. Sanmugam, H. Mohamed, N. M. Zaid, Z. Abdullah, B. Aris, and S. M. Suhadi, "Gamification's role as a learning and assessment tool in education," *International Journal of Knowledge-Based Organizations (IJKBO)*, vol. 6, no. 4, pp. 28–38, 2016.

[69] S. K. Sheth, J. S. Bell, and G. E. Kaiser, "Increasing student engagement in software engineering with gamification," *Department of Computer Science, Columbia University, New York, NY, USA. Columbia University Computer Science Technical Reports. Dpartment of Computer Science, Columbia University*, 2012.

[70] D. Kember*, "Interpreting student workload and the factors which shape students' perceptions of their workload," *Studies in higher education*, vol. 29, no. 2, pp. 165–184, 2004.

[71] C. Bryson and L. Hand, "The role of engagement in inspiring teaching and learning," *Innovations in education and teaching international*, vol. 44, no. 4, pp. 349–362, 2007.

[72] K. Struyven, F. Dochy, S. Janssens, and S. Gielen, "On the dynamics of students' approaches to learning: The effects of the teaching/learning environment," *Learning and Instruction*, vol. 16, no. 4, pp. 279–294, 2006.

[73] P. Ramsden, *Learning to teach in higher education*. Routledge, 2003.

[74] J.-R. Ruiz-Gallardo, S. Castaño, J. J. Gómez-Alday, and A. Valdés, "Assessing student workload in problem based learning: Relationships among teaching method, student workload and achievement. a case study in natural sciences," *Teaching and Teacher Education*, vol. 27, no. 3, pp. 619–627, 2011.