Interdisciplinary System Courses – Teaching Agile Systems Engineering

Andreas Seitz, Mariana Avezum, Bernd Bruegge
Chair for Applied Software Engineering
Technische Universität München
Munich, Germany
seitz@in.tum.de, m.avezum@tum.de, bruegge@in.tum.de

Stefan Wagner
Institute of Software Technology
University of Stuttgart
Stuttgart, Germany
stefan.wagner@iste.uni-stuttgart.de

***Abstract*—With the advent of technologies like the Internet of Things, Industry 4.0 and Cyber-Physical Systems, many software engineering courses turn into system engineering courses. Recent advances in technologies such as 3D printing and low-cost micro controllers enable to teach agile hard- and software co-design in system engineering courses. In this paper, we describe *Interdisciplinary System Courses* (ISC) – a teaching approach based on interdisciplinary projects, light-weight agile techniques and solving real problems by integrating industry customers. We describe our experiences from an exploratory case study where we applied ISC in a two-week international summer school with a customer from the aerospace industry. We derive a set of hypotheses on the effects of ISC.**

***Index Terms*—interdisciplinary, teaching, agile, hardware, software, systems course**

I. Introduction

With the advent of technologies such as Internet of Things, Industry 4.0 and Cyber-Physical Systems, systems engineering integrates evermore software engineering aspects. These systems do not only consist of software but also various types of hardware. This means that we not only have software engineers in the team but require interdisciplinary work, whose development process often is different than software.

At the same time, these software/hardware systems are more and more under the same pressures as pure software systems to be able to innovate quickly in short cycles. In software engineering, this gave birth to agile software development, whose application to hardware development [1]–[3] is still in its infancy. This makes working together in an interdisciplinary project difficult.

One approach to reduce this challenge in industry is to familiarize systems engineering students with these agile concepts still in their studies. To let participants already experience these difficulties and to work on making agile hard- and software development a reality, we propose the format of *Interdisciplinary System Courses* (ISC). The idea is to put participants with different backgrounds – such as computer science, electrical engineering and mechanical engineering – in a realistic project setting and let them work together in an agile way. The format of a summer school outside normal university constraints allow us to work together in a focused way while bringing in participants with various backgrounds and experience levels.

The ISC concept addresses the following three challenges:

1) Interdisciplinary teams with different skill sets, ways of working and terminology
2) Producing hard- and software prototype system in a few days up to a few weeks
3) Solving real problems by integrating industry as customer and domain experts

In order to achieve these goals, we will present how we implemented ISC through a 2-week course, overcame communication problems, and taught the participants to implement a system consisting of both hardware and software components.The paper is structured as follows: Section II deals with the basics of ISC and shows relevant fields of research. In Section III, we explain the concept of ISC and its structure. In the case study in section Section IV, we show how we applied ISC in a two-week summer school. The exploratory study enabled us to evaluate the concept of ISC. We present the general findings and derived hypotheses. The paper ends with the conclusion in Section VI, where we summarize the contributions and give an outlook on future work.

II. Foundations

An established approach for the introduction not only of software engineering but also of concepts such as Scrum are capstone courses, which have been around for a while [4] [5]. While these have been known to have good results, introducing interdisciplinary systems engineering and hardware components can directly conflict with Scrum concepts such as incremental and continuous iterations. To bridge these challenges, we look into relevant foundations to support the organization of the ISC course.

A. Systems Engineering Teaching

After the publication of the Agile Manifesto in 2001 [6], different frameworks appeared as how to implement these values in practice, the most common of which is Scrum. While these frameworks spread both in software engineering and in teaching, the shift has been slower in hardware development. Hardware development has traditionally used development processes that go deep into the requirement elicitation, and many industries use the V-Model as it allows the system and its interfaces to be completely defined before the system is

assembled [7]. This critical need of the industry has influenced systems engineering education, and university capstone courses usually follow the same development process. A review of engineering design capstone courses can be found in [8].

These two different approaches pose a challenge for interdisciplinary courses as participants may not be used to communicating or working with a different development process. While there have been teaching projects where these differences are managed, such as [9], the system developed in that study was a highly complex one, which makes it harder to reproduce the situation. As for industry, it has also tried to adapt agile practices for hardware development, however with mixed results [10]. One agile framework that deals with different development cycles, and the integration of the components thereof, is Scrum of Scrums.

B. Scrum of Scrums

Systems that involve components from multiple disciplines, and both hardware and software subsystems, will unavoidably involve a certain complexity and larger groups than are usually found in Scrum teams. To adopt Scrum to larger teams, the Scrum of Scrums method was introduced.

As the name suggests, this consists of layering different Scrum teams into a new Scrum group, where representatives of the lower-layer Scrum teams report their status in a second daily Scrum meeting [11]. This approach helps decompose not only the developed system into smaller components, but also the dependencies between the different teams and people involved. At the same time, it also ensures that the entire complete team is kept up to date with the project's status.

Big dependencies on other components is a critical aspect of hardware development, and thus, reducing these as much as possible makes it easier to introduce a faster development process. Furthermore, the Scrum of Scrums framework also allows for different development cycle lengths for each of the subsystems involved in the project, which then allows the hardware and software systems more flexibility in planning for a development phase that better suits their capabilities and needs.

C. Tornado Model

When working with different independent teams, it is essential to keep a close eye on system integration. As will be presented in Section IV, our summer school case study used the Tornado Model [5] to achieve this. It focuses on using a scenario-based development approach, useful for teaching courses, where participants work toward a presentation at the end of the project.

By selecting a visionary scenario that presents features of all components in the system, not only does a complete prototype demonstration become possible, but also the minimal required interfaces become clear. To further understand and correctly implement these interfaces, students from each sub-team are tasked to define and document these in system-wide models, which can be accessed by all sub-teams. Encouraging the participants to present formal UML models can present a good learning opportunity.

Furthermore, they are also presented with informal methods, which serve the same communication purpose, while giving the participants a greater creative freedom, and sometimes being easier for industry clients to understand. The Tornado Model suggests that participants prepare a Software Theater of their results [12], which presents an opportunity to record the presentations, and use it as a communication mechanism for all project stakeholders.

III. Interdisciplinary System Courses (ISC)

In the following section we describe the concept, structure and core values of ISC.

A. Features & Design

The aim of ISC is to bring together participants from different disciplines to work together on a challenging problem. The chosen course format is not a traditional one but a condensed course, which helps the organisational aspects of having students from different degree programs work together. The case study presented in Section IV takes place during the semester break and is aimed at motivated and talented students who want to expand their horizons. ISC is not part of a curriculum. Students can apply for this programme and will then be selected based on their performance and a letter of motivation. Travel and accommodation costs will be covered for the participants.

ISC is characterized by the following goals and design decisions:

(1) Interdisciplinarity: The core of ISC is interdisciplinarity. It is no longer sufficient for participants of individual disciplines to solely work in their one domain. It is necessary for them to be able to look beyond their subject areas and collaborate with other participants in other subject areas. Novel technologies and the fusion of hardware and software require cooperation across disciplines. The challenge here is different knowledge and processes. In addition, projects are becoming more and more demanding and cannot be realized with the knowledge of a single discipline.

(2) Agile Light-Weight Development Process: With regard to the process, we deliberately opted for a loose, adaptable and flexible process. We believe that this freedom can make cooperation more effective. In spite of all this freedom, there are also fixed deadlines and delivery times that the participants have to meet. In ISC we apply the concept of Chaordic Learning [13]. We give the participants the freedom to organize themselves, but create order through punctual structures. The participants are expected to produce a light-weight prototype of the system by the end of the course.

(3) Challenging Problem of Industrial Partners: To be able to challenge and motivate the participants of ISC, a real problem with innovative solution possibilities is required. The solution must require technical expertise from different fields and must be dynamically extensible. ISC requires industry partners to formulate and define the problem on the basis of

their expert knowledge. Furthermore, they should be available for the preparation meeting, design review and final customer acceptance test. Industry partner integration is also seen as a motivation source as the participants know that their work will be further used.

B. Skills & Techniques

To achieve these goals, the participants required core skills and tools which were taught throughout the course.

Communication & Soft Skills: An important aspect to address the challenges of interdisciplinary is efficient communication. The terminology and prior knowledge of the participants are initially heterogeneous. Their different backgrounds result fundamentally different ways of working. While computer science and software engineering participants are usually familiar with agile methods, other participants most often have no such experience, and thus it is important to provide them with a relevant theoretical overview and materials, in order for the necessary terminology to get better understood. Furthermore, ways and means must be found to facilitate cooperation between the disciplines.

Videos as Communication Models: A saying goes, a picture says more than a thousand words. In ISC the motto is: a video says more than a thousand words. In the course of ISC, we strongly rely on the use of videos as communication models.

First, it supports the problem description. Today, the web offers a variety of videos that can be used to illuminate and underscore problems, especially useful for cases where logistic or organisational constraints make it hard for the students to physically see the problem. Many problems today are solved by analog or conventional approaches, which can be shown to the students through videos. In order for the students in ISC to reach their goal and tackle these problems digitally, it is important that they understand the problem domain first.

Second, videos are also used to visualize solution concepts. The participants of ISC record their ideas in videos and share them with their team members and project partners from industry.

On-Site Application Domain Expert: The participants of ISC are solution domain experts but usually have little prior knowledge of the application domain. To close this gap, ISC relies on the fact that an expert of the application domain is available for questions. This expert can thus influence the developed solutions and ensure that the developed systems also meet expectations. In direct, non-formal communication, the requirements of a system can be constantly updated and checked. Having this expert physically present during the ISC also ensures efficient communication channels.

C. Infrastructure & Environment

An essential part of ISC is the working environment and infrastructure. ISC intends to take all participants out of their familiar working environment and to accommodate them together in one place for a certain period of time. This way, distractions can be minimized, and the participants can fully concentrate on dealing with the problem.

In addition to the working environment, the infrastructure is also important. ISC assumes that participants who have everything they need to work can also work better and more efficiently. Depending on the problem statement, this can imply special hardware and software. Since ISC involves elements of systems engineering, it also covers areas such as agile manufacturing. This requires special infrastructure such as 3D-printers and hardware prototyping material. Micro controllers such as Arduinos and RaspberryPIs are often necessary to form the interfaces between the hardware and software components.

D. Structure

ISC is divided into 3 phases: (1) Preparation Phase, (2) Development Phase and (3) Integration Phase (cf. Figure 1).

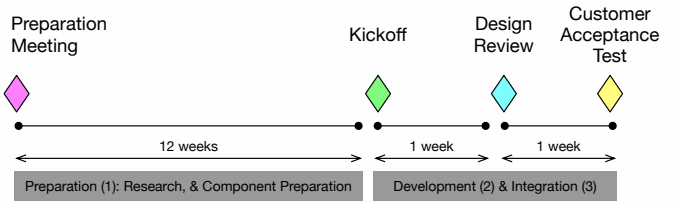

Fig. 1. Timeline of ISC: The preparation takes place before the actual course starts. During the period of the ISC course, development of the subsystems is first carried out and then integrated.

In the course of ISC different events with specific meaning are planned. We explain in detail the different events:

Preparation Meeting: The Preparation Meeting is the first time that the participants of an ISC course as well as the organizers and industry partners come together. The aim of the meeting is to get to know each other, to understand the problem, and to distribute work packages for the participants. The industry customer presents the problem to be dealt with in the form of scenarios. Before this meeting, the industry partners and organizers jointly create an initial top-level design of the architecture which is a key discussion point of the meeting. This architecture is then presented to the participants of ISC for the first time. Starting from the architecture, technological requirements and subsystem are defined. These form the basis for the teams that have to complete preparation tasks in the next 12 weeks, and thus the initial team allocation is done during the Preparation Meeting as well.

Kickoff: After the arrival at the location where ISC takes place the Kickoff Meeting happens. The problem statement will be discussed in more detail on the spot. To support team building and create a nice atmosphere, an ice breaker is done with all participants. Ideally, the icebreaker can be based on the topic to be worked on and should clarify the motivation and the understanding of the problem. Subsequently, each sub-team presents the results of the project work they have achieved in the past 12 weeks during preparation. The focus will be on a demonstration of what has already been achieved and how it should be incorporated into the common vision.

Design Review: The Design Review takes place halfway through the ISC course. All participants, application domain experts, and organizers come together to discuss the progress of the project. Depending on their time availability, industry partners may attend this or not. The focus is on design. Each sub-team will present its progress and any challenges they may face and give a glimpse of what they expect to achieve in the time remaining.

In preparation for this presentation, a video trailer can be produced. This trailer serves as a means of communication between participants and industry partners, to synchronize the vision of both parties. The trailer can also be sent to the industry partner in case they could not attend the meeting.

Customer Acceptance Test: Finally, there will be a joint event: the Customer Acceptance Test. This event serves to present the final results achieved by the participants and to have them accepted by the customer/industry partner. The form of the presentation is left to the participants. While slides can be used for communication, the main focus is to demonstrate a demo scenario and the functionality of the developed system.

E. Process

Between the fixed events, ISC provides a lightweight agile process. It is divided into three sprints. Sprint 1 between the Preparation Meeting and the Kickoff Meeting, Sprint 2 between the Kickoff Meeting and the Design Review, and Sprint 3 between the Design Review and the Customer Acceptance Test. During ISC, a typical day starts with a group breakfast, followed by a Daily Stand Up Meeting.

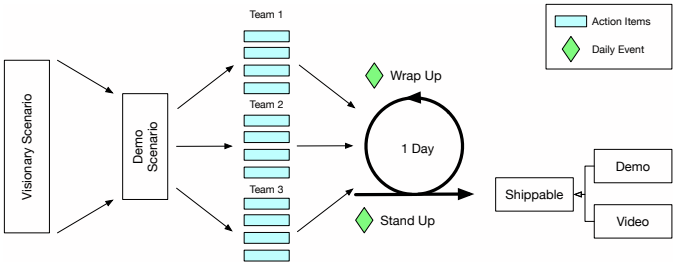

Fig. 2. ISC Process: Starting from a Visionary Scenario, this is broken down to a Demo Scenario. This demo scenario will be divided into action items which will be processed by the different sub-teams. Daily Scrum Meetings take place in the morning with each sub-team.

In this meeting everyone individually reports the progress of the previous day, the impediments they have and promises what they want to achieve in the course of the day. In this context, it is also worth mentioning that between the working days there are also days for leisure time. The combination of work and leisure is important since it motivates the participants.

The process is visualized in Figure 2. While the division into sub-teams can be loosened, it is important for each participant to have well-defined tasks, that do not rely too much on other subsystems, as that would hinder the agility of the process. The visionary scenario describes the functionality to be provided by the future system [14]; this can be tailored to a demo scenario. The demo scenario is an excerpt from the visionary scenario that can be realized in the short time frame of the course and can also be demonstrated. Action items then result from the demo scenario. These should be realizable within one day. Unlike Scrum, the daily result of ISC is not a potentially shippable product but either a demonstration or a video demonstrating the functionality.

IV. Case Study

In order to implement the objectives described in Section III, Chaordic learning [13] was used to address a problem statement presented by an aerospace industry partner. While the ISC instructors did provide general topics to be researched before the 2-week course and divided all the participants into sub-teams, the exact definition of who was responsible for what in each sub-team, which technologies to use, and what tasks to prioritize was self-organized by the participants.

In the Kick-off meeting the participants were presented with a problem statement from an aerospace industry partner and used that as a common reference to set the tasks developed during the 2-week course. The industry partner wishes to assist rescue agencies such as the red cross after any type of natural disaster, and thus the ISC participants should develop a Multi Operational Drone Collaboration Platform (MODCAP).

Problem Statement: Agencies such as the Red Cross need to coordinate and plan relief efforts after disasters. The focus of the project developed in our ISC implementation was to collect and analyze data gathered by any available fleet of drones in disaster areas.

MODCAP is a platform that orchestrates a collection of drones to perform search and rescue missions after natural disasters. Furthermore, the data collected by the platform should be used to map geographical changes caused by these disasters, as well as to help in the survivor rescue operations. To further assist the survivor rescue operations, the MODCAP platform should also allow for the modular integration of any smart wearable device the victim may be using, such as Smart Watches, or any transceivers, as is common for skiers to use on avalanche-prone areas.

A. Infrastructure & Environment

The ISC concept was implemented during a 2-week course where internet and hardware accessibility were limited, which is why most of the necessary infrastructure was transported by the organizers to the location. In this section, we explain how the environment and technical infrastructure made available influenced the implementation of the course.

To facilitate the adoption of agile manufacturing techniques and simultaneously keep a light-weight development process, rapid prototyping equipment was made available to all of the participants. This includes, but is not limited to, a 3D-printer and prototyping electronics such as Raspberry Pis and Arduinos. Since the problem statement implemented by the participant's had some drone components, it was important for the available electronics to be able to interface to the drone's software development kit. This hardware allowed for the different teams to quickly test and iterate their components.

One example of hardware iteration can be seen in the scoop mechanism developed for the drone. The idea behind the mechanism was to modularly connect a component to the drone so that samples of the areas affected by the disaster could be collected. As this mechanism was to be attached to any available drones in the area, it was important to keep it as modular as possible, but also stable enough to grab any necessary samples of the area. To solve this, the participants initially designed a sample mechanism, designed for a DJI Phantom 4, however, the first iteration of the component did not really fit the drone. By the end of the first iteration, they had adjusted the size and tested the component in flight, however, the stability of the system proved to be insufficient to actually snatch any sample from the ground. This was then successfully fixed in the following phase of the project.

An important aspect of agile development is the communication between developers. While this is classically supported through the use of online issue tracking tools, this was not possible due to the limited internet connectivity of the location where we carried out ISC. Furthermore, the fact that all participants were located in the same place meant that they could exchange much information through informal meetings, which greatly helped communication and fast decision making.

To allow all participants to keep track of the system's status, pen and paper were used in the place of online issue trackers, and a Kanban board with Backlog, In Progress, and Done columns was set up, as can be seen in Figure 4. Post-its were used for each task, where the exact action item and person responsible for it were written down.

At both the Design Review and Customer Acceptance Test milestones, a system scenario was presented. In the Design Review, the status was discussed between all participants and the organizers, and a video trailer was used to demonstrate the scenario envisioned by the end of the two weeks. This really helped the participants to know which tasks to focus on during the second week of the project, as these were the exact features to be demonstrated to the industry partner client at the Customer Acceptance Test.

It should also be noted that our industry partner appreciated the video trailer, as it allowed them to easily explain to objectives of the system to several other stakeholders inside of the company, who were not involved in the process.

B. Team Work

The fact that the ISC participants came from different study courses meant that they individually had very different skill sets. Furthermore, the Problem Statement presented to them required different backgrounds and a considerable amount of both hardware and software components.

As presented in Section III, all participants of the summer school met for a Preparation Meeting a few weeks before the Kick Off. It is relevant to note that the team assignment that took place in the Kick-Off Meeting, was not done based on skill set, but on the participant's personal preference.

While a skill set based assignment could make the development easier (and perhaps even produce a better final system),

Fig. 3. Work Environment

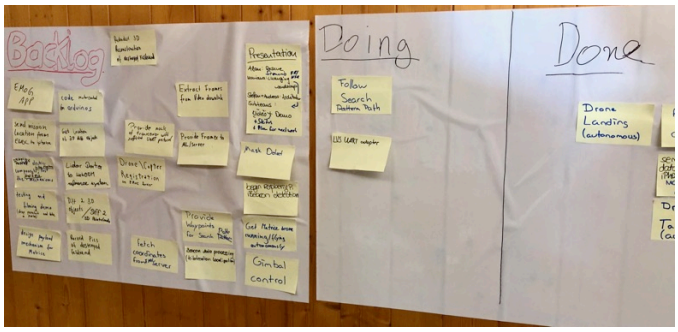

Fig. 4. Kanban with Pen & Paper

we believe that a team assignment based on personal preference increases team motivation and allows the participants the opportunity to learn new skills based on what they enjoy. Furthermore, the fact that this team assignment was made several weeks before the kick-off meant that the participants had time to research, prepare, and learn any fundamental skill sets that would be required during the two development weeks.

To get everyone onto the same page as far as what was possible or not during the two weeks, right on the first day of the ISC, each sub-team presented their research results. These were short 10-15min presentation from each sub-team, where some initial feedback was already provided as to how far their expectations were realistic. These presentations lead to important discussions on necessary frameworks, interfaces, and further capabilities. We have deliberately not defined a format here in which way the presentation should take place, to allow for self-organization of the different teams.

Another activity that was important at the beginning of the ISC was the Ice-breaker. The participants simulated a search and rescue operation, with the use of an analogue transceiver. This helped them not only bond more and overcome their different background, but also gave them insights in the current state of the art of the systems they would be working on.

V. Case Study Evaluation

In this section, we will look at the exploratory study we conducted during the case study and evaluate what the participants thought of the ISC. Based on a questionnaire sent to the participants, we present the results below and discuss the participants' feedback.

A. Objectives & Design

We carried out an exploratory study and collected data from the participants by means of a questionnaire to verify our assumptions as well as to validate whether the goals of the applied methodology are met. The questionnaire was distributed to the participants shortly after the Design Review, around halfway through the course. The questionnaire focuses on the following research questions:

1) **Interdisciplinarity**: Effects of different backgrounds of participants on how they work together.
2) **Agile Light-weight Process**: Effects of the chosen engineering approach regarding high-level scenarios, daily scrum meetings, demos, simple architecture descriptions
3) **Software Theater**: The effects and implications of using videos during the course to visualize problems. The videos serve as a communication model between the participants and the industry partners.

B. Questionnaire

Altogether we distributed the questionnaire to 19 participants, all of whom answered. The questionnaire contains 29 questions which are divided into 4 free-form questions, 24 Likert item questions and 1 multiple-choice question. The questions are divided into the subject areas: interdisciplinarity, process model and utilization of videos within the course. The applied Likert scale ranges from 1 (strongly agree) to 6 (strongly disagree). The mean and median values in the following section refer to this scale.

C. Results

In the following, we describe the answers to the general and to each research questions, for which we then propose hypotheses that are suggested by the data.

1) General Questions: Of our ISC participants, 26% have never previously worked with participants from other departments or disciplines. For 16% it was their second collaboration and 58% had already collaborated once with participants of other departments or disciplines in projects before ISC.

Figure 5 shows the self-assessed learning of the participants with regards to communication, programming, prototyping, system design and team work. A learning experience could be established across all points. However, it is also clear that ISC is not a programming course, but a systems engineering course where the focus on interdisciplinarity and team collaboration is more important than the actual implementation.

The answer to the question whether the participants would recommend the ISC summer school to other participants is extremely clear. Here, 16 out of 19 participants absolutely agree and none of the participants disagrees with this opinion. We regard this as a success of the overall concept.

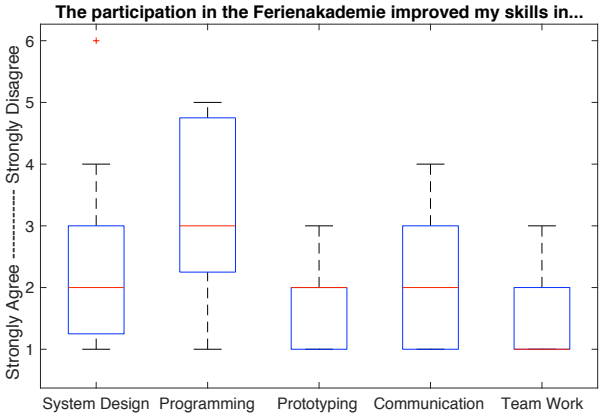

Fig. 5. Skill Improvement Box-Plot

2) Interdisciplinarity: The participants overwhelmingly stated that the communication with participants from other areas was easy (median = 6, min = 2, max = 6). We also asked how communication on the sub-team level worked. Communication within the sub-teams (median = 5, min = 2, max = 6) as well as with the other sub-teams (median = 5, min = 1, max = 6) were seen as working well.

The open question about describing the challenges and problems of working with fellow participants, however, reveals challenges: The participants report that it was difficult to define a common understanding of the tasks and problems (e.g. "differing implicit assumptions"). We also observed a heterogeneous previous knowledge that could be overcome through the course of time ("Unclear what skills people have"). Yet, several participants also reported no problems.

While the ISC structure probably contributed to the interdisciplinary communication flow, there are further relevant aspects that may also have avoided problems in this regard. The free-text answers of the questionnaire indicate that the location played an important role in relaxing the students, and that the free time activities helped them bond together. While an ice-breaker is part of the ISC structure, currently no data is available whether that was enough to reduce communication problems, or to what extent other activities contributed compared to it. The importance of the environment, however, is supported by the fact that all participants of the questionnaire agreed with the statement "You could work well in the created working environment (atmosphere, room, food)."

While the cause behind the improved interdisciplinary communication may be disputed, our data indicated that cross team communication in the ISC worked well. Interdisciplinary communication and work is commonly problematic in other contexts, for example, because of misunderstandings [15], [16], and we see little problems due to interdisciplinarity in ISC. Therefore, we formulate the following hypothesis:

> Hypothesis 1: The ISC concept reduces interdisciplinary communication problems.

3) Agile Light-Weight Process: We also evaluated the applied agile process. The whole process was considered suitable for the project (median = 5, min = 3, max = 6). In the free text answers, the participants suggest that some of the meetings lasted too long. There are also mentions that an earlier focus on integration would be useful.

When looking at the system iterations in particular, participants agreed that these were relevant to the development process (median = 2, min = 1, max = 4). Given the fact that students outside of the software engineering environment had limited previous access to agile processes and iterations, this can be seen a meaningful contribution of ISC. Figure 6 shows the detailed distribution of answers.

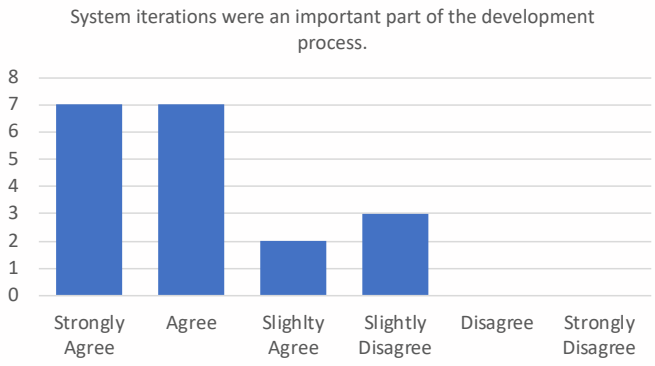

Fig. 6. System Iteration Value

The free text answers suggest that participants considered the meetings communicating the status as useful and important. It is however relevant to note that not all days spent during the ISC were work days, and there were some entire leisure days as well. In these cases, the morning meeting in the following day was sometimes criticised, as the participants had not achieved any work progress during the leisure day, and the meeting content thus became repetitive.

One aspect of agile development that was not adequately accessed in our evaluation were the problems and benefits of the Scrum of Scrums element. While both internal sub-team meetings and meetings with all sub-team took place, it is possible that the latter happened with too many participants, thus diminishing it's efficiency. Perhaps a more rigorous, with fewer people, implementation of the Scrum of Scrum process could have aided in the integration problems, however, it would likely also have diminished the learning effect, by limiting the number of students receiving input from other sub teams.

Furthermore, the questionnaire also suggest insights as to why the system integration became problematic at the end, even when the iterations were evaluated as useful. While the interfaces between each sub-team was defined, and the participants did know what they had to deliver for the demo-scenario, the data received from the external drone control library was different than expected, and at the end of the ISC there was not enough time to fix this. Had integration begun earlier the actual values would have been received earlier, and perhaps the system could have delivered a better performance.

> Hypothesis 2: System iterations support development, but complete subsystems decoupling hinders integration.

4) Problem Statement Understanding: Due to the different background from the participants, common understanding of the problem statement was of special importance. In the questionnaire, we asked how the different communication approaches facilitated this. We initially communicated the problem in written form as a problem description. 74% of the participants agreed on the importance of this.

The ice-breaker activity previously mentioned, further served the double function of showing the students how search and rescue operations are manually done today. It is relevant to note that this was in the beginning of the course, and thus, the general understanding of the problem was somewhat vague. By providing the students with a transceiver to find hidden objects in a field, the students both had fun, and learned something about the real world use of the system they were to develop.

The introduction of software theater aimed to further support the understanding. The participants also watched videos that showed them how search and rescue operations currently function, and this showed to be a good communication tool. 78% of the participants approved of the use of video tools.

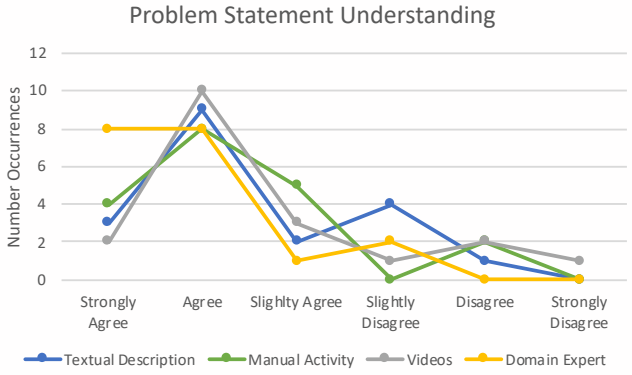

Fig. 7. Value of Each Understanding Technique

Figure 7 depicts the student's assessment of the usefulness of each of the different techniques used. As can be seen, the tool which showed most useful for problem statement understanding was the exchange with the expert from the application domain (median = 2, min = 1, max = 4). While this may also be due to the fact that the remote location of the course made the expert a more reachable resource, wee postulate that this finding may perhaps be generalized.

> Hypothesis 3: Exchange with a domain expert encourages system understanding and problem solving.

D. Discussion

All in all, we can say that the assumptions we have made regarding the ISC concept are working out well. We discuss the three goals described in the beginning and the feedback from both the participants and industry partners.

As far as interdisciplinary teams go, both the quantitative and qualitative feedback from the participants show that they learned how to communicate with people from other backgrounds. Our industry partner also mentioned that the fact that participants from different disciplines participated in the project as interesting for them.

At the end of ISC, the state of the developed system was only partially working. While the individual sub-teams mostly achieved their goals, the integration process required more time than was available in the end, thus resulting in one part of the demo scenario not working. While this was of concern to our industry partner, who was interested in the actual achieved results, it was not critical for the participant's view, as they did learn from both the process and the non-performance.

Last but not least, the collaboration with our industry partner proved to be motivating for the participants. The visit from the industry representatives in the Client Acceptance Test also was a good opportunity for the participants to understand how these systems are developed and considered in an industry context.

While the results of our case study can only be generalized with caution, they do suggest that the approach presented by ISC can stimulate more effective interdisciplinary teaching.

VI. Conclusion

In this paper, we present an approach to teach Interdisciplinary System Courses. In a case and explorative study, we show the applicability of the method. ISC focuses on the three core areas interdisciplinarity, lightweight agile development process and real problem solving by having an industry partner. The interdisciplinary aspect is achieved by focusing on the parallel development of hardware and software components, the communication between individual sub-teams as well as collaboration between these.

The case study and its evaluation show that the concept is applicable and adds value for both the participants and the industrial customers. Although the developed system leaves room for improvement, the realization of ISC and knowledge gained by the participants has proven to be a success.

The paper explains the concept of ISC to make it feasible for other universities and organizations to apply the method. We plan to continue running ISC courses in the coming years and want to apply the knowledge and experience to further improve the teaching of these innovative course formats.

References

[1] P. M. Huang, A. G. Darrin, and A. A. Knuth, "Agile hardware and software system engineering for innovation," in *Aerospace Conference, 2012 IEEE*, pp. 1–10, IEEE, 2012.

[2] T. Punkka, "Agile hardware and co-design," in *Embedded Systems Conference*, 2012.

[3] S. Wagner, "Scrum for cyber-physical systems: a process proposal," in *1st International Workshop on Rapid Continuous Software Engineering, RCoSE 2014*, pp. 51–56, ACM, 2014.

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

[5] B. Bruegge, S. Krusche, and L. Alperowitz, "Software engineering project courses with industrial clients," *Trans. Comput. Educ.*, vol. 15, pp. 17:1–17:31, Dec. 2015.

[6] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, "Manifesto for agile software development," 2001.

[7] A.-P. Bröhl, *Das V-Modell: Der Standard für die Softwareentwicklung mit Praxisleitfaden*. Oldenbourg, 1993.

[8] 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.

[9] L. Boskovski and M. Avezum, "Combining hardware and software development: A case study on interdisciplinary teaching projects.," in *Software Engineering (Workshops)*, pp. 12–15, 2018.

[10] M. Glas and S. Ziemer, "Challenges for agile development of large systems in the aviation industry," in *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pp. 901–908, ACM, 2009.

[11] L. Faria, "Scrum of scrums: Running agile on large projects," *Obtenido de scrumalliance. org, Junio*, 2013.

[12] S. Krusche, D. Dzvonyar, H. Xu, and B. Bruegge, "Software theater - teaching demo-oriented prototyping," *ACM Trans. Comput. Educ.*, vol. 18, pp. 10:1–10:30, July 2018.

[13] S. Krusche, B. Bruegge, I. Camilleri, K. Krinkin, A. Seitz, and C. Wöbker, "Chaordic learning: A case study," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*, ICSE-SEET '17, (Piscataway, NJ, USA), pp. 87–96, IEEE Press, 2017.

[14] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns and Java*. Prentice Hall, 2009.

[15] E. Cooley, "Training an interdisciplinary team in communication and decision-making skills," *Small group research*, vol. 25, no. 1, pp. 5–25, 1994.

[16] T. W. Reader, R. Flin, K. Mearns, and B. H. Cuthbertson, "Interdisciplinary communication in the intensive care unit," *British journal of anaesthesia*, vol. 98, no. 3, pp. 347–352, 2007.