Combining Hardware And Software Development: A Case Study On Interdisciplinary Teaching Projects

Ljube Boskovski
Technical University of Munich
Munich, Germany
ljube.boskovski@tum.de

Mariana Avezum
Technical University of Munich
Munich, Germany
m.avezum@tum.de

***Abstract*—Studies have shown that students retain more information when learning by doing, with practical projects. After graduation, students go on to work with real world projects, which often involve more than one discipline, and the components not covered in their university courses must then be learned on the job. Interdisciplinary university projects are often hard to implement due to lack of collaboration between respective parties, as well as different work practices used in each organization involved. While many advances have been made in teaching agile development to software engineering students, members of other faculties often have different, incompatible, work practices that require structured design and testing processes. We will show how incentivizing students from such different backgrounds to work together can have a huge impact in their learning experience and analyze multiple case studies of such interdisciplinary projects. Decisions such as team structure, collaboration, interdisciplinary change management and task prioritization will get evaluated based on the necessary complexity, and good practices will be drawn based on the results of the presented case studies.**

I. Introduction

Capstone courses have had several successes in teaching agile project development to computer science students in universities around the world [1], as well as in introducing students to the challenges and benefits of working with real industry partners [2]. These courses, however, mostly involve one specific discipline, and few effort have been made to motivate university students from different faculties or areas to work together. This is especially difficult due to the over-specialization that some students face in university.

Different reasons can be found as to why such interdisciplinary approaches are hard to implement. Beyond the fact that the necessary project partners often have little experience working with each other, they may not even know each other and find it hard to communicate. This distance becomes even bigger when applied to student teams. Moreover, when designing interdisciplinary systems to be developed by students, some design goals present themselves, which may be in direct contradiction with each other. While it is desired for the sub-teams in the different faculties to collaborate and work together in order to enhance the working experience, it is essential to limit the dependency (and thus effect) that they have on each other, as it must always be assumed that part of the designed system may fail.

The following paper presents the challenges, and approaches used in the WARR Hyperloop project, and deduce from that a few lessons learned from the team at the Technical University of Munich, where an attempt was made to allow 30 students from different levels and faculties to work together, in a project that heavily involves both hardware and software components. The project involves different iterations with evermore refined design requirements. After receiving instructor approval that the proposed design is viable to build, the students ultimately construct a physical hardware prototype, which participated in an international competition. All the while the team members, who were from seven different faculties, had to work collaboratively with very tight deadlines, which made sure that each individual had the highest interest in the rest of the team completely understanding their work.

Furthermore, we also give some project management examples of when and where problems occurred during the described course, as well as which techniques were used to mitigate it. Section II describes further details on how problem decomposition into smaller tasks can reduce the impact one group of students has on another, while still ensuring that they need to work together for the integration of the entire system. We will then finish by presenting some lessons learned from our experiences. As society changes, it is increasingly important to know how to communicate the skillset learned throughout life to people from different backgrounds, and to be able to learn from the experiences from others, which means that working with people from different backgrounds can be both a teaching and learning experience to all parties involved.

II. Challenges

One of the main challenges presented when organizing an interdisciplinary educational project, is how to organize the project and team so that students from all different areas can benefit from the interdisciplinary aspect and learn additional material without being blocked by other team members. This is only possible to attain with a clear team structure and responsibility definition between sub-teams, as team structures often define communication channels [3].

An example of how important such communication is, can be found in the battery management system of the second WARR Hyperloop pod, developed at the Technical University of Munich. The interior of the battery compartment had to be pressurized to about atmospheric pressure, since the cuboid shaped battery shells would not withstand the force to inflating in vacuum and thus possibly incinerate themselves. Furthermore, the batteries power the sensors and actuators of the pneumatic system, which controls the pressure levels of the battery compartments, among other subsystems. Notice how there is a cyclic dependency chain between the pneumatics system and the battery management system, which automatically leads to a high information flow between the according engineers. All in all, an alternated iterative work flow can be observed, which consists of requirements being passed along from one part of the team to the next, while always ensuring that the latter has enough action items to work on independently. With this information cycle, not only the development efficiency is kept high, but also reciprocal functionality of the software and hardware is assured constantly.

While this collaborative work is great for the educational process, much attention is necessary so that it doesn't extrapolate time and budget limitations. Although time constraints are usually given by the semester dates at any given university, budget may be harder to manage. Any project that involves (buying) hardware components inevitably requires more money than software-only projects. How much money actually is necessary obviously depends on the complexity of the project and the number of students involved, but if this complexity starts to get too big, it may make sense to introduce some finance and/or business students to the project, to help manage and acquire any financial resources necessary. This, in turn, adds one more interdisciplinary aspect, by having students manage their own budget, although it could add more external dependencies.

Our experience shows that one of the biggest challenges in setting up an interdisciplinary teaching project is how to manage the risk that inevitably remains due to dependencies inside and outside of the project. As any given risk can be accessed by considering its probability and severity [4], it generally is the potential of gaining or loosing some kind of value. Even if the probability is very low, bad fortune can lead to a massive drawback. Software engineering courses often solve testing by integrating automatic testing frameworks into the development process, which ensure that the code developed by students adheres to the necessary outputs and standards [5]. Such continuous testing, however, is not completely possible for hardware components, which can become a big dependency source.

For example, an external hardware supplier canceled the delivery of multiple essential components for the prototype of the WARR Hyperloop II team at a late point of time. As a direct consequence, the team had to quickly contact a new supplier, which lead to a price advancement of about

Fig. 1. The WARR Hyperloop II Team at their team building event at Tegernsee, Bavaria. The team divided into sub-teams which competed in building the most precise and powerful catapult out of limited resources together.

€ 40,000 and an additional delay of two weeks. While our experience shows that students can become very motivated to solve problems when others depend on the results of their work and vice versa, this may not always be possible. If a team later in the dependency chain is prevented from working because of the results of their predecessors, this may have big consequences not only for motivation, but possibly also on several organizational aspects. Therefore it is important to strengthen the feeling of belonging to the team and build up trust and respect to the others very early, for example with a team building event as seen in Figure 1.

All of these challenges go a long way to show how important communication is, but this can become quite hard when you have team members with different backgrounds, that may understand completely different concepts under similar nomenclature. In the history of the WARR Hyperloop team, there were in total 18 different nationalities among all participants.

There are not only social diversities, but also differences in experience. The study advancement of the students differ from the first semester of the bachelor's degree to almost the end of the PhD study. The high gaps in experience between collaboratively working students might seem antipathetic, but it actually can be utilized to benefit the learning process of the students, which will be discussed in the Section III.

Furthermore, it is often the case that each study discipline has its own work culture or style, which may be very incompatible with each other. While the iterative software engineering approach does handle changes well with repetitive prototyping, that is not the case with the incremental hardware design approach. Hardware cannot be physically adjusted as easily as software code can be changed fundamentally. This is proven by the fact that projects outside of the computer science department very hardly adhere to agile development techniques.

An example of projects where hardware design and software development are deeply combined can be found in real time systems where dozens of sensors are needed to autonomously monitor, control, and react to the environment (e.g. the prototype of the WARR Hyperloop II team. When time is limited by deadlines, the system has to be presented, verified and assessed [6] extremely early, which is especially difficult when software technologies have to be compatible with the components on the printed circuit boards and sensors and actuators have to be integrated into the structure.

III. Realization

We mitigated the lack of contact between various faculties by forming sub-teams, which are composed of software engineers as well as hardware designers. The flawless integration of both software and hardware components in the end product is only possible when direct communication between software developers and hardware designers is established, easiest archived by putting them in a small group of people who are in constant interaction, analogously to communication flows found in Scrum projects. A good example would be the electronics sub-team of the WARR Hyperloop III Team, which not only included electrical engineers who worked on the printed circuit board designs, and mechanical engineers, who worked in physical structure integration, but also of computer scientists, who developed the system's architecture. This was only possible due to the big variety of studies in the team. Figure 2 reflects this.

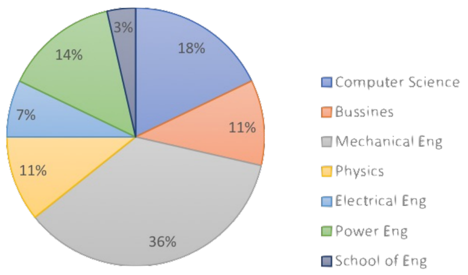

Fig. 2. Study Discipline Distribution

This partitioning of small sub-teams including only 3-4 people is also used in agile projects [7], promising an overview on the project progression despite heavy and frequent changes in requirements, as well as changes in the team composition. In order to sensibly subdivide the whole team into sub-teams tackling only one or two specific problems, the system has to be analyzed and decomposed into smaller subsystems [8]. With this technique an at first seemingly impossible problem gets decomposed into a collection of far more easily manageable sub-problems.

Problem decomposition and team subdivision reduces the negative reaction of requirements adjustments by resulting in a minor interaction in terms of communication and coordination and supporting adaptability to external changes [7]. Nonetheless, fundamental system modularity [9] is essential in dynamic projects. In the requirement elicitation phase of the agile projects life cycle the grade of cohesion and coupling of the subsystems [10] should be defined to find a golden mean between cost, quality of production and amount of time needed. In theory, quality of a module increases with increasing cohesion and coupling decrease, but more requirements regarding modularity also mean that more time is needed for developing and testing.

A good example for the importance of interchangeable and/or excludable subsystems is the first iteration of the WARR Hyperloop pod prototype. The so called Pusher Vehicle built by SpaceX promised to accelerate the student's prototype to speeds up to $100m/s$, which would allow for magnetic levitation using electromagnetic suspension [11] over the subtrack consisting of aluminum plates mounted onto a concrete fill bed. The problem is that when the relative movement between magnets and conductor is too low, a drag force against driving direction occurs by the shift of the magnetic field. Hours before the final competition ride, SpaceX informed the students that their Pusher Vehicle would not be able to accelerate the prototype to the promised speed, and the students then calculated that at the lower speed, the drag force created by the electrical currents in the conductor would be too big. In foresight, the students team modified their modular levitation system, even at a very late point of time, hence preventing repulsive forces against the direction of movement and, as a consequence of making the subsystems interchangeable, won the first SpaceX Hyperloop Pod Competition in January 2017.

Nevertheless agile methodologies used in software engineering are not always applicable in hardware design and manufacturing due to limitations in hardware adjustments. Therefore a forethought technical architecture and real time management of risks are of crucial importance. As stated in [12], risk management should be integrated into the project's development life cycle. This includes starting off each sprint or iteration with a critical review of each previous decision's consequences, as well as what are the next possibilities, and how these can affect the system as a whole. Additionally, it is necessary to communicate well with all involved stakeholders to minimize likelihood of risks as soon as possible.

That said, the terms cohesion and coupling in [10] are used to only depict software modules. In our case, we consider both hardware and software components of the vehicle to be independent modules, that should affect each other as little as possible. It is essential that when one student's components gets changed, it doesn't completely destroy or eliminate the work done by another team, as this would deeply affect other participant's motivation. Creating modular systems goes a long way into solving this problem, but can only do so to a certain extent, as hardware and software components always need some type of integration.

IV. Lessons Learned

Combining the incremental hardware design approach and the iterative agile software development life cycle adds to the student's understanding of project management. These principles are often taught to the students only based on theory, and when a student then receives a good grade in the final of the lecture, it is said that he/she has understood the subject material and learned something useful. Although this approach certainly has its advantages when referred to software management techniques and methodologies, ”learning” in the sense of the acquisition of knowledge or skills does not simply consist of immediate concrete experience as the basis of observation and reflection [13]. To completely ”learn” and understand project management techniques, reflective observation, abstract conceptualization and especially active experimentation is required.

Especially in an interdisciplinary context, different people behave differently when it comes to meeting deadlines, working in a team and showing initiative in group problem solving. For example, while one student might only attend the weekly team meetings and finishes his tasks solely at home, another one needs the physical presence to be up to date with the changes happening to the project. This self identification only happens when truly experiencing the experiments inside an interdisciplinary team [14], as the students only absorb generalized content by the lectures and it's exercise. The same principle can be applied to intuitive-experimental thinking: There are different types of thinking styles, for some teacher-centered teaching is effective, for the others not so much [15]. Only when a person with experimental-intuitive thinking style jumps in at the deep end, and has not yet mastered the extent of their study discipline, he encounters problems unforeseen and trains his decision making and data interpretation abilities.

As stated in Section II, experienced and unexperienced students were put together in small sub-teams, which were composed of both hardware and software components. An important sub-team, for example, was the propulsion mechanism, composed by both mechanical engineers and electronics. The business team, on the other hand, was composed by business students responsible for sponsoring, and web-developers responsible for the website.

By creating a practical learning environment similar to a classroom, not only the students with little to no project experience evolve individually, but also the lead of the sub-team in the role of a teacher [16]. The less experienced students are provided with information about the requirements, suitability and viability of possible technologies, expert knowledge as well as lessons learned from prior mistakes. With the guiding help of the experts the less experienced students can save a lot of time when trying to meet the expected requirements. Furthermore, the team members in the teaching role have the chance to evolve by taking a leading position and therefore sharpen soft-skill competencies such as time and budget management competencies, as well as how to explain technical aspects to an interdisciplinary audience. They also get a better sense of self-efficacy [17] by observing their competence in various disciplines while trying to convey ideas to others.

V. Conclusion

In conclusion, in order for a multidisciplinary team to function together, it is essential that each team member respects the roles and issues of the others. By decomposing big systems into smaller sub-problems, we were able to have each sub-team work on its own component, and always have continuous feedback about the complete system. We have learned, that the right composition of the team is necessary for it to function properly. The ideal sub-team configuration contains not only students from different faculties and backgrounds, but also with different levels of experience, which can help balance the learning experience for everyone involved.

In addition, techniques used in agile methodologies provide a good initial proposal of how to have these different teams collaborate with each other, as well as the structure of the emerged sub-systems in the system. Furthermore, it is of importance to define the grades of coupling and cohesion to guarantee a modular enough system design, while keeping the developing time low. Together with prospective risk management, the achievement of the project goals is thus best secured. By encouraging teams to work together and giving them goals that are dependent on complete collaboration, it is possible to ensure that every participant is able to experience new disciplines in a motivating manner.

References

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

[2] 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.

[3] M. E. Conway, “How do committees invent,” *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.

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

[5] S. Krusche and L. Alperowitz, “Introduction of continuous delivery in multi-customer project courses,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 335–343, ACM, 2014.

[6] C. U. Smith, G. A. Frank, and J. Cuardrado, “An architecture design and assessment system for software/hardware codesign,” in *Design Automation, 1985. 22nd Conference on*, pp. 417–424, IEEE, 1985.

[7] S. Augustine, B. Payne, F. Sencindiver, and S. Woodcock, “Agile project management: steering from the edges,” *Communications of the ACM*, vol. 48, no. 12, pp. 85–89, 2005.

[8] L. Rapanotti, J. G. Hall, M. Jackson, and B. Nuseibeh, “Architecture-driven problem decomposition,” in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pp. 80–89, IEEE, 2004.

[9] C. Baldwin and K. Clark, “Modularity in the design of complex engineering systems,” *Complex engineered systems*, pp. 175–205, 2006.

[10] H. Dhama, “Quantitative models of cohesion and coupling in software,” *Journal of Systems and Software*, vol. 29, no. 1, pp. 65–74, 1995.

[11] N. Grebennikov and A. Kireev, “Electromagnetic suspension used for high-speed vacuum transport,” *International Journal of Applied Engineering Research*, vol. 12, no. 12, pp. 3293–3297, 2017.

[12] A. Jaafari, “Management of risks, uncertainties and opportunities on projects: time for a fundamental shift,” *International journal of project management*, vol. 19, no. 2, pp. 89–101, 2001.

[13] D. A. Kolb, “Management and the learning process,” *California management review*, vol. 18, no. 3, pp. 21–31, 1976.

[14] L. Jaccheri and G. Sindre, “Software engineering students meet interdisciplinary project work and art,” in *Information Visualization, 2007. IV'07. 11th International Conference*, pp. 925–934, IEEE, 2007.

[15] S. Epstein, R. Pacini, V. Denes-Raj, and H. Heier, “Individual differences in intuitive–experiential and analytical–rational thinking styles.,” *Journal of personality and social psychology*, vol. 71, no. 2, p. 390, 1996.

[16] J. Harland and K. Kinder, “Teachers' continuing professional development: framing a model of outcomes,” *British Journal of In-service Education*, vol. 23, no. 1, pp. 71–84, 1997.

[17] A. Bandura, “Perceived self-efficacy in cognitive development and functioning,” *Educational psychologist*, vol. 28, no. 2, pp. 117–148, 1993.