REAL-WORLD TEAM PROJECTS AS PART OF THE INTRODUCTION TO SOFTWARE ENGINEERING

P. Petrovič

Comenius University Bratislava (SLOVAKIA)

Abstract

For more than 10 years, we have been running a course Introduction to software engineering for 3rd year students of Applied Informatics bachelor study program. While many higher education programs tend to integrate some kind of project as part of the practical work in this kind of subject, usually the project is an artificial model situation unrelated to any real-world problem and real customers. Our experience from such simulated projects is that the discussion of the group of student developers solving the project about the requirements specification and/or about the software design becomes ambiguous, superficial, futile, not anchored in the reality, without any true feedback loop, and therefore even misleading, frustrating, and potentially leading to meaningless arguments. Another motivation for using real-world projects comes from our experience in teaching other programming courses - such as introduction to Java programming language. Students in that course solve tens of little tasks during one semester. Each task is focused on a particular concept, improving their specific programming skills. Nothing else, however, could give them a more intense learning experience than taking the time to implement their final, more complex project incorporating many features they learned throughout the semester in a mutual concert, so we have incorporated this kind of project at the very end of the course. Yet, those projects are still in the category of a "larger homework" and they can be solved by an individual student investing one or several days of focused work. In the software engineering course, that we present in this article, the projects are for the duration of the whole semester, for groups of four students, supervised by a teacher in weekly meetings. We will explain how we prepare the topics for the projects, how do students select them, what is the overall development process, deliverables, and organization of work to share our experiences in a wider community. We describe the types of projects and clients, and various pitfalls that we ought to carefully avoid.

Keywords: Software engineering, team project, university-industry collaboration.

1 INTRODUCTION AND RELATED WORK

Open a website of a random academic institution, such as a particular department at some university. Without much effort, you will find the list of members of that department, and for each of them, you will find their achievements – publications, grants, and projects, patents, and other important significant contributions of a particular scholar. Do the same with a random industrial company. If you will find a name of a single employee working in that company, you must have just won a jackpot. In the academic world, people are expected to be strong individualists, everyone working hard in order to put yet another star on their rank insignia decorating their shoulder. On the other hand, in practice, success is seldom achieved only through an efficient and well-organized teamwork. Most of the time children and later students spend in the academic education, they work to collect the little stars, and while it is important that each and every one of them passes some agreed minimum requirements for a well-rounded personality, we still spend too little time on preparing them for the cooperation, collaboration, communication, negotiation, self- and peer-evaluation, responsibility for the common result, and sharing the efforts for the benefit of the group, and community. Schools and education systems as such in some countries put more, while in others less emphasis on team and group activities.

Already during the early years of the millennium, I have been partially involved in a team project class at NTNU in Trondheim [1]. In this course, heterogenous groups of students from different faculties formed teams and worked on a particular concept during an intense period of several weeks. This period was reserved only for the course, leaving space for the ability to focus all efforts on the project. Even though the approach was a somewhat optimistic endeavor, the trend was clear, and benefits for the relevant student experiences significant. Later, I have been deeply involved in organizing FIRST LEGO League competition in the years 2008-2017 in Slovakia [2-4], localizing the content to our language, running training sessions for coaches, referees, and judges. One of the four graded categories in the Central-European version of FLL as adjusted by Hands on Technology in Leipzig was the teamwork. Not only is the whole preparation for the competition a team effort, where teams of 3-10 students

cooperate on building and programming a robot and preparing a team research project, in addition to that, they were contested at the tournament site with how well they can cope with a novel task given by the referees that ought to be solved only thanks to an efficient cooperation and communication. Only those teams that applied proper methods and correct team spirit of teamwork during the several months of preparation to the tournament could excel in that category. After seeing such diverse examples of the need and effects of the teamwork, my personal attitudes towards the importance of careful implementation of more teamwork in the educational process at all levels strengthened. And thus when my Department of Applied Informatics at Comenius University was preparing a new version of Applied Informatics Study program and a new course on the development of information systems (a kind of gentle introduction to software engineering) was about to be established sometimes around 2010, my requirement for accepting my active involvement in this course was that the course will also become the team project course of our study program. Since that time I have been teaching in this course, and guiding the groups of students through their team projects. In some of the years, I get help from the doctoral students associated with our Department, but in the last couple of years, I have been doing all the work. Let me explain all the interesting pieces and how they fit together in the forthcoming sections.

2 THEORETICAL COURSE CONTENT

Before focusing on the student project, let us explain its context in this course. Students typically take this course in their 5th semester. They had already passed courses Programming I till Programming IV – the first two using Python as their first programming language, followed by C++ and Java. They also had some hands-on experience with the assembly language, web technologies, such as HTML, CSS, PHP, JavaScript, they passed an introductory Database, and Algorithms and Data Structure courses, and of course several math and logic courses. Some of them experienced other programming paradigms, such as functional programming in elective courses. In all the programming courses, they individually worked on smaller-scale projects to convert the acquired knowledge into active knowledge. All in all, when starting the 5th semester, they are prepared for being challenged with a larger-scale software project. However, apart from some gentle small-scale experience with the object-oriented design, and useful hands-on experience with clean-code code reviews, they did not have any courses related to software engineering, and that is the gap the course is attempting to fill. The idea is to feed the theory about software engineering – at about the same time as it will be needed in the practical project, in a synchronized manner with the project development. The following topics are covered: 1) requirements specification, 2) traditional software development models, 3) principles of software design, 4) software architecture and architectural styles, 5) fundamentals of the UML, 6) design patterns, 7) integration of applications, 8) agile methods of software development, 9) clean code, 10) soft skills. We usually also invite experts from software companies to give talks about selected topics, such as cloud, microservices, and others. In the two recent years, the course got extra goals - to prepare students for software business, and for this purpose we invite leaders of successful software companies to talk about their success stories, and lessons learned – for instance in this season we invited leaders from SAP Labs, Photoneo, and UI42 – all local companies with interesting and motivating business stories. Unfortunately, we did not find a single suitable textbook, which would cover our selected topics. We therefore have several books as the recommended literature, and I spent considerable time to write/collect a comprehensive study material equivalent of about 115 page textbook containing the information I expect the students to learn. The first part of each lecture is a written test that summarizes the material from the last week's lecture. Earlier, we used a more traditional model – of single exam at the end of the semester, but I the new model leads to better learning outcome, students need to work with the material regularly for a longer period of time, and they need to be motivated. Otherwise, too many of them would not resist the tendency to postpone learning of a too large a portion of the course material until the examination period with more shallow traces in their minds. UML diagrams are the topic of midterm written test taking about 2 hours, where the students are given a few paragraphs description of an information system they are going to design – and draw the 5 basic diagrams, such as the use-case diagram, state chart for a selected entity, a sequence diagram for a selected use-case(s) – with the focus on design, not specification, a class diagram for some non-trivial part of the design, and the deployment diagram.

3 EXAMPLE PROJECTS

To give the reader a better idea what kind of projects we are going to be talking about, let us first show examples of three different student projects developed in the recent years.

3.1 HP Controller - HP4191a machine manipulation and data visualization cross-platform desktop app (2020/21)

The Department of experimental physics at our Faculty owns a couple of valuable instruments produced back in the last millennium, and they are still in a good shape. They even have a special interface for connection to a PC, but the available software did not satisfy the requirements on control and data acquisition of the researchers' experimental setup. They have thus worked with two student groups to specify their software requirements. Student teams have developed a useful software that collects and visualized data, and controls the various features of the instruments over the standard GPIB interface. As this happened during the pandemic lock-downs, they performed all the work remotely from their homes, while I was operating the instruments manually in the lab based on their instructions when performing tests and debugging. One group developed a Python application for a different type of instrument, while this group has produced a Java desktop application, as shown in the figure 1. See the project repository and their technical developmental documentation for more details. [5]

Figure 1. Screenshots of the control software with the actual instrument operation shown in the insets.

3.2 Warehouse (2023/24)

A multinational logistics corporation CEVA (formerly GEFCO) is running many operations withing Slovakia in various territories. Transporting parts, completed cars, running logistics warehouses, renting their warehouse space to other companies, and much more. In one particular smaller-scale subsidiary, they identified a need for a new software for maintaining a variety of their warehouse operations. One team from this year's issue of the course have just completed the required software that can handle contents of all the warehouse storage location, manages inputs, outputs, invoices, inventories, materials, clients, produces statistics, and visualizes the warehouse at a glance. [6]

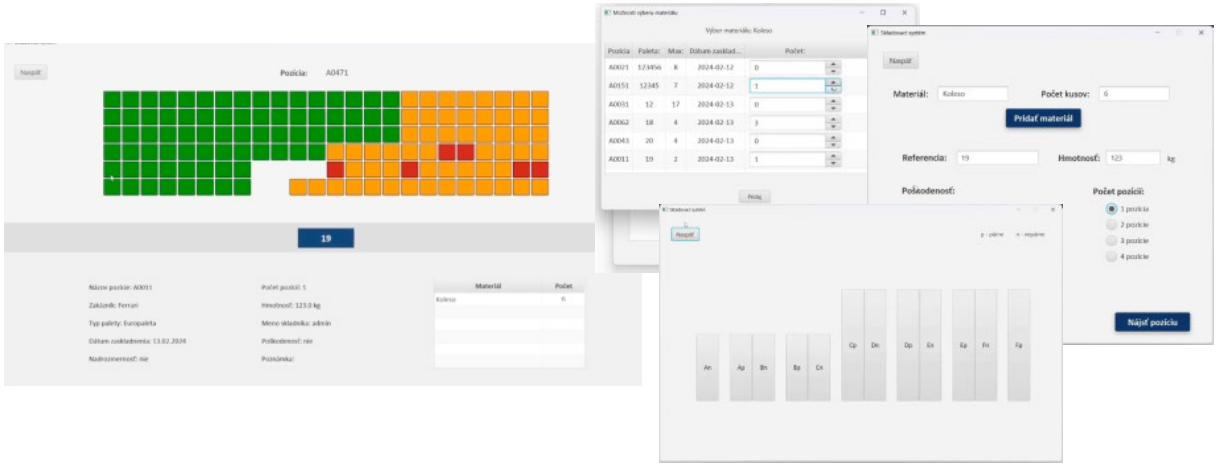

Figure 2. Screenshots of the warehouse management software.

3.3 Hot-air – Industrial fan control (2022/23)

BOGE Elastmetall Slovakia is a successful factory for producing parts combining metal, plastic, and rubber materials with its own development/design department as well as produced parts testing department. To support testing of the designed parts, the parts encounter periods of exposure to various temperatures, which was until now done by a personal operator manually tuning several hot-air fans. Students have developed a system consisting of a hardware device that connects directly to the hot-air fan and a local ethernet network, and a software server, and gui controller that allow configuration and automatic remote control of the fans during a several day lasting testing sessions. The software was implemented in Java, C, and C++. [7]

Figure 3. Software (and hardware) for controlling industrial hot-air burner shown on bottom the left.

4 PROJECT INITIATION PHASE

Our autumn semester typically starts in the second half of September, and my responsibility is to find the clients with projects typically during the summer season, before the semester starts. In general, we are trying to avoid restricting the development platform, or programming language. Students should develop a technology-neutral requirements specification, and then be free to select (agree with their clients on) a most suitable development platform – based on all the constraints identified. However, since our students have mostly been exposed to programming simple desktop applications, we try to negotiate with the clients a condition that they agree that the system can be developed as a desktop application. However, due to the obvious advantages, the trend is to develop web applications in most of the situations today. We are not able to require the students to implement a full-fledged web application, but many of them already have that experience and they are willing to share it with their team mates, while others are willing to learn from them to gain experience and part of those skills too.

4.1 Finding clients of the projects and negotiating

Our policy is that if both the client and the students agree that the system will be a web application, we are not going to stand in the way. We contact various potential clients – mostly in our Faculty – teachers and researchers who could be in a need of a relatively small software project that they could use in their research or teaching. However, since our colleagues are involved in various free-time, community or voluntary activities, the variety of project ideas goes much more beyond. Several years ago, one of the former doctoral students, started to work in a development department of a factory, but since he remembered our internal Faculty-scope calls for projects, he suggested to propose an idea that would be useful in some of their internal operations. We have formalized the cooperation in an official agreement between the University and the company [8]. Rumors spread and another large company joined in a similar manner. In the recent years, our students have thus been producing little open-source free applications that could be useful in various sections of these industrial companies. As a bonus, they can see their own student work to be useful for practical applications, they cooperate, negotiate and communicate with the industrial staff during the development, and in this way gain valuable experiences, not only in the teamwork, collaboration, and cooperation, but also bits of a real-world industrial experience.

4.2 Forming groups of students

In order not to complicate the process, originally, students were allowed to form groups as they preferred with a little bit of our support if they struggled finding their own group. However, this has not proved to be a successful strategy, because more advanced and productive students naturally formed highly efficient groups, whereas the very opposite type of students filled other groups. Naturally, the results of the groups were too uneven, and some projects turned to be too challenging to complete consequently.

In the last six years, we have turned to a randomly generated groups strategy. It helped to solve the issue of uneven group performance considerably, but it has also helped the students in practicing their communication and collaboration skills. During their studies, most students tend to form relatively closed small groups of study peers and friends and find it difficult to communicate with others. However, in the real-world, employed workers are often forced to cooperate with strangers, and we feel it is our responsibility to prepare them to cope with such situations efficiently. Overcoming their comfort zone during their studies to learn about more peers of their age might turn to be an asset in their future professional career. On the other hand, we were surprised, how much resistance this transition caused. Students were already forming groups even before the semester started, to make sure they will be in a "nice group", and crossing their plans made them even to contact several different officers of the Department, study program, and the Faculty and (of course, unsuccessfully, but it was tight) demand that we revert to the original strategy. Their rants in the anonymous (and public at the same time – a very toxic combination!) feedback system ran by our Faculty resonated on this topic for several forthcoming years. We collect their feedback in various other ways too, and even this year, we have received a suggestion that "ok, if the groups are random, we would like to get to know each other a few weeks before the project starts so that we can get to know each other ahead of the project". Well, the times have certainly changed since I was a student, sometimes in a questionable direction.

4.3 Selecting project topics by the teams

Before the semester starts, we have already had negotiated with the clients of projects, discussed their possible requirements in considerable detail to assess which projects are suitable, and which ought to be filtered, or perhaps redirected to different purposes – such as topics for bachelor and master theses, or specialized courses. At the end of this process, we finally select exactly the same number of projects as is the expected number of student teams – perhaps with some flexibility in the projects that we have more familiar connection to the clients. Therefore, practically every topic gets selected, and thus we can guarantee the clients their topic will be implemented so that they can start the required preparation processes without risking it being a wasted effort. However, we would like to give the students as much freedom to choose their preferred topic as possible, and we achieve that in the following way: A list of all available projects with a few paragraphs of description is available to the students in the first week of the semester. They discuss the projects in the groups and assign priorities 1-10 to each project depending on how well they think they could complete the project regarding their technical skills, how motivating it is the topic for their group. We encourage the students to possibly negotiate in between the groups regarding their priorities if they like, this helps in balancing their interest among the projects. After all groups submit their preference scores to the individual projects, we run either an automatic [9] or a manual optimization to maximize the total satisfaction score. In the current season, with one or two exceptions, all teams received a project they prioritized with at least 7 out of 10 points, most of them with score 9 or 10.

4.4 Initial student interviews with the clients of the projects

Immediately after the projects get selected, students contact their clients of projects to schedule an interview. Students are asked to use a sound recorder at the interviews and produce detailed notes to send back to their clients for confirmation and/or filling additional details that might have been omitted in the meetings. They are required to dig as much information as possible in a structured but open-ended interview and the recording and notes serve as the basic solid ground for further requirements elicitation, analysis and negotiation.

5 SOFTWARE DEVELOPMENT MODEL AND DELIVERABLES

When discussing the current methods of software development with developers of various software houses, we are usually (but not always) receiving an answer that they use an agile method of software development, typically some version of SCRUM adjusted to their specific needs. At the same time, those methods expect a larger team of full-time developers working at least 3x5 full days, or more. The total amount of work in one SCRUM sprint is thus comparable to the overall work invested by the students in our projects. Splitting the work to further individual smaller-scale sprints would not make much sense. Instead, the evaluation of the completed work is performed naturally after each new feature is implemented and integrated into the latest version.

One of the most serious challenges in software development is that the software developers are not experts in (do not understand) the world of the problem domain, and that other stakeholders are not able to describe their needs to software developers. Often, they can't even imagine what the developers

are capable of building, and they cannot grasp the constraints and limitations, contours of which only become apparent later in the development process. One of the main goals of our course is to let the students understand this situation based on their own experience, and cope with it.

5.1 Requirements specification

Based on the first interview, and any number of required follow-up consultations, the task for the students is to write the Requirements Specification document according to the standard IEEE/ANSI 830-1998 [10]. Applied Informatics students are usually more technology-oriented personalities. This part is an important training in formulating sentences in concise language, without ambiguities, conflicts, and having the list of requirements complete. They are not used to atomic separation of ideas as required in the detailed specifications part of the document. Drafts of all students fail to reach the criteria, and we iterate until the documents they produce satisfy them. Only after reaching that stage, they start to iterate with their clients about the content to reach a common understanding about the planed implementation project. A formal approval from the clients is required, but our model of IS development is not waterfall. Students are allowed to start writing code from the day 1, if they are sure, the code is useful for the product, they can be analyzing, designing, testing, all that before the final draft of the Requirements is approved. However, we demand that they dedicate enough efforts to reach as complete Requirements Specification as early as possible, while adhering to the expectations. We could compare this process to preparation for a spring planning meeting in SCRUM, which also fixes the list of planned tasks and features to be implemented in the upcoming spring.

5.2 Design

The scale of the project students implement could be classified as small-size. It is therefore not inevitable to produce a very detailed high-quality design. However, all projects must complete several steps.

5.2.1 User interface – mock-ups

First, design a user interface (GUI or console/command line). Students draw wireframes for all the dialogs, windows, pages, etc. the application is going to display. Also, the group of student developers must meet with the client and discuss in very detail every single control in the UI, its behavior and consequences. This is a wonderful opportunity to disclose any hidden ambiguities and misunderstanding that may have still been forgotten in the already approved specification.

5.2.2 Use-case analysis

Preparing a use-case diagram is not obligatory, but some groups perform this step. In each case, they should analyze the software requirements, and get ready for the thinking from the point of view of the developer.

5.2.3 Data and function decomposition and detailed specification

Formats of all persistently stored data must be clearly defined in detail – this includes entity-relationship diagram of database, if any, formats of any data or config files, and data communication protocols between internal or with external entities. Students perform data and functionality decomposition and depict their findings in a UML component diagram, and in short descriptions of the individual modules, their responsibilities, and interfaces in between them should also be provided. Students either select some important use-case, or an important entity, and model them as UML sequence, or UML state diagrams. A detailed UML class diagram evolves with the implementation.

5.2.4 Testing scenarios

To avoid unintentional omitting of implementation of some features, or postponed fixing of issues, students in the design stage also need to prepare testing scenarios, which are a kind of acceptance tests – sequences of actions to be performed to verify that all requirements specified in the Requirements Specification document are in place.

5.2.5 Implementation plan

For many students, this project is their first experience of working together in a common code base. Therefore, they devise a plan of implementation steps, analyze their dependencies, and agree on their interfaces so that they can always keep track of the state and progress of the implementation. Part of this

is assignment of individual increments to individual developers or developer groups so that they are aware who is responsible for each part and should be aware of the correctness of the individual interfaces.

5.3 Collaborative infrastructure

Managing this kind of projects can only be efficient if a suitable supporting infrastructure is in place. Otherwise, each student team would select their own way to organize, and it would be impossible to keep the necessary overview of all the ongoing projects, making sure that all students are fully involved, and be able to remove any impediments appearing.

5.3.1 Software repositories

All student projects are therefore maintained in public Github repositories (unless exceptionally required by the client otherwise), which are created in a common organization. Each repository has an associated Github project and Github team consisting of four students with maintenance access and a teacher with full administrator access. The main branch has a branch protection rule so that all changes need to be introduced in separate branches and merged to the main branch through pull requests that require reviews.

5.3.2 Kanban agile management method

Github projects provide the project boards with issues arranged into columns in Kanban style organization. Students are free to define the columned issue groups they wish, but they are required to have at least three groups: new, in progress, done, but usually they follow the prescribed template, which defines new, backlog, ready, in progress, in review, and done with some default automation. All communication of the developers about the project is suggested to take place in the discussion threads of individual issues, and the following rule is enforced: no developer can perform activity with the project, unless there is an issue for that activity created in the project first. This implies that all the developers are informed about all the activity ongoing. It is required that all four members of the team divide both the implementation and documentation work evenly, and everyone participates on both. The four members of the team agree on fixed roles in the beginning of the term, documentation, implementation, communication, work organization. Developers are encouraged to organize their work organically and democratically based on complete mutual consensus after discussions, and these roles are only meant as a backup for the situations when a free organization fails, and an assertive decision must be made. Each role thus brings the responsibility for the segment of work, but also the power and authority to make decisions in situations when it is necessary.

5.3.3 Meetings

Teams are required to schedule weekly meetings, where all developers must be present. In these meetings, they evaluate the work performed on the tasks since the last meeting, discuss all the open questions, make decisions, and plan and divide the work for the time period until the next meeting. At the meeting, one of the members takes notes and produces minutes in a standardized format that are uploaded on their Github repository wiki – so that all team members and the teacher can use them for reference. In addition, all teams have a regular weekly team meetings with a teacher, where all team members are required to participate. They receive feedback on the documents they have delivered, discuss their ongoing work, and issues, present their progress, and communication with their clients of projects, and the plan. When a delivered output does not meet the expectations, the team has to work further and submit a new version of the document – as many times as is needed for it reaching the expected standard and being approved.

5.3.4 Hybrid and Online Projects During Pandemic

Due to the extensive lockdowns in the 2020/2021 academic year, students were required to work from home. University provided the MS Teams communication platform. All the meetings with clients, teams, and the teacher took place on-line. Students did not meet each other, their clients, or the teacher in person through the whole semester. Despite this handicap, all projects in this season worked very well, if not better than usual. In some cases, it required extra efforts from the teacher in supporting the projects that used lab equipment that the students accessed remotely.

5.3.5 Obligatory outputs

Every team's final output is the working code on the Github. A working software must be demonstrated to the client, installed and integrated in their infrastructure. Clients have some time to test the software, and in case of any discrepancies with the approved Requirements Specification document or

bugs/issues discovered, teams must implement the required changes. The development documents – Requirement Specification, Design, and Testing Scenarios are typically merged into a single PDF file as developers' technical documentation. An installation guide must be published either in Github's README or as part of the technical documentation. The client must send a formal approval of acceptance of the resulting project for the students to get credits for the course.

5.3.6 Optional outputs

When agreed or required, students can prepare a more detailed user guide, generate javadoc-style technical API documentation, provide internationalization support so that the clients can easily add further language versions for the user interfaces without the need to modify the source-code. In special exceptions where hardware components are developed, teams in cooperation with the Faculty can produce several instances of the components for larger scale deployment.

6 OVERVIEW OF THE DEVELOPED PROJECTS

The course has been running since the autumn 2010, i.e., 14 consecutive years. The number of teams in each year oscillates around 15, resulting in about 200 projects developed till today. Most of them have been successfully deployed by the client, put in use, and many of them are still in use today, or have been replaced by newer versions. The failure rate is typically kept at 0-3 projects. In several cases a project that was started had to be stopped after only a few weeks of development due to some unexpected circumstances by the client – such as availability of certain data, API, or political reasons in the organization. In these cases, students started to work on different projects with slightly relaxed requirements, but typically completing them in excellence. Failures are usually caused by teams loosing one or two members (students fail in some other course and decide to completely leave the study program), unexpected circumstances by the client discerned too late in the process, and limited flexibility of some handicapped students in very few instances. Otherwise, students and clients were always very motivated to complete the project and reach a working stable version.

For reference, we list here a classified list of projects from the last 2 years, project titles have been translated to English for more clearer illustration.

2023*: Projects with a client provided internally by the course:*

- RoboCup - registration online system for robot competition
- Robot League - new features for the site of on-line Robotics Contest
- Backups - automated/incremental backup of multiple internet sites
- *Projects with a client from within our Department or Faculty:*
- Personal Account - detailed overview of individual expenses of shared account at our Department
- Business trips - support for business trip administration in our Department
- Night Sky - extracting/searching metadata from astronomical telescope observations
- Astronomy - remote automated and scheduled control of shutters of telescope
- Bottle Registry - bookkeeping and situation with gas storage tanks in experimental physics lab
- *Projects with a client from our external colleagues:*
- Sport Race - registration system for orienteering sport club

Projects with a client from industrial companies:

- Warehouse - complete management of goods stored in logistics warehouse
- Trains - support for loading cars on trains in logistics company
- Visits - support for safety instruction overview and visitors registration in company reception
- Groups - support for loading cars on trucks in logistics company
- Component Supply - ordering system for components needed by a factory for production of parts
- Work Instructions - a kiosk in the workplace with browsable working instructions for individual tasks

- Text Analysis - analysis of documents for research in large networks
- Delivery Control - feedback for the dispatchers on delayed goods delivery

***2022**: Projects with a client from within our Department or Faculty:*

- List Formulas - update of our LMS to allow calculation of student score from various types of tasks sets
- Personal Account - detailed overview of individual expenses of shared account at our Department
- Spectra - stepper-motor controlled spectrometer with millivoltmeter
- Spectroscope Microscope - Spectral mapping of microscopic objects
- *Projects with a client from industrial companies:*
- Hot-Air - remote control with scheduling of hot-air fans in industrial testing lab
- Optimal Dispatcher - Application that collects data from delivery dispatchers and optimizes resources in logistics company
- Medical Examinations - Maintains information about workers medical examinations and notifies about upcoming examinations
- Form Versions - Registry of product forms with 3D software Catia interaction
- Lessons - registry of FEM calculations performed in product development
- Multi-excel Search - Intelligent searching in multiple excel files used in product design management using intelligent criteria
- Transport Audit - Verification of transport packages based on barcodes
- Weight Inspection - Verification of packages based on automated weighing and barcodes

The repositories of the projects can be found at github.com/orgs/TIS20XY-FMFI for recent years, or github.com/TIS20XY for 2017 and earlier. Before 2014, we used Google Code, and dotproject.

7 FEEDBACK AND CHALLENGES

We collect feedback at various levels: at any time, students can submit an anonymous comment or question using a form on the course website: they receive a unique link at which the teacher may publish an answer in short time. The course has a MS Teams group open, where any questions and issues may be raised. At the end of the course, we have a reflection retrospective session, when we discuss with the students the lessons learned – topics including the technical work, teamwork, and course organization, and finally using the official feedback polls organized by the Faculty as required by the legislation.

The immediate feedback has been useful to be important to resolve some misunderstandings early before they develop to larger-scale problems. In the retrospective session, students usually provide positive informal feedback, which is run at the time they have successfully delivered the project. The most frequent claims are: useful teamwork experience; interesting to be involved in real projects; cooperating with real business or client; experience of feature negotiation with clients. On a negative side, students claim there is too much work in the course given the credits earned (6-7), and the negotiation and requirements specification phase taking too long time, see figure 4, even though only about 30% of the students responded (typically there are 45-65 students taking the course).

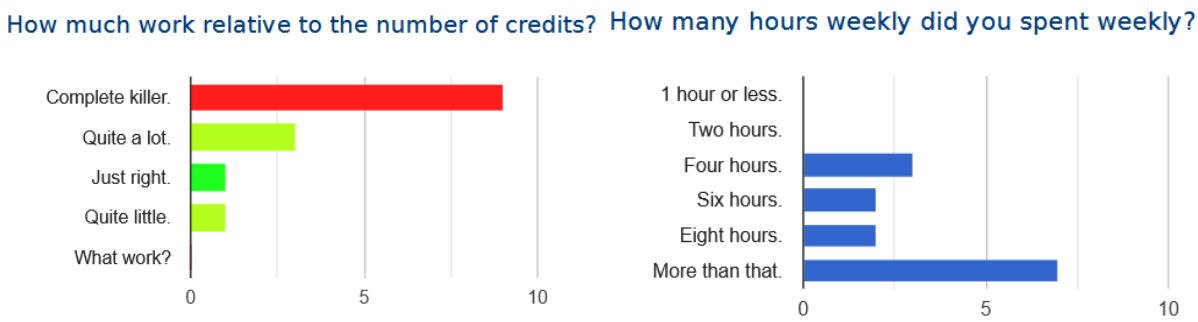

Figure 4. Student response in official feedback system for 2022 season.

So the most important challenge is the limited time, however we are pressed by the placement of the course in the study program: it cannot be moved/extended to the spring semester, because that is the time students are working on completing their bachelor theses, and the spring semester of their second year is also too crowded, so we have to live with the space we have. Due to this limitation, we cannot make the design phase detailed enough and thus some teams report difficulties in integration of their individual contributions to the whole, while others would rather drop the design completely, and run in an ad-hoc more agile manner as they progress on implementation. A huge challenge is that many teams undergo a learning experience, since they choose to use a platform/framework they have never used before – such as Django, Laravel, Symfony, and similar, or one of the students has experience with the framework and others learn from him or her. In these circumstances, it is difficult for the students to produce a good design, since there are still features of the framework they are discovering during the development process. All teams however reported that they prefer the freedom to choose the platform/framework on their own over having a more streamlined process with limitation to a prescribed framework. The feedback from the clients of the projects have always been positive since we select motivated clients who are supportive and welcome cooperation with students and are aware that projects may fail in some cases. In several instances we ran follow-up projects, where students did not develop their project from the scratch, but worked with legacy code of the previous group, however, we are trying to minimize this type of projects, and direct these requests to another course created just for that purpose: *Life Cycle of Information Systems* at master level study program – unfortunately that course is elective and thus only few students participate in it.

8 CONCLUSIONS

In this article we have presented the framework and collaborative infrastructure that we use in a team project in software engineering at our Department for the course in Applied Informatics study program. Team projects provide an excellent connection to industrial applications of the material covered in the course both to the students and to the teacher. The model of software development is in between the traditional approaches and agile methods. We feel it is important to give students an experience with more formal methods of software development first so that they can better judge and compare when participating in more agile software development projects later. The side effect of the student learning experience is a lot of working software in use by our partners in academia and industry [11].

REFERENCES

[1] G. Sindre, T. Stalhane, G. Brataas and R. Conradi, "The cross-course software engineering project at the NTNU: four years of experience," Proceedings 16th Conference on Software Engineering Education and Training, 2003. (CSEE&T 2003)., Madrid, Spain, 2003, pp. 251-258, doi: 10.1109/CSEE.2003.1191384.

[2] K.W. Lau, H.K. Tan, B.T. Erwin, P. Petrovič, “Creative Learning in School with LEGO Programmable Robotics Products,” *Proceedings to Frontiers in Education'99*, IEEE CS Press, pp. 12D4/26 - 12D4/31 vol.2, 1999.

[3] J. Svetlik, P. Petrovič, “Vzdelávanie v oblasti modulárnej stavby a konstrukcie objektov prostredníctvom FLL a Robot-Game,” International Scientific Herald, vol.2, number 21, part 2, pages 352 – 359, 2011.

[4] P. Petrovič, D. Onačilová, J. Svetlík, “Skúsenosti s prípravou sútaze v stavbe a programovaní robotov FIRST LEGO League z pohladu organizátora, trénera a rozhodcu,” Didinfo 2010, Banská Bystrica, April 8-9, 2010.

[5] R. Nagy, J. Žák, M. Pikuleva, J. Pastorek, “HP controller, HP4191a machine manipulation and data visualization cross-platform desktop app”, github repository, on-line: github.com/TIS2020-FMFI/hp. 09/2020 – 01/2021.

[6] S. Snidová, P. Filipiak, M. Palider, “Sklad, Storage system“, github repository, on-line: https://github.com/TIS2023-FMFI/sklad, 09/2023 – 01/2024.

[7] K. Kubinová, M. Dinka, T. Hruškovic, I. Böhman, “Hot-air, Ovládanie teplovzdušných dúchadiel”, github repository, on-line: https://github.com/TIS2022-FMFI/hot-air/, 09/2022 – 01/2023.

[8] Zmluva o spolupráci Z/2020/1445/IX/FMFI/DEK, Univerzita Komenského v Bratislave, BOGE Elastmetall Slovakia , a.s, 2020.

[9] M. Mečiar, "Optimalizácia priradenia tímových projektov riešiteľským skupinám", bachelor thesis, Faculty of Mathematics, Physics, and Informatics, Comenius University, Bratislava, 2016.

[10] IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[11] R. Kebisková, “Univerzita Komenského a GEFCO spolupracujú na testovaní a implementácii IT riešení“, transport.sk, on-line: transport.sk/logistika/gefco-spolupracuje-s-univerzitou-komenskeho-na-testovani-a-implementacii-it-rieseni-navrhnutych-studentmi/, 30.06.2021.