THIRD-YEAR INTEGRATIVE PROJECTS FOR COMPUTER AND SOFTWARE ENGINEERING STUDENTS AT POLYTECHNIQUE MONTRÉAL

Jérôme Collin and Olivier Gendreau
Department of Computer and Software Engineering
Polytechnique Montréal
{jerome.collin, olivier.gendreau}@polymtl.ca

Abstract *– This paper presents the most important aspects of the third-year project for students in computer and software engineering at Polytechnique Montréal. In computer engineering, the third-year project mainly focusses on FPGA-based embedded systems, mobile application development (Android application), network communication and protocols, and user interface design. In software engineering, the third-year project mainly focusses on software reengineering (of the second-year project), mobile application development (iPad application), network communication and protocols, and user interface design. The important role of team and project management is also underlined. Teachers' evaluation of students, as well as students' evaluation of teachers are discussed.*

Keywords: integrative project, project-based learning (PBL), mobile development, computer engineering, software engineering.

1. INTRODUCTION

In 2005, Polytechnique Montréal introduced project-based learning (PBL) in its curriculum. The main goals of this approach are concept integration, student active participation, and autonomy improvement [8]. The central idea was to offer one integrative project at the end of each year. At the beginning, effort was spent in the development of projects for the first two academic years. It was also clear very early that the capstone project (senior-year integrative project) would be defined by a non-academic client (enterprise or nonprofit organization from different industries), like in many other engineering programs around the world [9]. However, when it was time to define the third-year project, ideas were much less clear.

We observed that the third year in a curriculum is different than the others. During the first two years, students want to discover their capabilities and find out they really like. In the fourth year, students are about to finish, they have a project with a non-academic client, and have opportunities to attend optional courses they want. They think about their short-term future. Third-year students are in between: they have accumulated enough technical background, but are not going to finish soon. This is why their motivation can be lower at this point of their studies.

Coming up with a good project can be a challenge in this context. At Polytechnique Montréal, the first two computer engineering and software engineering projects were a great success. The first-year project is the development of a little robot to focus on the understanding of computer architecture and basic software development [2]. The second-year project is the development of a low-level (OpenGL) computer game focused on the understanding of more complex software design principles [5]. Our capstone project is based on the development of a project defined by a non-academic client [6].

Even before knowing what would be the subject of the third year project itself, we established some requirements for it. First, the project would need to be managed with proper techniques. It was also clear that interpersonal and team interaction skills development would continue to be an important aspect. After a few years, we also realized that leaving room to student creativity was important. Also, the third-year project was targeted as a good place to measure CEAB graduate attributes [3].

On the technical side, we knew that the system would involve computer networks and mobile application development on tablets, as it is an important new trend. Theses aspects are not part of the first two integrative projects. Third-year courses around these subjects are offered to students so it makes sense to introduce them to the projects as well. Trying to expose students to up-to-date technologies was also viewed as important. Finally, students in computer engineering and software

engineering would have distinct projects with emphasis on their respective curriculum.

This paper is structured as follows. Section 2 presents third-year integrative projects objectives. Section 3 describes the computer engineering project technical details, and section 4 describes them for the software engineering project. Section 5 explains the mobile application development concepts common to both projects. Section 6 describes team and project management. Section 7 presents how teachers evaluate third-year integrative projects students, and section 8 analyzes students' evaluation of both project courses.

2. THIRD-YEAR PROJECTS OBJECTIVES

An important curriculum decision was taken a decade ago: the first two integrative projects would be common to both computer engineering and software engineering students but the third-year and senior-year projects would be specific to each program. Therefore, each project must also reflect the differences between them.

On the one hand, the third-year computer engineering integrative project relies mostly on hardware concepts, since it is based on hardware courses exclusive to the curriculum. For instance, in the fifth semester, an advanced digital design course explores VHDL details and digital circuits. The following semester, students attend an embedded system course. Therefore, the computer engineering project uses FPGA boards to favour hardware design.

On the other hand, the third-year software engineering integrative project relies mostly on software reengineering. Relevant fifth semester courses include software tests and computer networks. The following semester, students attend a software development processes course. Therefore, the main objective of the third-year software engineering project is the second-year project reengineering, by adding two main features: online multiplayer gaming, and mobile editing functionalities.

3. COMPUTER ENGINEERING PROJECT

FPGA boards are versatile and it makes them a great educational tool in a laboratory with third-year students. Projects can include custom circuit design with VHDL or reuse of existing IP Core that usually come with the programming environment on PC. Hardcode embedded CPUs can run software, with or without operating systems. Many chips usually complete the boards for various needs: Ethernet and serial communications, digital sound input/output, USB device connections, LCD and push buttons, VGA or HDMI video output, etc. We have used different FPGA boards over the years for this third-year project. The old Digilent XUP Virtex-II Pro was followed by the Digilent Genesys Virtex-5. We now use the ZedBoard Zynq-7000 with dual ARM processors. We have explored many different ways to use them but we have found that turning the board into a TCP/IP server provides a very good and flexible development subsystem. We complement our system using Android tablets to host the client application. The communication between the two usually uses HTTP protocol with a REST interface.

In this framework, it is possible to develop projects at very low levels using VHDL or IP Cores and also at higher levels using Java code on the tablet. Almost everything in between can be integrated. Our approach offers a broad range of possibilities both for the teacher, who can diversify easily a specific project, and for students, who can pick which part of the project they prefer to spend time on, depending on their natural inclinations or skill set.

This computer engineering third-year integrative project is offered in fall (September-December) and winter (January-April) semesters. Students are always excited if we can come up with a totally new project every semester. This is hard if the whole system has to be done from scratch but easier if external pieces of code or entire libraries available on the internet are allowed in the design of the system. Over the years, mp3 codex (MP32PCM or libMAD), JSON parsers (cJSON or YAJL) and LwIP have been reused. Sometimes, these software parts are imposed to teams but most of the time, two or three choices are mentioned as available. In most cases, students find by themselves their preferred ones. This software reuse is a modern approach to software development and we encouraged it. Selection of code libraries is likely to happen again during the capstone project the following year, and in their respective careers.

Java is obviously used to program the Android device. On the Zedboard, C is used. This might seem unusual but this old language is still very popular [1]. As a matter of fact, Java and C/C++ are at the top of the list. Students usually realize at the end that they don't know C as much as they thought. Many discussions about Java/Android versus C/Zedboard and their respective development environment take place between students during the semester, which is a good outcome of exposing students to a framework based on both platforms.

In this broad choice of available technologies, teams must design a good architecture and elaborate an appropriate development plan. If they don't, problems will rise from various parts of the system: memory or interrupts conflicts, bad configuration of peripherals, software code incorrectly integrated, TCP/IP communication problems, etc. Large-scale systems also help improve debugging skills, which is an underestimated know-how.

It has always been our desire to offer complete systems integrating as many computer fundamentals and technic as

possible: computer graphics, performance issues and trade-offs, user interfaces, networking, processor and interrupt management, complex I/O subsystems, etc. This kind of environment is perfect to illustrate all the relationships between concepts. Many of our projects have been implemented following this architecture: an MP3 player ("modern jukebox"), video games (Snake, Lunar landing), an automated restaurant ordering system, a projector control system, a programmable logic controller (PLC) simulator, etc.

4. SOFTWARE ENGINEERING PROJECT

The main objective of the third-year software engineering project is the second-year project reengineering, by adding two main features: online multiplayer gaming, and mobile editing functionalities. The second-year project is both a map editor/tester and a videogame using maps created with the editor. The application runs on a single PC without any network communication. In the third-year project, students are provided with a Vision Document specifying a customer's (teacher's) vision of the bare minimum they must develop in the project. For instance, they must allow at least 2 groups of 2 to 4 players to play together online. Moreover, a user must be able to edit a map using an iPad tablet (iPad mini 4). Also, they must add chatting functionalities, pay close attention to user experience, and system performance.

From the Vision Document, teams must produce their own Software Requirements Specification (SRS) as the final product evaluation is based on their own SRS. Therefore, each team will choose what they will develop. This flexibility is valued by students as it empowers them to live with the consequences of their decisions. Of course, from an educational viewpoint, we have to make sure that students achieve a certain level of mastery, which we do by forcing them to include in their SRS mandatory requirements, as well as desirable ones.

The main architecture of the project is divided in three nodes: a "heavy client", a "light client", and a server. The "heavy client" is the enhanced, network-ready version of the second-year project. The "light client" is a migration of the C++/C# (second-year) editor to either Objective-C or Swift, depending on the students' preferences. Students must take advantage of the touch screen and the different relevant gestures. In order to allow multiplayer online gaming, students usually choose to implement a central server, yet other solutions are allowed, such as peer-to-peer networking.

Mandatory requirements of the "heavy client" and the server include: chatting basic functionalities (window integration, channels), network disconnect management, user profile, and tutorial.

Mandatory requirements for the "thin client" include: basic edition functionalities (migration of the second-year map editor), gesture usage, visual effects, sound effects, map presentation, map management, and tutorial.

On top of the SRS, third-year software engineering integrative project students must also produce several other common artifacts such as a Software Architecture Document, a Software Development Plan, a Communication Protocol, a Test Plan, and a Test Report.

5. MOBILE APPLICATION DEVELOPMENT

Tablets are used for both computer and software projects. From an educational standpoint, students must acquire knowledge and know-how about mobile programming in today's world, but there is more to it than just another device to program. We have observed that it's a way for some students to express their creativity during user interface design.

Typically, a team of five or six students will allocate two students to mobile application development. It is pretty common for students to underestimate mobile applications development, as online resources are abundant, and it seems similar to desktop development. However, mobile development offers new challenges, such as a new kind of user experience (shorter session lengths, limited presentation), extensive testing needed (device and OS diversity), and very short release cycles [4].

For the moment, mobile development is done on Android for the computer engineering project, and on iOS for the software engineering project. However, evolution of any platform might force to revisit these choices at any time. Both programming environment and tablet models change at a rapid pace and it is hard to commit to long-term decisions.

6. TEAM AND PROJECT MANAGEMENT

We let students form their own teams. Usually, groups of friends tend to have similar work habits and values. By the end of the third year, students know each other well and prefer to work with friends. We don't consider that it would be a good idea to break these natural ties.

Of course, team dynamics can always be improved. Therefore, interpersonal and team interaction skills specialists follow each team to continue developing what was introduced during the first two projects. The first intervention of the semester involves asking each team to come up with a list of common goals and/or rules for the team. It is interesting to note that this straightforward assignment often leads to long conversations. Students express strong opinions about team organization:

comments in the code, meetings, Git repository structure, code merges and reviews, etc.

Project management lectures began during the second-year project and continue during this third-year project: types of contractual agreements, advanced project planning and monitoring, project steering, and leadership. Also, a case study is analyzed, helping students to understand issues involved in the transition from engineering to management positions. The fact that students had industrial experience as interns just a few months prior to these lectures makes them more receptive to project management concepts and leads to interesting discussions.

Deeper usage of Redmine as a project management tool is mandatory. Each team must detail a work breakdown structure (WBS) of all the tasks of the project. Every week, the teacher reviews it with each team and discusses the projects progress. The progress of each deliverable is monitored and modifications to the original plan are tracked.

Each team must "bid" on a fictive request for proposal. A proposal has to describe the technical solution to the requested system, and also provide the project plan to manage it. This forces students to work on the technical details and project management at the same time. A proposal must also include various important aspects: tests, configuration management, team structures and responsibilities, deliverables details, etc.

Mandatory project management leads to mainly two benefits. Firstly, it makes the weekly visit of each team easier for the teacher because what is supposed to be done is explicitly written. Secondly, at the end of the project, each team will have a base of comparison to assess what went right or wrong and how much time has been spent on each task. Conclusions should be clear and interesting in the final report and the final oral presentation.

7. EVALUATION

Obviously, the quality of the developed system is the most important evaluation aspect of the course. The system evaluation is done once around mid-semester and once at the end of the semester. A rubric is used to evaluate the system on various criteria such as features, user experience and performance.

The system evaluation process is also a good moment to discuss with students about various informal aspects as well, while the teaching staff "experiments" with the developed systems. The fact that a complete system is being developed brings enthusiasm in the laboratory, even during these evaluations.

Reports are another important part of the evaluation, the two most important ones being the response to the request for proposal (RFP) and the final report (and associated documents).

Students are also evaluated on a final oral presentation, a project management exam, the quality of their work planning, and the quality of their time sheets.

Students must also proceed to a self- and peer-assessment, as they do for every other integrative project. As confirmed by Johnston and Miles [7], knowledge that individual contributions to the group project will be assessed results in a low incidence of free-riding and, consequently, more involvement in group-based learning.

Finally, in this third-year project, almost all CEAB graduate attributes are developed but only 6 are formally assessed: 2. problem analysis, 3. investigation, 4. design, 5. use of engineering tools, 6. individual and team work and 12. life-long learning.

8. RESULTS

The third-year computer engineering integrative project course was evaluated by 45 out of the 47 registered students during the winter and fall semesters of 2015 with the standard project evaluation form used at Polytechnique Montréal. Table 1 presents the results to some questions pertaining to the course. These results demonstrate a very good appreciation by the students. Results for previous semesters were similar.

Table 1: Third-year computer engineering project course evaluation results (winter and fall of 2015)

	Perception Results			
	Disagree		**Agree**	
Evaluation questions	**- -**	**-**	**+**	**+ +**
The teacher has paid attention to team aspects	0 %	2%	11%	87%
The skills development was in accordance with the project objectives	0%	6%	19%	74%
Workload is well distributed throughout the semester	5%	9%	14%	73%
Final mark is based on various aspects	0%	2%	11%	87%
The level of difficulty is appropriate for this project	0%	4%	7%	89%
The project is well organized	0%	7%	11%	82%

The third-year software engineering integrative project course was evaluated by 37 out of the 43 registered students during the winter semester of 2015 with the standard project evaluation form used at Polytechnique Montréal. Table 2 presents the results to some questions

pertaining to the course. These results demonstrate a very good appreciation by the students. Results for previous semesters were similar.

Table 2: Third-year software engineering project course evaluation results (winter 2015)

	Perception Results			
	Disagree		Agree	
Evaluation questions	- -	-	+	+ +
The teacher has paid attention to team aspects	3%	11%	33%	53%
The skills development was in accordance with the project objectives	0%	3%	36%	61%
Workload is well distributed throughout the semester	5%	11%	38%	46%
Final mark is based on various aspects	0%	0%	23%	77%
The level of difficulty is appropriate for this project	0%	5%	30%	65%
The project is well organized	0%	5%	27%	68%

Table 1 and 2 present pretty similar results, meaning that both third-year integrative projects are appreciated by students, while some aspects could be improved, such as workload distribution throughout the semester, yet some could argue this aspect is an everlasting complaint.

9. CONCLUSION

One of the reasons explaining the success of our third-year projects is their differences from the two previous ones and the following one. It would be easy to repeat the same project structure each year for four years, but it would bring very little to our curriculums and to the students. Instead, we have designed every single project to fit the specific needs of every curriculum year.

The level of maturity of third-year students in computer and software engineering offers a great opportunity to propose an integrative project based on project management, computer networking, mobile applications and recent technologies specific to a student's program.

The evaluation of third-year project courses is mainly based on the quality of the developed system, the response to the request for proposal, and the final report.

Both third-year integrative projects are appreciated by students, which bodes well for the capstone project the following year.

Acknowledgements

The authors would like to thank the Department of Computer and Software Engineering of Polytechnique Montréal for its support, especially financially thru the purchasing of modern equipment required for these projects. We also wish to underline the good work of Mr. Laurent Tremblay, our analyst, for his continuous help on these projects.

References

[1] Stephen Cass, "The 2015 Top Ten Programming Languages", *IEEE Spectrum*, July 20, 2015. Available as of May 9, 2016 from http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages

[2] Jérôme Collin, "First-Year Integrative Project for Computer and Software Engineering Students at Polytechnique Montréal", in *Proc. CEEA Canadian Engineering Education Conf., CEEA14*, (Canmore, AB, 8-11 June 2014), 6 pp., 2014.

[3] Engineers Canada Accreditation Board, "2015 Accreditation Criteria and Procedures", 2015, 115 pp. Available as of May 9, 2016 from https://www.engineerscanada.ca/sites/default/files/accreditation_criteria_procedures_2015.pdf

[4] Gartner, "Gartner Says Traditional Development Practices Will Fail for Mobile Apps", Available as of May 9, 2016 from http://www.gartner.com/newsroom/id/2823619

[5] Olivier Gendreau, "Second-Year Integrative Project for Computer and Software Engineering Students at Polytechnique Montréal", in *Proc. CEEA Canadian Engineering Education Conf., CEEA14*, (Canmore, AB, 8-11 June 2014), 4 pp., 2014.

[6] Olivier Gendreau and Jérôme Collin, "Les Projets intégrateurs finaux en génie informatique et logiciel: un contexte d'apprentissage favorisant la transition vers le marché du travail", in *Proc. CEEA Canadian Engineering Education Conf.*, CEEA13, (Montréal, QC, 17-20 June 2013), 5 pp., 2013.

[7] Lucy Johnston and Lynden Miles, "Assessing contributions to group assignments," *Assessment & Evaluation in Higher Education*, vol. 29, no. 6, pp. 751-768, 2004.

[8] Richard Prégent, Huguette Bernard, and Anastassis Kozanitis, *Enseigner à l'université dans une approche-programme*. Montréal, QC : Presses internationales Polytechnique, 2009, 330 pp.

[9] Debbie Richards, "Designing Project-Based Courses with a Focus on Group Formation and Assessment", *ACM Transactions on Computing Education*, Vol. 9, No. 1, pp. 2:1-2:40, March 2009.