Second-Year Integrative Project for Computer and Software Engineering Students at Polytechnique Montréal

Olivier Gendreau

Department of Computer and Software Engineering
Polytechnique Montréal
olivier.gendreau@polymtl.ca

Abstract – In 2005, Polytechnique Montréal introduced project-based learning (PBL) in its curriculum. The second-year integrative project for computer and software engineering students is software-oriented, as the main goal is to develop an interactive graphical software application. Students are given a software requirements specification (SRS), which details every feature the application must comply to. The students must then implement these features, starting from a provided framework, and various common software libraries. The main technical concepts integrated in the project are software design patterns, computer graphics, unit testing, and user interface design. Students are also provided with tools to put into practice project management concepts such as scope management, time management, cost management, and risk management. The second-year project is also a good opportunity for students to develop their non-technical skills, such as collaboration, communication and critical thinking.

Keywords: integrative project, project-based learning (PBL), 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 [6]. The second-year integrative project is a mandatory course for all computer and software engineering students. Unlike the first-year integrative project, where hardware is an important part [2], the second-year project is software-oriented. The application to develop is both a map editor/tester and a videogame using the created map. Each year, the specific details of the project change, but the core features remain the same.

Section 2 presents Polytechnique Montréal's first-year, third-year, and senior-year integrative projects. Section 3 details the second-year integrative project in terms of

concept integration and deliverables. Section 4 explains the structure of the course and its support. Section 5 presents the evaluation of the course.

2. INTEGRATIVE PROJECTS

PBL is built around projects. At Polytechnique Montréal, each academic year offers a specific integrative project that aims to put into practice theory seen in other courses of that same academic year. Other than the second-year integrative project, which is the main topic of this paper, there are three other integrative projects: first-year, third-year and senior-year (capstone projects) [3].

2.1. First-Year Integrative Project

In computer engineering and software engineering, the first-year integrative project is based on the design and implementation of a program allowing control of a small mobile robot.

Students start by welding and mechanically assembling the robot they purchased in a box. Then, they get to familiarize with the hardware architecture of a microcontroller, therefore putting into practice knowledge acquired in hardware courses (digital logic and microcomputer architecture). Also, software development concepts learned in the software engineering course are put into practice. At the end of the project, the robot must follow a relatively complex predefined path.

2.2. Third-Year Integrative Projects

Computer engineering and software engineering have different third-year integrative projects, which require more autonomy from students than the prior two integrative projects. In software engineering, the main goal of the project is to reengineer the second-year integrative project in order to add two major features: network multiplayer abilities and mobile (tablet) map edition. In computer engineering, students get to program

a FPGA board, implying embedded systems design. Both projects put into practice computer network concepts.

2.3. Capstone Projects

Capstone projects require even more autonomy from students than the other three integrative projects. From an academic standpoint, it is nearly impossible to define projects related specifically to notions taught in senior-year classes. Unlike the other three integrative projects, each team gets an exclusive capstone project defined by a non-academic client: enterprise or nonprofit organization from different industries. The diversity of available projects usually allows students to work on a project meeting their interests, thus increasing their motivation.

Dealing with real clients means that on top of technical considerations, students must deal with human expectations, which can be even trickier. Capstone projects offer a good opportunity for students to put into practice both technical and non-technical skills.

3. SOFTWARE-ORIENTED PROJECT

The second-year integrative project for computer and software engineering students is software-oriented, unlike first-year integrative project where hardware is an important part of the project.

The application to develop is both a map editor/tester and a videogame using the created map. Each year, the specific details of the project change, but the core features remain the same. Over the past years, students have developed games such as Mars Pathfinder, pinball, pool, car driving, mini golf, and air hockey.

Students are provided with a framework including two layers: a thin Java user interface layer and a C++ application logic layer. Both layers are meant to be extended by students. JNI (Java Native Interface) is used as a bridge allowing native C++ to be invoked by Java. The C++ layer is mainly comprised of a rendering tree using OpenGL.

The framework includes many open source libraries such as Assimp, for 3D models loading, CppUnit, for unit testing, FMOD, for loading and playing music and sounds, FreeImage, for texture loading, FreeType, for text rendering, and TinyXML, for XML saving and loading.

3.1. Concept integration

As prescribed by PBL, one of the main academic goals is concept integration. The second-year project integrates concepts of various courses.

Table 1: Courses concept integration

Course	Integrated concepts
Software design	Object-oriented analysis and design
	Design patterns
Data structure and algorithms	Graph search algorithm
Computer Graphics	Camera control
	3D Modeling
	Texturing
	Lighting
Software Engineering	Unit testing
	Quality metrics
User Interface Design	User interface heuristics
	Windowing system

Table 1 presents a non-exhaustive list of concepts integrated in the second-year project. From the software design course, students must integrate object-oriented analysis and design techniques in order to translate requirements eventually into code. The main design patterns needed in the project are template method, composite, observer, facade, abstract factory, singleton, state, and visitor. From the data structure and algorithms course, students benefit from graph search algorithms. Since the project is a videogame, the computer graphics course is the most important basis: camera control, 3D modeling, texturing, lighting, and many others concepts. From the software engineering course, students integrate unit testing in order to verify critical functions in their project and quality metrics, such as cohesion, coupling, readability, and maintainability. Finally, from the user interface design course, students integrate windowing system concepts and user interface heuristics, such as Bastien-Scapin's [1].

3.2. Deliverables

The second-year integrative project is divided into 3 deliverables.

Table 2: Deliverable 1 features

Feature
Main menu
Menu bar, toolbar
Properties configuration
Pan and zoom view
Resize graphical window
Objects edition
Save and load map
Unit testing

The main goals of deliverable 1 are to allow students to familiarize with the provided framework, to develop a 2D map editor, and to design and implement unit tests. Table 2 presents deliverable 1 features. Students must develop a main menu, a menu bar, a toolbar, and properties configuration possibilities. They must also deal with the windowing system (resize graphical window, pan and zoom view). Most importantly, they must implement objects edition features such as selecting, moving, adding, duplicating, rotating, scaling, and deleting. They must permit saving and loading the edited map in XML format. Finally, they must integrate unit testing for critical functions.

Table 3: Deliverable 2 features

Feature
3D models integration
Options configuration
Game debugging module
Virtual player behaviour
Test mode
Game mode
Campaign mode
Game physics
Game rules

The main goal of deliverable 2 is to add playing capabilities to the application. Table 3 presents deliverable 2 features. Students must implement a test mode, a game mode, a campaign mode and a game debugging module. Therefore, 3D models integration, option configuration, game physics, including collision detection, and game rules must be implemented. Also, a virtual player behavior implementation will provide a very basic form of AI (artificial intelligence).

Table 4: Deliverable 3 features

Feature
Orbit camera
Perspective (3D) edition
Heads-up display (HUD)
Skybox
Lighting
Game sounds
Animations

The main goal of deliverable 3 is to add a few useful bells and whistle in order to provide an enjoyable user experience. Table 4 presents deliverable 3 features. The map editor is migrated to 3D and an orbit camera is added. HUD provides a user interface directly in the OpenGL canvas. A skybox (background environment), lighting, game sounds and animations complete the

deliverable 3 features. Finally, students must optimize the efficiency of their user interface, based on Bastien-Scapin's heuristics.

In addition to the specific features in the 3 deliverables, students must pay attention to the overall quality of the product therefore avoiding graphics lag, memory leaks and crashes. User documentation (Javadoc and Doxygen) is also mandatory. Finally, students must deliver their product as a self-executable package.

Figure 1 presents an example of a second-year project application: Air Hockey.

Fig. 1. Air Hockey main menu

4. COURSE STRUCTURE AND SUPPORT

On the first day of the course, teams of 5 or 6 students are formed and provided with a framework and all the documentation they need for the whole semester: project presentation, software requirements specification (SRS), elicitation report, and development guide. The project presentation (about 10 pages long) explains the general context of the project and presents high level features to develop. The SRS is a more formal document (about 20 pages long), very common in traditional software development, detailing the specific requirements of the system. The elicitation report (about 40 pages long) is written in a form of a Q&A between a developer and a client. Its purpose is to explain into even greater details the requirements of the project, in a less formal way than in the SRS. Finally, the development guide (about 40 pages long) offers technical information and advices related to the provided framework.

Students get more autonomy in the second-year integrative project than in the first-year one. It is their first full-semester project. There is no week-by-week assignment, only 3 deliverables for the whole semester. In order to help students, 4 lectures are given on project management theory: scope management, time management, cost management, and risk management.

Also, students are provided with a project management tool, Redmine [7], which offers a Web interface allowing task definition and scheduling, timesheet management, and many other support tasks.

Since computer and software projects deal with a lot of files, configuration management tools are essential. In the second-year integrative project, students are provided with Git [4], a popular distributed revision control and source code management system. With Redmine, it is possible to link tasks to files controlled by Git, which is very useful for process development control such as requirements traceability.

Teaching assistants are available 6 hours a week to answer technical questions and the teacher does a weekly supervision meeting.

Finally, a non-technical skills specialist meets each team around mid-project in order to help students take advantage of effective collaboration, communication and critical thinking.

5. EVALUATION

The main goal of the second-year integrative project course is to develop a project application, but the evaluation of the course is diversified.

Table 5: Course evaluation

Evaluation	Weight	Type
Project (3 deliverables)	50 %	Team
Project management exam	10 %	Individual
Technical report	10 %	Individual
Technical exam	10 %	Individual
Oral presentation	10 %	Individual
Timesheet	5 %	Individual
Work planning	5 %	Team

Table 5 presents second-year project course evaluation. While the team evaluation of the project stands for half of the evaluation of the course, there are a few individual evaluations: project management exam, technical exam, technical report, and oral presentation. Also, students must take advantage of the Redmine tool to plan development of the project (team evaluation) and also log their timesheet (individual evaluation).

A peer-reviewing Web tool, iPeer [8], is used. After each of the three deliverables, every student must proceed to a self- and peer-assessment on four criteria: team role fulfillment, team interaction, team performance contribution, and team development contribution. As confirmed by Johnston and Miles [5], 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.

6. CONCLUSION

Second-year integrative project is an important part of computer engineering and software engineering bachelor

degrees. The progression of a student's autonomy from first-year project to capstone project is very significant and the second-year project is an important part of the progression. The project integrates many concepts such as software design patterns, computer graphics, unit testing, and user interface design. Students are also provided with tools to put into practice project management concepts. Also, the second-year project is a good opportunity for students to develop their non-technical skills.

Ever since its creation, the course has undergone continuous improvements: small ones such as the introduction of a new configuration management tool, or bigger ones such as the migration of the provided framework from a 32-bit version to a 64-bit version.

Since computer and software technologies evolve very fast, it is a great challenge to keep up to date. Currently, the biggest improvement yet is underway: a new version of the framework is being developed, involving cutting-edge technologies. Hopefully it will last a few years!

Acknowledgements

The author would like to thank the Department of Computer and Software engineering of Polytechnique Montréal for its support of the second-year integrative project initiatives over the last 8 years, especially Benoit Ozell, Michel Desmarais, Laurent Tremblay, and Steven Chamberland.

References

- [1] Christian Bastien and Dominique Scapin, *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces*, INRIA technical report, no 156, 1993.
- [2] Jérôme Collin, "First-Year Integrative Project for Computer and Software Engineering Students at Polytechnique Montréal", in *Proc. CEEA Canadian Engineering Education Conf., CEEAI4*, [to be published] (Canmore, AB, 8-11 June 2014), 6 pp., 2014.
- [3] 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., CEEAI3*, (Montréal, QC, 17-20 June 2013), 5 pp., 2013.
- [4] Git [En ligne], <http://git-scm.com/>
- [5] Lucy Johnston and Lynden Miles, "Assessing contributions to group assignments," *Assessment & Evaluation in Higher Education*, vol. 29, no. 6, pp. 751-768, 2004.
- [6] Richard Prégent, Huguette Bernard, and Anastassis Kozanitis, *Enseigner à l'université dans une approche-programme*. Montréal, QC : Presses internationales Polytechnique, 2009, 330 pp.
- [7] Redmine [Online], <http://www.redmine.org/>
- [8] University of British Columbia, iPeer [Online], <http://ipeer.citl.ubc.ca/>