MONASH University
Information Technology

FIT3140
Advanced programming

Unit Guide

Summer semester, 2014

Last updated: 18 Nov 2014

Table of Contents

FIT3140 Advanced programming - Summer semester, 2014

This unit develops the students' ability to design, implement and maintain moderately complex, realistically-sized programs using an Agile software development methodology. It builds upon the basic programming techniques introduced in introductory programming unit and offers the first introduction to the implementation of more complex real-world programs. Examples of such systems include compilers and interpreters, simulations, visualisation tools, drawing packages, database systems, graphical games. Such systems may be implemented in the context of non-traditional computing environments such as smartphone "apps". The unit may offer students the opportunity to get acquainted with a second programming language within the procedural-object oriented paradigm, such as C++, Python or one of their cousins, depending on the scope of the project chosen in a particular semester.

The unit bridges between core programming knowledge and the large-scale software engineering context. It will emphasise the implementation and use of intermediate to advanced data structures (such as search trees, hash structures, graphs and graph algorithms etc.) and the embedding into an actual computing system (i.e. interacting with the O/S, networking components etc).

Mode of Delivery

Clayton Summer semester A (Day)

Workload Requirements

Minimum total expected workload equals 12 hours per week comprising:

(a.) Contact hours for on-campus students:

- Two hours of lectures
- One 3-hour laboratory

(b.) Additional requirements (all students):

- A minimum of 7 hours independent study per week for completing lab and assignment work, private study and revision

Additional workload requirements

Substantial parts of the project work must be completed in pairs. Students will need to schedule time to meet with their project partners outside scheduled classes.

Unit Relationships

Prerequisites

FIT1008

Chief Examiner

Dr Robert Merkel

Campus Lecturer

Clayton

Robyn McNamara

Tutors

Clayton

Robyn McNamara

Michael Gill

Your feedback to Us

Monash is committed to excellence in education and regularly seeks feedback from students, employers and staff. One of the key formal ways students have to provide feedback is through the Student Evaluation of Teaching and Units (SETU) survey. The University's student evaluation policy requires that every unit is evaluated each year. Students are strongly encouraged to complete the surveys. The feedback is anonymous and provides the Faculty with evidence of aspects that students are satisfied and areas for improvement.

For more information on Monash's educational strategy, see:

www.monash.edu.au/about/monash-directions/ and on student evaluations, see:
www.policy.monash.edu/policy-bank/academic/education/quality/student-evaluation-policy.html

Previous Student Evaluations of this Unit

The unit received reasonably high SETU ratings last time it was run, and there were no major issues identified.

The major change to the unit this semester is due to changes in the prerequisite units, as students no longer learn Java in FIT1008. This has meant that we have switched to Python and Kivy as our tools this semester. The previous native Android toolchain was not popular with students, as it was difficult to use, slow, and buggy. While this will obviously change the specifics of the programming techniques learned this semester, the design and management aspects of the unit have not changed significantly.

If you wish to view how previous students rated this unit, please go to
https://emuapps.monash.edu.au/unitevaluations/index.jsp

Academic Overview

Learning Outcomes

Upon successful completion of the unit, students will have an understanding of:

- agile software development practices including iteration, test-driven development, spiking, and continuous customer involvement;
- how to design moderately complex programs where that design will typically incorporate a number of modules and a number of levels of refinement;
- the role of software architecture in program design and a knowledge of a number of commonly-applied software architectures;
- how to make use of design patterns, re-usable components and software libraries in designing modular software;
- how to make design decisions that take into account desirable quality attributes such as flexibility, maintainability and re-usability;
- how to implement programs in a systematic manner using an integrated testing procedure in such a way that modules are highly likely to function as specified;
- how to isolate faults within a program in a systematic manner;
- how to use software tools to aid in the program design and implementation process. These tools might include program design tools, integrated program development environments, configuration management systems, re-factoring tools, automatic testing environments and debuggers;
- how to adequately document a software project.

They will have developed attitudes that enable them to:

- evolve a software system in response to feedback over time;
- recognise the importance of process in achieving quality in a repeatable manner;
- appreciate the distinction between analysis of program requirements and design that seeks to meet specifications;
- develop software designs that place appropriate importance to the user experience;
- adopt an approach to making design decisions that involves considering a range of options for design decisions and evaluating potential design decisions with reference to a system of values;
- evaluate product and process development with the aim of continuously improving their software development methodology;
- understand the importance of being able to communicate all aspects of the program development process, and identify the most appropriate medium for that communication.

They will have developed the skills to:

- design moderately complex, real-world programs where that design involves multiple levels of refinement and the specification of a non-trivial number of modules;
- learn a new programming language or environment efficiently when that programming language or environment is similar to a programming language the student already knows;
- develop software in a modern software environment that may include software development tools such as those found in an integrated, programming environment, configuration management systems and automated testing systems;
- design and implement programs that can interface with complex software systems such as graphical-user interfaces, database systems and mathematical libraries;
- design and implement programs that may need to communicate via a computer network with software systems on other computer devices;
- design and implement systems on alternative computing platforms including smartphones and embedded systems;

- identify performance-critical aspects of a software system and learn to apply their analysis skills in larger systems.

They will have demonstrated the communication skills to:

- create design documents that can be used to present a view of the software to other stakeholders;
- identify items that are insufficiently clear from written documentation and proactively seek clarification from stakeholders;
- create documents or on-line help that enable people to understand how to use the program;
- create documents that allow a programmer to understand the program in sufficient detail to allow the software to be maintained;
- produce literate programs, i.e. program source statements that are well commented.

Unit Schedule

Week	Activities	Assessment
0		No formal assessment or activities are undertaken in week 0
1	Unit overview, the problem domain, the development environment, introduction to Agile	Assignment 1 due Friday 28 November 2014
2	project lifecycles, requirements gathering, design and modeling, user interfaces	Assignment 2 due Friday 5 December 2014
3	performance analysis, software quality	Assignment 3 due Friday 12 December 2014
4	Agile practices, software engineering tools, releasing software, unit summary	Assignment 4 due Friday 19 December 2014
5		Assignment 5 due Monday 6 January 2015
6		
7		
8		
9		
10		
11		
12		
	Examination period	LINK to Assessment Policy: http://policy.monash.edu.au/policy-bank/academic/education/assessment/assessment-in-coursework-policy.html

*Unit Schedule details will be maintained and communicated to you via your learning system.

Teaching Approach

- **Lecture and tutorials or problem classes**
  This teaching and learning approach provides facilitated learning, practical exploration and peer learning.
- **Laboratory-based classes**
  This learning and teaching approach provides the opportunity for practical experimentation with approaches taught in lectures.

Assessment Summary

Examination (3 hours): 50%; In-semester assessment: 50%

Assessment Task	Value	Due Date
Spiking and vision statement	5%	

		Friday 28 November
Requirements, architecture and risk analysis (pairs)	10%	Friday 5 December
Iteration 1 (pairs)	15%	Friday 12 December
Iteration 2 (pairs)	15%	Friday 19 December
Final report (individual)	5%	Monday 6 January
Examination 1	50%	To be advised

Assessment Requirements

Assessment Policy

Faculty Policy - Unit Assessment Hurdles
(http://intranet.monash.edu.au/infotech/resources/staff/edgov/policies/assessment-examinations/assessment-hurd

Academic Integrity - Please see resources and tutorials at
http://www.monash.edu/library/skills/resources/tutorials/academic-integrity/

Assessment Tasks

Participation

Students are expected to attend lectures and lab classes.

While lab exercises will not be marked, many of the labs will be used for project work in pairs, including assessment. As such, students will need to attend these.

Students will identify a project partner to work with by the end of the first laboratory, who is enrolled in the same lab class. Students will work with their partner for subsequent in-semester assessment unless there are exceptional circumstances.

- **Assessment task 1**

Title:
Spiking and vision statement
Description:
Students will *individually* complete prototype proof-of-concept "spiking" exercises to reduce the technical risk associated with the project theme in future assignments.

Each individual spike will consist of some exploratory/proof-of-concept coding, and a report which documents the information gained.

Students will, *in pairs*, also complete a "vision statement", i.e. a brief, high-level description of the key requirements of the project which will form the basis for subsequent assignment tasks.
Weighting:
5%
Criteria for assessment:

- Reporting, including:

- risk description
- description of coding tasks
- clear reporting of outcome
- assessment of risk after spiking.

- Quality of coding practices
- Quality of code documentation.
- Appropriateness of content of vision statement.
- Quality of presentation of vision statement.

Due date:

Friday 28 November

- **Assessment task 2**

Title:

Requirements, architecture and risk analysis (pairs)

Description:

Project pairs will

- Describe the user requirements for the system they will build and capture these as "user stories".
- Devise, and describe using appropriate notation including UML, a high-level system architecture capable of supporting the requirements.
- Prepare a risk analysis document that outlines potential problems that the project team may face, and details the actions that team members will take to avoid or ameliorate them.

Weighting:

10%

Criteria for assessment:

- Effectively capturing key functional requirements.
- Identification and description of key attributes of system context.
- Identification and analysis of likely risks
- Designing feasible misk mitigation strategies.
- Completing a design that supports functional and non-functional (including quality) requirements.
- Appropriate use of standard notations for capturing design.
- Readability and presentation quality of documents.

This assignment will be completed in pairs. Pairs are expected to share the workload and will usually receive the same mark. Individual contributions will be documented, and if workloads are not appropriately shared over the semester marks will be adjusted accordingly.

Due date:

Friday 5 December

- **Assessment task 3**

Title:

Iteration 1 (pairs)

Description:

Project pairs will complete their first iteration of the active development phase of the project.

They will deliver working and tested software implementing several "user stories".

The deliverables will also include release notes and user documentation as required.

Weighting:

15%

Criteria for assessment:

- Completeness of implementation.
- Quality of implementation.
- Appropriateness of functional testing.

- Rigorousness of performance tuning
- Correct use of provided tools.
- Appropriate supply of documentation.

Pairs are expected to share the workload and will usually receive the same mark. Individual contributions will be documented, and if workloads are not appropriately shared over the semester marks will be adjusted accordingly.

Due date:

Friday 12 December

- **Assessment task 4**

Title:

Iteration 2 (pairs)

Description:

Project pairs will complete two iterations of the active development phase of the project. They will deliver working and tested software implementing several "user stories" as developed in earlier assignments. In the first iteration (for initial submission) they will complete initial basic functionality; in the second iteration, as well as adding additional functionality they will refine the user interface and other non-functional aspects. The deliverables will also include release notes and user documentation as required.

Weighting:

15%

Criteria for assessment:

- Completeness of implementation.
- Quality of implementation.
- Appropriateness of functional testing.
- Ease of use, including responsiveness of user interface.
- Appropriate level of documentation.

Due date:

Friday 19 December

- **Assessment task 5**

Title:

Final report (individual)

Description:

Students will reflect on their project work, including:

- Reflecting on the overall quality of the deliverables to date.
- Characterising their individual contribution to the project and that of their partner.
- Identifying the strengths and weaknesses of the development processes used in the context of the current project.
- Determining a realistic timeframe for completion of the full vision as described earlier in semester.
- Describing the process they would follow to achieve completion.

Weighting:

5%

Criteria for assessment:

- Quality and depth of reflection.
- Plausibility of analysis of future prospects.
- Quality of proposed process.

♦Clarity of written report.

Due date:

Monday 6 January

Examinations

- **Examination 1**

Weighting:

50%

Length:

3 hours

Type (open/closed book):

Closed book

Electronic devices allowed in the exam:

None

Learning resources

Reading list

- The Scrum Primer. Free download: http://www.scrumprimer.com/
- Scrum and XP from the Trenches by Henrik Kniberg. Free download: http://www.infoq.com/minibooks/scrum-xp-from-the-trenches

Monash Library Unit Reading List (if applicable to the unit)
http://readinglists.lib.monash.edu/index.html

Faculty of Information Technology Style Guide

Feedback to you

Examination/other end-of-semester assessment feedback may take the form of feedback classes, provision of sample answers or other group feedback after official results have been published. Please check with your lecturer on the feedback provided and take advantage of this prior to requesting individual consultations with staff. If your unit has an examination, you may request to view your examination script booklet, see
http://intranet.monash.edu.au/infotech/resources/students/procedures/request-to-view-exam-scripts.html

Types of feedback you can expect to receive in this unit are:

- Informal feedback on progress in labs/tutes
- Graded assignments with comments

Extensions and penalties

Submission must be made by the due date otherwise penalties will be enforced.

You must negotiate any extensions formally with your campus unit leader via the in-semester special consideration process: http://www.monash.edu.au/exams/special-consideration.html

Returning assignments

Students can expect assignments to be returned within two weeks of the submission date or after receipt, whichever is later.

Resubmission of assignments

No resubmission of assignments will be permitted.

Referencing requirements

Code based on algorithms or information from third-party sources (such as books or websites) must acknowledge these sources in comments, in sufficient detail for markers to find and check these sources. Some assignments will explicitly prohibit the use of this - if so, you must not use them.

Where third party material is used in written assignments, they should be cited. Students may use any of the referencing methods described in the Monash University Library's tutorial on referencing to do so:

http://guides.lib.monash.edu/content.php?pid=88267&sid=656564

Assignment submission

It is a University requirement (http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-managing-pla for students to submit an assignment coversheet for each assessment item. Faculty Assignment coversheets can be found at http://www.infotech.monash.edu.au/resources/student/forms/. Please check with your Lecturer on the submission method for your assignment coversheet (e.g. attach a file to the online assignment submission, hand-in a hard copy, or use an online quiz). Please note that it is your responsibility to retain copies of your assessments.

Online submission

If Electronic Submission has been approved for your unit, please submit your work via the learning system for this unit, which you can access via links in the my.monash portal.

Recommended Resources

Students will develop for Android using the Google SDK integrated with Eclipse. Eclipse will be provided in laboratory work, but students may choose to use any Android development setup that they prefer..

Students may choose to use their own laptops if they wish; however, they will be responsible for all technical support and ensuring that their code runs on the specified platform.

Recommended text(s)

Kent Beck, Cynthia Andres, Erich Gamma. (). *Extreme Programming explained*. (2nd Edition) Addison-Wesley Professional (ISBN: 978-0321278654).

Examination material or equipment

Students should refer to the unit website for details about the exam.

Other Information

Policies

Monash has educational policies, procedures and guidelines, which are designed to ensure that staff and students are aware of the University's academic standards, and to provide advice on how they might uphold them. You can find Monash's Education Policies at: www.policy.monash.edu.au/policy-bank/academic/education/index.html

Key educational policies include:

- Student Academic Integrity Policy and Student Academic Integrity: Managing Plagiarism and Collusion Procedures ; http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-policy.h
- Assessment in Coursework Programs; http://www.policy.monash.edu/policy-bank/academic/education/assessment/assessment-in-coursework-po
- Special Consideration; http://www.policy.monash.edu/policy-bank/academic/education/assessment/special-consideration-policy.ht
- Grading Scale; http://www.policy.monash.edu/policy-bank/academic/education/assessment/grading-scale-policy.html
- Discipline: Student Policy; http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-discipline-policy.html
- Academic Calendar and Semesters; http://www.monash.edu.au/students/dates/
- Orientation and Transition; http://intranet.monash.edu.au/infotech/resources/students/orientation/
- Academic and Administrative Complaints and Grievances Policy; http://www.policy.monash.edu/policy-bank/academic/education/management/complaints-grievance-policy.h

Faculty resources and policies

Important student resources including Faculty policies are located at http://intranet.monash.edu.au/infotech/resources/students/

Graduate Attributes Policy

http://www.policy.monash.edu/policy-bank/academic/education/management/monash-graduate-attributes-policy.ht

Student Charter

www.opq.monash.edu.au/ep/student-charter/monash-university-student-charter.html

Student services

The University provides many different kinds of support services for you. Contact your tutor if you need advice and see the range of services available at http://www.monash.edu.au/students. For Malaysia see http://www.monash.edu.my/Student-services, and for South Africa see http://www.monash.ac.za/current/.

Monash University Library

The Monash University Library provides a range of services, resources and programs that enable you to save time and be more effective in your learning and research. Go to www.lib.monash.edu.au or the library tab in my.monash portal for more information. At Malaysia, visit the Library and Learning Commons at http://www.lib.monash.edu.my/. At South Africa visit http://www.lib.monash.ac.za/.

Disability Liaison Unit

Students who have a disability or medical condition are welcome to contact the Disability Liaison Unit to discuss academic support services. Disability Liaison Officers (DLOs) visit all Victorian campuses on a regular basis.

- Website: http://www.monash.edu/equity-diversity/disability/index.html
- Telephone: 03 9905 5704 to book an appointment with a DLO; or contact the Student Advisor, Student Commuity Services at 03 55146018 at Malaysia
- Email: dlu@monash.edu
- Drop In: Equity and Diversity Centre, Level 1, Building 55, Clayton Campus, or Student Community Services Department, Level 2, Building 2, Monash University, Malaysia Campus