



MONASH University

**FIT3077**  
**Software engineering: architecture and design**

**Unit Guide**

**Semester 2, 2009**

The information contained in this unit guide is correct at time of publication. The University has the right to change any of the elements contained in this document at any time.

*Last updated : 06 Aug 2009*

# Table of Contents

<b><u>FIT3077 Software engineering: architecture and design - Semester 2, 2009</u></b> .....	<b>1</b>
<u>Chief Examiner:</u> .....	1
<u>Lecturer(s) / Leader(s):</u> .....	1
<u>Clayton</u> .....	1
<u>Introduction</u> .....	2
<u>Unit synopsis</u> .....	2
<u>Learning outcomes</u> .....	2
<u>Contact hours</u> .....	3
<u>Workload</u> .....	3
<u>Unit relationships</u> .....	3
<u>Prerequisites</u> .....	3
<u>Prohibitions</u> .....	3
<u>Relationships</u> .....	3
<u>Teaching and learning method</u> .....	4
<u>Timetable information</u> .....	4
<u>Tutorial allocation</u> .....	4
<u>Unit Schedule</u> .....	4
<u>Unit Resources</u> .....	7
<u>Prescribed text(s) and readings</u> .....	7
<u>Recommended text(s) and readings</u> .....	7
<u>Required software and/or hardware</u> .....	7
<u>Equipment and consumables required or provided</u> .....	7
<u>Study resources</u> .....	7
<u>Assessment</u> .....	9
<u>Overview</u> .....	9
<u>Faculty assessment policy</u> .....	9
<u>Assignment tasks</u> .....	9
<u>Examination</u> .....	10
<u>Due dates and extensions</u> .....	10
<u>Late assignment</u> .....	10
<u>Return dates</u> .....	11
<u>Appendix</u> .....	12

# **FIT3077 Software engineering: architecture and design - Semester 2, 2009**

## **Chief Examiner:**

**Dr David Squire**

Senior Lecturer

Phone: +61 3 990 59013

Fax: +61 3 990 55159

## **Lecturer(s) / Leader(s):**

### **Clayton**

**Dr Peter Tischer**

Senior Lecturer

Phone: +61 3 990 55208

Fax: +61 3 990 55159

## Introduction

Welcome to FIT3077 'Software Engineering: Architecture and Design' for semester 2, 2009. This 6 point unit is core to both the Bachelor of Software Engineering and the Bachelor of Computer Science. In this unit you will learn about both large and small scale software architecture and design, including both design and analysis patterns, architectural patterns such as Model-View-Controller, and emerging topics such as service-oriented architectures and commercial-off-the-shelf components (COTS). You will learn about incremental design improvement through refactoring. These designs, patterns, and architectures will be described using the Unified Modeling Language (UML).

## Unit synopsis

This unit builds on introductory units to analysis and design. It provides the professional software engineer with advanced knowledge and skills in high-level architectural design, its theoretical foundations, industrial best practice, and relevant application context. In the software life-cycle, software architecture sits between analysis/specification and design/implementation. The field of software architecture has come of age with a thriving research community and numerous high-level models, methods, tools and practices widely used in industry.

## Learning outcomes

At the completion of this unit students will have knowledge and understanding of:

1. modelling and design of flexible software at the architectural level, basics of model-driven architecture;
2. architectural styles and patterns, middleware & application frameworks;
3. product lines, design using COTs software;
4. configurations and configuration management;
5. in-depth look at software design, design patterns;
6. design of distributed systems using middleware;
7. design for qualities such as performance, safety, reusability, etc;
8. evaluation and evolution of designs, reverse engineering.

At the completion of this unit students will have developed attitudes that enable them to:

1. apply variety of design patterns;
2. appreciate analysis fundamentals, analyse well-formedness (completeness, consistency, robustness, etc);
3. analyse correctness (eg. static analysis, simulation, etc.);
4. analyse quality requirements (eg. root cause analysis, safety, usability, security, etc.).

At the completion of this unit students will have the skills to:

1. take requirements for simple systems and develop software architectures and designs at a high level;
2. use configuration management tools effectively;
3. apply a variety of frameworks and architectures in designing software.

## Contact hours

3 x contact hrs/week

## Workload

Workload commitments are:

- two-hour lecture and
- one-hour practice class
- A minimum of 2-3 hours of personal study per one hour of contact time in order to satisfy the reading and assignment expectations.

## Unit relationships

### Prerequisites

FIT2001 or CSE2305, FIT2004 or CSE2304

### Prohibitions

CSE3308

### Relationships

FIT3077 is a core unit in the Bachelor of Software Engineering and the Bachelor of Computer Science.

Before attempting this unit you must have satisfactorily completed FIT2001 or CSE2305, and FIT2004 or CSE2304, or equivalent.

You may not study both this unit and CSE3308 in your degree.

## Teaching and learning method

There will be two one-hour lectures per week in this unit. During lectures, the lecturer will present the theory underlying the topics in the unit, and illustrate this with concrete examples.

There will also be a one-hour 'practice class' for the unit. The lecturer and tutors (if any) will be present for the practice class. Students are expected to work on example problems or their assignments during the practice class, and ask the teaching staff for assistance when necessary. Teaching staff may occasionally choose to explain a common problem to the whole class.

Note that a practice class is *not a tutorial*. It is time for you to work on the example problems provided, with a staff member present to answer any questions you might have. To get value from the practice class, it is vital that you attempt the example problems provided.

The practice classes are intended to be the primary mechanism for consultation in this unit.

## Timetable information

For information on timetabling for on-campus classes please refer to MUTTS, <http://mutts.monash.edu.au/MUTTS/>

## Tutorial allocation

On-campus students should register for tutorials/laboratories using the Allocate+ system:

<http://allocate.cc.monash.edu.au/>

## Unit Schedule

Week	Topic	References/Readings	Key dates
1	Introduction to FIT3077; What is Software Architecture?; Object-Oriented Analysis using UML	Bass L., Clements P. and Kazman R. Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003, Ch. 2.; Fowler, Martin, UML Distilled, Addison-Wesley, 1997, 2000, or 2003.	
2	Object-Oriented Analysis using UML	Fowler, Martin, UML Distilled, Addison-Wesley, 1997, 2000, or 2003.	Assignment One specification available
3	Principles of Object-Oriented Analysis and Design; Design Patterns	Page-Jones, Meilir, Fundamentals of Object-Oriented Design in UML, Addison-Wesley, 2000 (Ch. 8, 9); Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissides, John, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, Chs. 1, 3, 4, 5.	

4	Principles of Object-Oriented Design	Martin, Robert C., Design Principles and Design Patterns, 2000 (available via Blackboard).	Assignment One due
5	Principles of Object-Oriented Design	Martin, Robert C., Design Principles and Design Patterns, 2000 (available via Blackboard).; Martin, Robert C., Granularity, 1997 (available via Blackboard).	Assignment Two, Stage One specification available
6	Design Principles and Design Patterns	Martin, Robert C., Design Principles and Design Patterns, 2000 (available via Blackboard).	
7	Analysis Patterns; Refactoring	Fowler, Martin, Analysis Patterns: Reusable Object Models, Addison-Wesley, 1997 (Chs. 1, 2); Fowler, Martin, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999, Chs. 1, 2, 7.	
8	Software Architecture; Architectural Structures	Bass L., Clements P. and Kazman R. Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003, Ch. 2.	
9	Documenting Software Architectures; The Model-View-Controller Architectural Pattern	Bass L., Clements P. and Kazman R. Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003, Ch. 9.; Microsoft Corporation, Model-View-Controller, Microsoft Patterns & Practices Developer Center, 2008.	Assignment Two, Stage One due; Assignment Two, Stage Two specification available
10	Architecture and Design with COTS components	Bass L., Clements P. and Kazman R. Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003, Ch. 18.	
Mid semester break			
11	Software Product Lines: Re-using Architectural Assets	Bass L., Clements P. and Kazman R. Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003, Ch. 14.	
12	Service Orientation; Service-Oriented Architecture	Allen, Paul and Schlamann,	Assignment Two,

		Hermann, Service Orientation: Winning Strategies and Best Practices, Cambridge University Press, 2006, Chs. 1, 3, 7	Stage Two due
13	Revision		



## Unit Resources

### Prescribed text(s) and readings

There is no set text for this unit.

### Recommended text(s) and readings

- Fowler M., UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley, 3rd ed., 2003.
- Gamma E., Helm R., Johnson R., Vlissides J. M., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
- Martin R. C., Design Principles and Design Patterns, 2000.

◆ This and other Robert C. Martin articles are made available through the Blackboard site for the unit.

- Fowler M., Analysis Patterns: Reusable Object Models, Addison-Wesley, 1996.
- Fowler M., Beck K., Brant J., Opdyke W., Roberts D., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
- Bass L., Clements P. and Kazman R., Software Architecture in Practice, Addison-Wesley, 2nd ed., 2003.

◆ full text available electronically through the Monash library

- Allen, P and Schlamann, H., Service Orientation: Winning Strategies and Best Practices, Cambridge University, 2006.

◆ full text available electronically through the Monash library

### Required software and/or hardware

You will need access to:

- A tool for creating UML diagrams, such as Rational Rose, Visual Paradigm, etc.
- An object-oriented programming language

On-campus students may use this software which is installed in the computing labs. Information about computer use for students is available from the ITS Student Resource Guide in the Monash University Handbook.

### Equipment and consumables required or provided

Students studying off-campus are required to have the minimum system configuration specified by the Faculty as a condition of accepting admission, and regular Internet access. On-campus students, and those studying at supported study locations may use the facilities available in the computing labs. Information about computer use for students is available from the ITS Student Resource Guide in the Monash University Handbook. You will need to allocate up to 9 hours per week for use of a computer, including time for newsgroups/discussion groups.

### Study resources

Study resources we will provide for your study are:

- Weekly detailed lecture notes outlining the learning objectives, discussion of the content, and required readings;

FIT3077 Software engineering: architecture and design - Semester 2, 2009

- Regular practice class exercises, with feedback and solutions provided *in the practice classes*;
- Assignment specifications;
- A sample examination and suggested solutions presented *in the week 13 lectures*;
- Discussion groups;
- This Unit Guide outlining the administrative information for the unit;
- The unit web site on MUSO, where resources outlined above will be made available.

## Assessment

### Overview

Exam: 40%

practical assignment: 60%

### Faculty assessment policy

To pass a unit which includes an examination as part of the assessment a student must obtain:

- 40% or more in the unit's examination, and
- 40% or more in the unit's total non-examination assessment, and
- an overall unit mark of 50% or more.

If a student does not achieve 40% or more in the unit examination or the unit non-examination total assessment, and the total mark for the unit is greater than 44% then a mark of no greater than 44-N will be recorded for the unit.

The unit is assessed with two assignments and a three hour closed book examination. To pass this unit, a student must obtain:

- 40% or more in the unit's examination and
- 40% or more in the unit's non-examination assessment, and
- an overall unit mark of 50% or more

If a student does not achieve 40% or more in the unit examination or the unit non-examination assessment then a mark of no greater than 44-N will be recorded for the unit.

### Assignment tasks

#### Assignment coversheets

Assignment coversheets are available via "Student Forms" on the Faculty website:

<http://www.infotech.monash.edu.au/resources/student/forms/>

You MUST submit a completed coversheet with all assignments, ensuring that the plagiarism declaration section is signed.

**Assignment submission and return procedures, and assessment criteria will be specified with each assignment.**

#### • Assignment task 1

**Title:**

UML Design Assignment

**Description:**

Students will be provided with software requirements scenarios for which they must produce a conceptual design for an object-oriented solution. This design must be described using UML diagrams. The purpose of this assignment is to reinforce UML knowledge and ensure that students are ready for the large assignment.

**Weighting:**

20%

**Due date:**

Week 4

• **Assignment task 2**

**Title:**

Team Software Architecture Assignment

**Description:**

This is a team assignment. Students will work on it in pairs. This assignment will have two stages, each requiring students to analyse a problem, design a solution, and implement it in an object-oriented programming language. The second stage will build on the first, and the ease with which the students' stage one design can be extended for stage two will depend on appropriate use of architecture and design knowledge presented in this unit.

**Weighting:**

40%

**Due date:**

Week 12

## Examination

- **Weighting:** 40%

**Length:** 3 hours

**Type (open/closed book):** Closed book

**See Appendix for End of semester special consideration / deferred exams process.**

## Due dates and extensions

Please make every effort to submit work by the due dates. It is your responsibility to structure your study program around assignment deadlines, family, work and other commitments. Factors such as normal work pressures, vacations, etc. are not regarded as appropriate reasons for granting extensions. Students are advised to NOT assume that granting of an extension is a matter of course.

Students requesting an extension for any assessment during semester (eg. Assignments, tests or presentations) are required to submit a Special Consideration application form (in-semester exam/assessment task), along with original copies of supporting documentation, directly to their lecturer within two working days before the assessment submission deadline. Lecturers will provide specific outcomes directly to students via email within 2 working days. The lecturer reserves the right to refuse late applications.

A copy of the email or other written communication of an extension must be attached to the assignment submission.

Refer to the Faculty Special consideration webpage or further details and to access application forms:

<http://www.infotech.monash.edu.au/resources/student/equity/special-consideration.html>

## Late assignment

Assignments received after the due date will receive a mark of  $M \cdot (0.9)^n$ , where M is the raw mark for the assignment, and n is the number of days that the assignment is late (including weekends).

## **Return dates**

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

## Appendix

Please visit the following URL: <http://www.infotech.monash.edu.au/units/appendix.html> for further information about:

- Continuous improvement
- Unit evaluations
- Communication, participation and feedback
- Library access
- Monash University Studies Online (MUSO)
- Plagiarism, cheating and collusion
- Register of counselling about plagiarism
- Non-discriminatory language
- Students with disability
- End of semester special consideration / deferred exams