

Subject Number :	48433
Subject Name :	Software Architecture
Teaching Unit	Engineering
Credit Points:	6 cp
Modes of Presentation:	Weekly
Applicable Courses:	BE DipEngPrac; BE; BEngSc; and associated combined degrees
Subject Availability:	
Restrictions:	
Pre-requisites:	48024 Object-Oriented Design
Co-Requisites:	n/a
Anti-Requisites:	n/a

Aims:

This subject aims to provide the student with a solid grounding in the fundamental concepts and practices of software architecture. It assumes that students have a good understanding of the basics of programming and software construction, and of the principles of software engineering and software lifecycles. Through a set of integrated lectures, tutorials, laboratories, and assignments, the student is exposed to and develops skills in the analysis and design of more complex software systems from an *architectural* perspective. After completing the subject, students will be able to:

- Understand and describe the system context in terms of stakeholders, quality attributes, and constraints and enablers.
- Develop and refine multiple views of a software system architecture, based on the conceptual, execution, and implementation viewpoints.
- Understand key issues in implementing distributed and real-time systems, including networking, distribution, performance, testing and reliability, and real-time qualities.
- Implement architectural prototypes using off-the-shelf components, middleware, and custom code.

Contribution to Overall Course Aims:**Stages 3 & 4 - Focus on developing a set of field of practice skills and knowledge**

Facilitate students' development of specialist skills and knowledge, and strategies for doing so in one or more fields of practice.

- developing professional communication skills
- developing a theoretical base for their field of practice knowledge
- applying core skills and knowledge to a field of practice

Stages 5 & 6 - Focus on applications and understanding workplace contexts

Provide opportunities for students to integrate core and fields of practice specialist knowledge in project based work; and to examine how the knowledge applies in a workplace context.

- applying tools and methodologies of a field of practice in a design project
- developing knowledge of organisational and social contexts of a chosen field of practice
- applying skills and academic knowledge in a workplace

Stages 7 & 8 - Focus on becoming a reflective practitioner

Provide situations and resources for developing an awareness and facility for critically reflective practice.

- Developing a critical awareness of the interdependence of different professions, academic disciplines, and community interests in relation to the practice of engineering
- gaining experience in self-directed learning and research
- carrying out major engineering projects which require both theoretical and practical competencies

Material to be Covered:

- Software architecture in context
- Architectural analysis
- Conceptual architecture
- Implementation architecture
- Execution architecture
- Architectural styles
- Quality attributes
- Real-time systems
- Web systems
- Industrial systems
- Enterprise systems
- Design and culture

Learning Outcomes:

PROFESSIONAL FORMATION	Learning Outcomes
PF1 <u>Values and social contexts</u> - ability to act responsibly as a professional software system engineer with due regard to the social and environmental contexts of their practice	<ul style="list-style-type: none">• Understanding and ability to express the broader context of software systems development, including stakeholder needs and expectations, and constraints and enablers related to policy, legal constraints, and ethics
PF2 <u>Management skills</u> - ability to exercise sound judgement; and ability to effectively manage resources, critically appraise and work within or challenge constraints and specifications	<ul style="list-style-type: none">• Ability to analyse trade-offs in the architectural design of a software system
PF2 <u>Technical expertise</u> - ability to exercise theoretical and practical competence relevant to the intended field of practice	<ul style="list-style-type: none">• Skills in the analysis of the architectural context of a software system• Skills in the design of a software system architecture• Ability to effectively use a source code control system• Understanding and ability to apply iterative, incremental development
PERSONAL DEVELOPMENT	Learning Outcomes
PD3 <u>Self-fulfilment</u> - ability to identify, take responsibility for, and achieve personal goals.	<ul style="list-style-type: none">• Confidence in their ability to tackle an ill-defined problem, and by using a selection of technical and analytical tools and skills, to successfully complete a solution to that problem
ACADEMIC DEVELOPMENT	Learning Outcomes
AD1 <u>Academic literacy, numeracy, and oral comprehension and presentation skills</u> - ability to engage critically in the academic discourse of one's discipline	<ul style="list-style-type: none">• Knowledge and understanding of the terminology and approaches used in the field of software architecture• Ability to produce relevant, <i>focussed</i>, architectural documentation
AD3 <u>Problem posing and solving</u> - ability to identify, assess, and formulate problems relevant to ones academic discipline, and apply appropriate approaches and methods of problem solving	<ul style="list-style-type: none">• Ability to analyse an ill-defined problem, to identify and evaluate trade-offs, and to make a series of decisions, structured into iterative cycles, that proceed towards an appropriate "solution."

Learning and Teaching Strategies:

Face-to-face contact consists of a 1-hour lecture and a two-hour tutorial or laboratory. Attendance at lectures, tutorials, and laboratories is required. Instructors are available in the LDC to work with teams on their assignments, and additional working sessions may be scheduled by the instructor if required.

The lectures, tutorials, and laboratories are closely integrated. The assignments also tie in closely with the delivery of course materials. Students' understanding of the design, technological and implementation aspects of software architecture are reinforced by the structure of the assignments and the delivery model. That is, the

assignments are delivered in a series of three milestones, each with clearly defined outcomes, that mimic the Conception and Elaboration phases of the Unified Process. Each assignment includes both a written and a programming component, to ensure that the final deliverables are both well-documented and realistic.

Students are supported by an extensive collection of resources (see below). An active online discussion board, which also supports individual blogs (weblogs), encourages students to share in their learning experiences in the subject, and to discuss and debate relevant topics.

General Assessment Overview:

This is a typical example of how this subject will be assessed:

Assessment	Nature of task	Purpose	% weighting
1. Assignments	Three assignments, spaced roughly evenly throughout the semester	Provide a framework within which students develop and demonstrate their knowledge and design skills. Each assignment contains both a written part, culminating in the Software Architecture Document, and a programming part, culminating in the Executable Architectural Prototype. Assignments are done in small teams, but individual contributions in the form of a personal reflection, logbook, and electronic check-ins, are required and monitored.	3 x 20
2. Mid-term exam	A 60- to 90-minute examination.	Encourages students to "catch up" on the delivered course material. Familiarises them with the style of questioning used in the final exam.	10
3. Final exam	A 3-hour formal exam	Asks students to demonstrate the skills and knowledge that they have developed throughout the semester.	30

Resources:

UTSOnline:

Used for distribution of course materials, announcements, and to inform students of their grades.

Softwarepractice.org:

Used for discussion forums, blogs, electronic logbooks, and announcements.

Textbook:

John Reekie and Rohan McAdam. *An Introduction to Software Architecture*. (In preparation).

Written materials:

- 48433 Software Architecture Subject Guide
- 48433 Software Architecture Workbook
- 48433 Software Architecture Reader
- 48433 Software Architecture Slides

Faculty Handbook Text:

Software architecture is the study and design of the high-level structure and behavior of complex software systems. Although it is a relatively new sub-discipline of software engineering, it is also increasingly being seen as vital to the successful construction and deployment of complex software systems. Architecture is often seen as sitting at the interface between requirements (functional, non-functional, and business-related) and the system design itself. It plays a role during the development of a system *and* during its subsequent deployment, operation, maintenance, and termination. As such, it considers not only runtime quality attributes such as performance, reliability, and usability, but also non-runtime quality attributes such as maintainability, testability, and configurability.

This subject introduces the field of software architecture from a pragmatic and practice-based viewpoint. The subject assumes a good grounding in the basics of programming and software construction, and some understanding of the basic principles of software engineering and software lifecycles. In the course of this subject, students will learn to understand and analyse the context within which systems development takes place, to develop and refine multiple views of a software systems architecture, and to describe and implement an architectural prototype using off-the-shelf components, middleware, and custom code.