Software Engineering
SOEN6222
MODULE OUTLINE 2023
(First Edition: 2018)

Table of Contents

Introduction

"Software engineering is an engineering discipline concerned with all aspects of software production from the early stages of system specification to maintaining the system after it has gone into use." (Sommerville, 2016)

You already know how to write software programs in various programming languages for different purposes. But there is much more to successful software development in real-world projects than just writing code. When you work on a larger project with numerous others, the first important aspect to understand is that software engineering has a code of ethics. This helps to build trust within a team since you know what kind of behaviour to expect from your colleagues.

Large projects with strong safety and privacy concerns in industries such as aerospace or healthcare often need very specific development processes. Activities undertaken during the lifecycle of a project include specification, design, implementation (this is where your programming knowledge fits in), validation and software evolution. We will look at these activities in a fair amount of detail. We will also learn how to engineer software for safety and resilience. And we will look at how to reuse software so we do not reinvent the wheel for every new project.

Using this Module Outline

This module outline has been developed to **support your learning**. Please note that the content of this module is on Learn as well as in the prescribed material. You will not succeed in this module if you focus on this document alone.

- This document does not reflect all the content on Learn, the links to different resources, or the specific instructions for the group and individual activities.
- Your lecturer will decide when activities are available/open for submission and when these submissions or contributions are due. Ensure that you take note of announcements made during lectures and/or posted within Learn in this regard.

This Module on Learn

Learn is an online space designed to support and maximise your learning in an active manner. Its main purpose is to **guide and pace** you through the module. In addition to the information provided in this document, you will find the following when you access Learn:

- A list of prescribed materials;
- A variety of additional online resources (articles, videos, audio, interactive graphics, etc.) in each learning unit will further help to explain theoretical concepts;
- Critical questions to guide you through the module's objectives; and
- Collaborative and individual activities (all of which are gradable) with time-on-task estimates to assist you in managing your time around these.

Kindly note:

- Unless you are completing this as a distance module, Learn does **not** replace your contact time with your lecturers and/or tutors.
- SOEN6222 is a Learn module, and as such, you are required to engage extensively with the content on the Learn platform. Effective use of this tool will provide you with opportunities to discuss, debate, and consolidate your understanding of the content presented in this module.
- You are expected to work through the learning units on Learn in your own time – especially before class. Any contact sessions will therefore be used to raise and address any questions or interesting points with your lecturer, and **not** to cover every aspect of this module.
- Your lecturer will communicate **submission dates** for specific activities in class and/or on Learn.

Icons Used in this Document and on Learn

The following icons are used in all your modules on Learn:

Icon	Description
Objectives	A list of what you should be able to do after working through the learning unit.
Prescribed Work	Specific references to sections in the prescribed work.
ThinkAbout	Questions to help you recognise or think about theoretical concepts to be covered.
Active Learning	Sections where you get to grapple with the content/theory. This is mainly presented in the form of questions which focus your attention and are aimed at helping you to understand the content better. You will be presented with online resources to work through (in addition to the textbook or manual references) and find some of the answers to the questions posed.
Connect the dots	Opportunities to make connections between different chunks of theory in the module or to real life.
That is life!	Real life or world of work information or examples of application of theory, using online resources for self-exploration.

REMEMBER:

You need to log onto Learn to:

- Access online resources such as articles, interactive graphics, explanations, video clips, etc. which will assist you in mastering the content; and
- View instructions and submit or post your contributions to individual or group activities which are managed and tracked on Learn.

Module Resources	
Prescribed Material (PM) for this Module	• **PM1: Sommerville, I. 2016. Software Engineering, Global Edition. 10th ed. Boston: Pearson. ISBN: 9781292096131 (Softcover) or ISBN: 9781292096148 (eBook)** • PM2: Dam, R and Siang T. 2020. *10 Insightful Design Thinking Frameworks: A Quick Overview*. [Online]. Available at: https://www.interaction-design.org/literature/article/design-thinking-a-quick-overview [Accessed 30 May 2023]. • PM3: *What is UX Design? Differences Between UX and UI Design* Available at: https://bootcamp.cvn.columbia.edu/blog/what-is-ux-design/ [Accessed 30 May 2023].
Recommended Readings, Digital, and Web Resources	• RR1: Three Reasons Software Engineers Need to Understand UX Design [Online]. Available at: https://www.springboard.com/blog/software-engineering/software-engineering-ux-design/ [Accessed 30 May 2023]. Please note that several additional resources and links to resources are provided throughout this module on the Learn platform. You are encouraged to engage with these as they will assist you in mastering the various objectives of this module. They may also be useful resources for completing any assignments. You will not, however, be assessed under examination conditions on any additional or recommended reading material.
Module Overview	You will find an overview of this module on Learn under the *Module Information* link in the Course Menu.
Assessments	Find more information on this module's assessments in this document and on the Student Portal.

Module Purpose	
The purpose of this module is to teach students the fundamental concepts of software engineering, using various design strategies and testing methods to plan and implement a software engineering process.	
Module Outcomes	
MO1	Develop an awareness of the role and responsibilities of a professional software engineer.
MO2	Analyse software problems and identify solutions using appropriate methods of analysis and design.
MO3	Apply various design strategies to translate a requirements specification into an implementable design.
MO4	Formulate a testing strategy for a software system, employing various testing methods.
MO5	Plan and implement a software engineering process, based on knowledge of widely used development life-cycle models.

Assessments

Integrated Curriculum Engagement (ICE)	
Minimum number of ICE activities to complete	4
Weighting towards the final module mark	10%

Formatives	Part 1	Part 2
Weighting	25%	30%
Duration	10 hours	10 hours
Write/Submit after	LU5	LU7
Learning Units covered	LU 1-4	LU 5-7
Resources required	Prescribed Material	Prescribed Material

Summative	POE
Weighting	35%
Duration	15 Hours
Total marks	100
Open/Closed book	Open
Resources required	Prescribed Material
Learning Units covered	All

Assessment Preparation Guidelines	
Format of the Assessment	Preparation Hints
Part 1	
This part will assess your knowledge of Learning Unit 1 to 4 of this module and will consist of a case study and several simulation questions. You will be expected to apply your skills as per the objectives for these learning units.	• Ensure that you work through all the relevant activities, **exercises,** and revision questions on Learn and in your prescribed material, as well as online sources. • Brainstorm possible questions based on the learning outcomes and objectives provided. • Pay attention to the instructions and to the mark allocations of each question to ensure that you can meet the requirements. •
Part 2	
The part will assess your ability to integrate and apply the content in Learning Units 5-7 of this module to build on the key concepts completed in Part 1.	• Read through the prescribed chapters and content and ensure that you have engaged before you proceed with the part. • Remember to analyse all elements required and ensure that your part meets the requirements. • Improve the quality of your part by using the provided rubric where applicable and addressing any areas of concern prior to submitting it for marking.
Portfolio of Evidence (PoE)	
All learning units will be assessed in the PoE, and reflection on your learning will be included.	• Ensure that you work through all the activities, exercises, and revision questions on Learn and consult your prescribed material. • Include the parts as submitted, together with your lecturer's feedback and your corrected parts based on the feedback received. • Include the reflection of your learning. • Complete other activities included in the PoE.

Module Pacer			
Code	Programme	Contact Sessions	Credits
SOEN6222	BIS3; BCA2; BCIS2	48	15
Learning Unit 1	Professional Software Development and Ethics		

Overview:

In this learning unit, you will explore what it means to be a professional software engineer. This discussion includes the codes of ethics followed by software engineers. The case studies that are used throughout the textbook will also be introduced.

Learning Unit 1: Theme Breakdown		
Sessions: 1 – 3	Theme 1: Professional Software Development	Prescribed Material (PM)
Academic Week: 1	LO1: Explain why software projects fail. LO2: Differentiate between the two kinds of software products. LO3: Define software engineering. LO4: Differentiate between the types of software applications.	• PM1: Chapter 1
Related Outcomes: MO1	Theme 2: Software Engineering Ethics	
	LO5: Identify the professional responsibilities of software engineers. LO6: Identify the eight principles that software engineers should adhere to.	
	Theme 3: Case Studies	
	LO7: Identify critical system requirements for different types of systems.	

Learning Unit 2	Software Processes
Overview: A software process is a set of activities to produce software. In this learning unit, you will learn about the different software processes and models. You will also see how software processes can accommodate change. You will also explore how software processes can be improved over time.	

Learning Unit 2: Theme Breakdown		
Sessions: 4 – 7	Theme 1: Software Process Models	Prescribed Material (PM)
Academic Week: 1 – 2	LO1: Differentiate between the three general process models.	• PM: Chapter 2
Related Outcomes: MO5	Theme 2: Process Activities	
	LO2: Define software specification. LO3: Define software design. LO4: Define software validation. LO5: Explain how software evolves.	
	Theme 3: Coping with Change	
	LO6: Contrast prototyping and incremental delivery as mechanisms to cope with change.	
	Theme 4: Process Improvement	
	LO7: Compare the two approaches to process improvement. LO8: Identity the levels in the process maturity model.	

Learning Unit 3	Agile Software Development
Overview: In this learning unit, you will learn about the importance of agile development practises such as user stories and refactoring, as well as gain a deeper understanding of the scrum approach to agile project management. At the end of this unit, you should have a clear understanding of best practices to be used in agile development.	

Learning Unit 3: Theme Breakdown		
Sessions: 7-10	Themes	Prescribed Material (PM)
Academic Week: 1 – 2 Related Outcomes: MO5	Theme 1: Agile development techniques	• Chapter 3
	LO1: Explain the rationale for agile software development. LO2: Discuss the principle of agile methods. LO3: Explain various Agile development techniques. LO4: Discuss the benefits of user stories.	
	Theme 2: Scrum	
	LO5: Apply the scrum approach to project management.	

Learning Unit 4	Requirements Engineering
Overview: In this learning unit, you will learn what user and system requirements are. You will see what the difference is between functional and non-functional requirements, and you will be introduced to how requirements are gathered. In the last theme, we will focus on the stages in the change management process.	

Learning Unit 4: Theme Breakdown		
Sessions: 8 – 13	Theme 1: Functional and Non-functional Requirements	Prescribed Material (PM)
Academic Week: 2 – 4	LO1: Differentiate between user and system requirements. LO2: Differentiate between functional and non-functional requirements.	• PM: Chapter 4
Related Outcomes: MO2	Theme 2: Requirements Engineering Process	
	LO3: Identify the main activities in the requirements engineering process.	
	Theme 3: Requirements Elicitation	
	LO4: Identify the activities to be performed during requirements elicitation. LO5: Identify the techniques used for requirements elicitation.	
	Theme 4: Requirements Specification	
	LO6: Contrast the different types of specifications.	
	Theme 5: Requirements Validation	
	LO7: Explain how requirements can be validated.	
	Theme 6: Requirements Change	
	LO8: Apply the stages of the change management process.	

Learning Unit 5	System Modelling
Overview: System modelling is the process of creating abstract models of a system. In this learning unit, you will learn about the different types of models and how to draw them using Unified Modelling Language (UML). Finally, you will be introduced to model-driven architecture.	

Learning Unit 5: Theme Breakdown		
Sessions: 14 – 18	Theme 1: Behavioural Models	Prescribed Material (PM)
	LO1: Differentiate between data-driven and event-driven modelling. LO2: Differentiate between model-driven engineering and model-driven architecture.	• Chapter 5
	Theme 2: Model-driven Architecture	
	LO3: Identify the types of system models recommended by model-driven architecture.	

Learning Unit 6	Architecture, Design and Implementation
Overview: In this learning unit, you will learn why architectural design is important and will get to know some architectural patterns. You will learn about design patterns and consider issues around software re-use and open-source development.	

Learning Unit 6: Theme Breakdown		
Sessions: 19 – 24	Theme 1: Architectural Design Decisions	Prescribed Material (PM)
Academic Week: 6	LO1: Define architectural design.	• PM: Chapter 6
Related Outcomes: MO2 MO3	Theme 2: Architectural Views	
	LO2: Identify the four fundamental architectural views.	
	Theme 3: Architectural Patterns Analysis	
	LO3: Differentiate between the various architectural patterns.	
	Theme 4: Application Architectures	
	LO4: Explain how the models of application architecture can be applied.	• PM: Chapter 7
	Theme 5: Design Patterns	
	LO5: Explain the purpose of design patterns. LO6: Describe the elements of a design pattern.	
	Theme 6: Implementation Issues	
	LO7: Explain common implementation issues.	

Learning Unit 7	UI/UX Design

Overview:
This learning unit introduces you to the fundamentals of User Experience Design. We will begin by discussing the difference and the relationship between User Interface (UI) and User Experience (UX) Design and will explore the dynamics between them. In the last theme, you will be introduced to the user-centred design process, and we will briefly examine the purpose and activities of each phase in the process.

Learning Unit 7: Theme Breakdown		
Sessions: 25-32	Theme 1: What is UX Design and UI Design?	Prescribed Material (PM)
Academic Week: 7	LO1: Differentiate between User Experience (UX) Design and User Interface (UI) Design. LO2: Explain the dynamic between UX and UI Design. LO3: Describe why software engineers need to collaborate with UX designers.	• PM2 and PM3.
Related Outcomes: MO001 MO002		

Learning Unit 8	Software Testing and Quality Management
Overview: This learning unit introduces testing during the various stages of the software lifecycle. Formalised quality management is discussed, including how quality management works in agile methods.	

Learning Unit 8: Theme Breakdown		
Sessions: 33-40	Theme 1: Development Testing	Prescribed Material (PM)
Academic Week: 8	LO1: Contrast validation and verification. LO2: Differentiate between unit testing, component testing and system testing.	• PM1: Chapter 8
Related Outcomes: MO4 MO5	Theme 2: Test-driven Development	
	LO3: Explain the test-driven development process.	
	Theme 3: Release Testing	
	LO4: Differentiate between the different types of release testing.	
	Theme 4: User Testing	
	LO5: Identify the different types of user testing.	
	Theme 5: Software Quality	• PM1: Chapter 24
	LO6: Define software quality management.	
	Theme 6: Software Standards	
	LO7: Identify the types of software standards.	
	Theme 7: Reviews and Inspections	
	LO8: Explain the software review process.	

Learning Unit 9	Software Evolution
Overview: Software systems must evolve to remain relevant. In this learning unit, you will learn how software evolves, why legacy systems are important, and how legacy systems are maintained.	

Learning Unit 9: Theme Breakdown		
Sessions: 41-48	Theme 1: Evolution Processes	Prescribed Material (PM)
Academic Week: 9	LO1: Explain the steps involved in changing software.	• PM1: Chapter 9
Related Outcomes: MO05	Theme 2: Legacy Systems	
	LO2: Describe the maintenance of legacy systems.	
	Theme 3: Software Maintenance	
	LO3: Explain how software maintenance can be predicted. LO4: Choose between reengineering, replacement and refactoring for a given software application.	

Glossary of Key Terms for this Module

Term	Definition
Agile Methods	"Agile methods are incremental development methods in which the increments are small, and typically, new releases of the system are created and made available to customers ever two or three weeks." (Sommerville, 2016: 74)
Application Framework	"[A] framework is a generic structure that is extended to create a more specific subsystem or application." (Sommerville, 2016: 443)
Architectural Description Language (ADL)	"[S]pecialised architectural description languages (ADLs) [are used to] describe system architectures. The basic elements of ADLs are components and connectors, and they include rules and guidelines for well-formed architectures." (Sommerville, 2016: 175)
Architectural Design	"[The activity] where you identify the overall structure of the system, the principal components (sometimes called subsystems or modules), their relationships, and how they are distributed." (Sommerville, 2016: 57)
Design Pattern	"A well-tried solution to a common problem that captures experience and good practice in a form that can be reused. It is an abstract representation that can be instantiated in several ways." (Sommerville, 2016: .762)
Embedded System	"A software system that is embedded in a hardware device, i.e. the software system in a cell phone. Embedded systems are usually real-time systems and so must respond in a timely way to events occurring in an environment." (Sommerville, 2016: 762)
Hazard	"A condition or state in a system that has the potential to cause or contribute to an accident." (Sommerville, 2016: 764)
Incremental Development	"This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version." (Sommerville, 2016: 46)
Legacy System	"A socio-technical system that is useful or essential to an organisation but has been developed using obsolete technology or methods. Because legacy systems often perform critical business functions, they must be maintained." (Sommerville, 2016: 765)

Term	Definition
Model-driven Architecture (MDA)	"An approach to software development based on the construction of a set of system models, which can be automatically or semi-automatically processed to generate an executable system." (Sommerville, 2016: 765)
Open-Source Software	"An approach to software development where the source code for a system is made public and external users are encouraged to participate in the development." (Sommerville, 2016: 766)
Performance Test	"Performance tests have to be designed to ensure that the system can process its intended load." (Sommerville. 2016: 248)
Process Improvement	"Changing a software development process with the aim of making that process more efficient or improving the quality of its outputs. For example, if your aim is to reduce the number of defects in the delivered software, you might improve a process by adding new validation activities." (Sommerville. 2016: 767)
Refactoring	"Modifying a program to improve its structure and readability without changing its functionality." (Sommerville, 2016: 769)
Resilience	"A judgment of how well a system can maintain the continuity of its critical services in the presence of disruptive events, such as an equipment failure and cyberattacks." (Sommerville, 2016: 769)
Safety-critical System	"A system whose failure can result insignificant economic, human or environmental losses." (Sommerville. 2016: 761)
Software Analytics	"Automated analysis of static and dynamic data about software systems to discover relationships between these data. These relationships may provide insights about possible ways to improve the quality of the software." (Sommerville, 2016: 771)
Software as a Service (SaaS)	"Software applications that are accessed remotely through a web browser rather than installed on local computers. Increasingly used to deliver application services to end-users." (Sommerville, 2016: 771)
Software Design	
Software Metric	"An attribute of a software system or process that can be expressed numerically and measured. Process metrics are attributes of the process such as the time taken to complete a task; product metrics are attributes of the software itself such as size or complexity." (Sommerville, 2016: 771)
Software Process	"The activities and processes that are involved in developing and evolving a software system". (Sommerville, 2016: 772)

Term	Definition
System Testing	“The testing of a completed system before it is delivered to customers.” (Sommerville, 2016: 773)
Test-driven Development (TDD)	“An approach to software development where executable tests is written before the program code. The set of tests are run automatically after every change to the program.” (Sommerville, 2016: 773)
Use Case	“A specification of one type of interaction with a system.” (Sommerville, 2016: 773)
Unified Modelling Language (UML)	“A graphical language used in object-oriented development that includes serval types of system model that provide different views of a system. The UML has become a *de facto* standard for object-oriented modelling.” (Sommerville, 2016: 773)
Unit Testing	“The testing of individual program units by the software developer or development team.” (Sommerville, 2016: 773)
User Testing	“User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.” (Sommerville, 2016: 49)
Validation	“The process of checking that a system meets the needs and expectations of the customer.” (Sommerville, 2016: 774)
Verification	“The process of checking that a system meets its specification.” (Sommerville, 2016: 774)
Waterfall Model	“This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design implementation and testing.” (Sommerville, 2016: 45)