Unit 7: Software Development Lifecycles

Unit code	**K/618/7408**
Unit level	**4**
Credit value	**15**

Introduction

The software development lifecycle is an integrated process that promotes building good quality, secure software throughout the entire development process. The aim of this unit is to give students the knowledge and skills needed to understand software development lifecycles so that they can demonstrate their knowledge by implementing a software development lifecycle with a suitable methodology.

The unit introduces students to lifecycle decision making at different stages of the software development process. They will examine various lifecycle models and learn to appreciate their particular characteristics in order to understand for which project environments they are most appropriate. Theoretical understanding will be translated into practical skills through an actual software development lifecycle project. Students will become confident in the use of particular tools and techniques relevant to a chosen methodology.

Among the topics included in this unit are iterative and sequential models of software development lifecycles and reference frameworks for initially capturing conceptual data and information through a feasibility study, and requirement gathering techniques through to analysis, design and software implementation activities.

Students will develop skills such as communication literacy, critical thinking, analysis, reasoning and interpretation, which are crucial for gaining employment and developing academic competence.

Learning Outcomes

By the end of this unit students will be able to:

LO1 Describe different software development lifecycles

LO2 Explain the importance of a feasibility study

LO3 Undertake a software development lifecycle

LO4 Discuss the suitability of software behavioural design techniques.

Essential Content

LO1 **Describe different software development lifecycles.**

Software development lifecycles:

Describe different software development lifecycles.

Understand and use different lifecycle models, including predictive (Waterfall, Prototyping, RAD), adaptive (Spiral, Agile, DSDM), sequential and iterative software development models.

Lifecycle stage and connectivity, including feasibility study, analysis, design, implementation, testing, review or analysis, design, implementation, maintenance, planning, requirements traceability.

Testing and integration, including relationship between test activities and software development activities, levels of testing, building test environments, developing test harnesses, black box and white box testing, incremental testing, system testing, acceptance test and integration approaches, changeover strategies, trials and Go-Live prerequisites.

Understand the role and utilisation of analysis artefacts:

The creation of analysis artefacts in a software development project, e.g. software requirements specification, use case or user stories, user profiles, workflow model, wireframes, logical data model, data dictionary etc.

The purpose and activities of the gap analysis process.

Roles and responsibilities in a large-scale software project development lifecycle:

Identify the different individuals in a project, e.g. project manager, business analyst, systems analyst, programmer, DevOps engineer, testing engineer etc.

Contributions, including quality assurance, common core skills, tools and behaviours.

Explore how the psychology and mindset of testing differs to that development mindset and their possible influence on the overall success of a software project.

LO2 **Explain the importance of a feasibility study**

Requirement gathering:

Requirement gathering techniques, including how to categorise, validate and prioritise, e.g. MosCow method, functional requirements, non-functional requirements, users and constraints.

Interviews, observation, investigation.

Importance of feasibility study:

Feasibility criteria considerations, e.g. legal, social, economic, technical, timescales, organisational constraints.

Components of feasibility study, including purpose, structure, intended audience, outcomes.

The purpose of process modelling and the importance of an organisational view of business processes.

Key drivers for change, including performance and efficiency, legacy systems upgrade, automation, elimination of human error.

LO3 **Undertake a software development lifecycle**

Carry out software development lifecycle:

Follow company, team or client approaches to continuous integration, version and source control.

Apply an appropriate software development approach according to the relevant paradigm, e.g. object oriented, event driven or procedural.

Identify stakeholder requirements.

Scope of project, including inputs, outputs, processes and process descriptors, consideration of alternate solutions and security considerations, required quality assurance and testing.

Constraints specific to activity, e.g. costs, organisational policies, legacy systems, hardware requirements.

Create simple software designs to effectively communicate understanding of the program.

Follow agreed software designs and technical and functional specifications.

Follow organisational policies and procedures relating to the tasks being undertaken, e.g. the storage and treatment of GDPR sensitive data.

Report documentation, including structure, e.g. background information, problem statements, data collection process and summary, recommendations and appendices.

Use of appropriate systems analysis terminology and tools, including data stores and entities, data flows, process representation techniques relationships (1:1, 1:M and M:M).

Investigation, e.g. upgrading computer systems, designing new systems.

Techniques and documents for documenting business requirements and processes relevant to selected methodology, e.g. Context Diagrams, Data Flow Diagrams (DFDs), Entity Relationship Diagrams (ERDs), Business Systems Options (BSOs), Technical Systems Options (TSOs) and requirements traceability.

Analyse documented requirements to remove duplication, conflict and overlap.

Quality considerations, e.g. Total Quality Management (TQM).

LO4 **Discuss the suitability of software behavioural design techniques**

Evaluate suitability of software behavioural design techniques:

Flowcharts, pseudocode, formal specification methods, event/state/data driven, finite state machines extended-FSM/FSP.

Problem of e-FSM state explosion, reachability analysis, safety, liveness properties.

Automatic analysis and animation tools.

Understand the characteristics of software architecture that impact on software testing in the development lifecycle.

Learning Outcomes and Assessment Criteria

Pass	Merit	Distinction
LO1 Describe different software development lifecycles		**D1** Assess the merits of applying the Waterfall lifecycle model to a large software development project.
P1 Describe two iterative and two sequential software lifecycle models. **P2** Explain how risk is managed in software lifecycle models.	**M1** Discuss using an example, why a particular lifecycle model is selected for a development environment.	
LO2 Explain the importance of a feasibility study		**D2** Assess the impact of different feasibility criteria on a software investigation.
P3 Explain the purpose of a feasibility report. **P4** Describe how technical solutions can be compared.	**M2** Discuss the components of a feasibility report.	
LO3 Undertake a software development lifecycle		**D3** Evaluate the process of undertaking a systems investigation with regard to its effectiveness in improving a software quality.
P5 Undertake a software investigation to meet a business need. **P6** Use appropriate software analysis tools/techniques to carry out a software investigation and create supporting documentation.	**M3** Analyse how software requirements can be traced throughout the software lifecycle. **M4** Discuss two approaches to improving software quality.	
LO4 Discuss the suitability of software behavioural design techniques		**D4** Present justifications of how data-driven software can improve the reliability and effectiveness of software.
P7 Discuss, using examples, the suitability of software behavioural design techniques.	**M5** Analyse a range of software behavioural tools and techniques. **M6** Differentiate between a finite state machine (FSM) and an extended FSM, providing an application of use for both.	

Recommended Resources

Textbooks

Dennis, A. and Haley, W. (2009) *Systems Analysis and Design*. John Wiley & Sons Ltd.

Lejk, M. and Deeks, D. (2002) *An Introduction to System Analysis Techniques*. 2nd Ed. Addison-Wesley.

Murch, R. (2012) *The Software Development Lifecycle: A Complete Guide*. Kindle.

Smart, J. F. (2014) *BDD in Action: Behavior-driven development for the whole software lifecycle*. Manning.

Web

www.freetutes.com	FreeTutes *Systems Analysis and Design – Complete Introductory Tutorial for Software Engineering* (Tutorial)
www.ijcsi.org	*IJCSI International Journal of Computer Science* Vol. 7, Issue 5, September 2010 *A Comparison Between Five Models Of Software Engineering* (Research)
www.ijcsi.org	*IJCSI International Journal of Computer Science* Vol. 6, Issue 1, 2015 *Software Development Life Cycle Models – Comparison, Consequences (Research)*

Links

This unit links to the following related units:

Unit 6: Planning a Computing Project

Unit 16: Computing Research Project

Unit 22: Application Development

Unit 35: Systems Analysis & Design

Unit 42: Game Design Theory

Unit 43: Games Development

Unit 54: Prototyping.