Module: Software Analysis & Design 281

Module name:	Software Analysis & Design 281
Code:	SAD281
NQF level:	7
Type:	Fundamental – Bachelor of Computing (Software Engineering stream)
Contact Time:	38 hours
Structured time:	6 hours
Self-directed time:	46 hours
Notional hours:	90 hours
Credits:	9
Prerequisites:	PMM281

Purpose

This module covers the processes, methods, techniques and tools that organisations use to determine how they should conduct their business, with a particular focus on how computer-based technologies can most effectively contribute to the way business is organised. It covers a systematic methodology for analysing a business problem or opportunity, determining what role computer-based technologies play in addressing the business need, specifying alternative approaches to acquiring the technology capabilities needed to address the business requirements and the requirements for the information systems solution.

Outcomes

Upon successful completion of this module, the student will be able to:

- Demonstrate integrated knowledge of the types of business needs that can be addressed using information technology-based solutions.
- Understand a range of methods of enquiry in the discipline and their suitability to be applied in specific investigation to resolve problems related to the context of methodologies, writing of clear, concise business requirements documents and converting them into technical specifications.
- Identify, analyse, evaluate, critically reflect on and address complex problems, applying evidence-based solutions and theory-driven arguments in designing of high-level logical system characteristics (user interface design, design of data and information requirements).
- Develop and communicate their ideas and opinions in well-formed arguments, using appropriate academic discourse.
- Take decisions and articulate ethical, cultural, and legal issues and their feasibilities among alternative solutions.
- Present and communicate complex information, evidence-based solutions and theory-driven arguments reliably and coherently using appropriate academic conventions, formats and technologies for a given context.

Assessment

- Continuous evaluation of theoretical work through two written assignments, one formative, and one summative test.
- Continuous evaluation of project work, whereby the student must analyse, recommend, redesign and report on the outcome for a given scenario.
- Final assessment through a written examination.

Teaching and Learning

Learning materials

- Software Analysis & Design (2017) – IT without frontiers.
- Presentation notes, lecturer hand-outs, samples and lab exercises.

Additional Reference Material:

- 🕮 Dennis, A., Wixom, B. H., and Tegarden, D. (2005): *Systems Analysis and Design with UML Version 2.0*, 2nd Edition. John Wiley & Sons [ISBN 0-471-34806-6]
- 🕮 Uml.org. (2018). Welcome To UML Web Site! [online] Available at: http://uml.org/ [Accessed 13 Jun. 2018].

Learning activities

The teaching approach consists of a combination of formal lectures on theoretical and practical concepts, solving real-world problems through exercises, demonstrations of feasible solution in a specific context and discussions of high-level design specifications. It is dialogue-oriented with a practical approach with mandatory assignments, projects, and written examinations, formative and summative assessments that must be completed during the module.

Notional learning hours

Activity	Units	Contact Time	Structured Time	Self-Directed Time
Lecture		27.0		14.0
Formative feedback		7.5		
Project	1	3.5		9.0
Assignment	2			6.0
Test	2		4.0	8.0
Exam	1		2.0	9.0
		38.0	**6.0**	**46.0**

Syllabus

- Identification of opportunities for IT-enabled organisational change
- Structuring of IT-based opportunities into projects
- Fundamentals of IS project management
- Analysis and specification of system requirements

- Different approaches to systems analysis & design: structured SDLC, unified process/UML, agile methods
- Different approaches to implementing information systems to support business Requirements
- Impact of implementation alternatives on system requirements specification