Level 6 Advanced Diploma in Computer Science (907)
203 Credits

Unit: Software Engineering **Exam Paper No.:** 5	**Total Qualification Time:** 240 **Number of Credits:** 24
Prerequisites: Good knowledge in System Analysis and Design.	**Corequisites:** A pass or better in Diploma in System Design or equivalence.
Aim: The principal objective of this unit is produce learners well versed in the principles of design, measurement and analysis, applied in the context of the development of software-based systems. A process used for building software is just like processes used for building any custom made product, hence a software engineer needs to follow a defined engineering process to build software, just like an architect who constructs a building following a defined process. Various system development models, including current software engineering theory and practice, methodologies, techniques, CASE tools are covered. Learners also receive a solid foundation in technical communication skills, professional responsibility, social-effects, ethical aspects of software engineering practice, interpersonal relationships, teamwork and time management. The unit looks at software engineering methods and tools for application development, including design and system organisation; using and creating reusable libraries; building, testing, and debugging; and performance evaluation. Overview of computer organization, algorithm design, introduction to programming in C++, input / output statements, arithmetic expressions, assignment statements, logical expressions, conditional statements, control statements, functions and function calls, math library, i/o library, character library, introduction to arrays and pointers, program testing and debugging, built in and user defined data types, arrays, lists, strings, arrays, records, classes and data abstraction, C++ object oriented software developments, inheritance, composition, dynamic binding and virtual functions, pointers, dynamic data, reference data types, recursion; software processes, software requirements engineering, system models, architectural design, object-oriented design, software reuse, verification and validation, software testing, software cost, quality management and process improvement are also covered.	
Required Materials: Recommended learning resources.	**Supplementary Materials:** Lecture notes and tutor extra reading recommendations.
Special Requirements: Thorough research on Software Engineering and implementing practical applications using an object oriented programming language.	

Intended Learning Outcomes:	**Assessment criteria:**
1 Software engineering, the historical origins and many activities of software engineering in addition to programming.	1.1 Analyse software engineering trend 1.2 Define software engineering 1.3 Define reasons why software engineering was born and the issues it addresses. 1.4 Describe relationship between software engineering and systems engineering. 1.5 Describe relationship between software engineering and programming. 1.6 Define main activities in software engineering. 1.7 Describe the meaning and importance of the software engineering process.
2 Software; its nature, qualities and analyse the classification of software qualities.	2.1 Analyse reasons for the human-intensiveness of software engineering. 2.2 Describe the meaning of functional requirements, the definition and kinds of qualities required from software development 2.3 Differentiate between external and internal software qualities

Business & Computing Examinations (BCE)

	2.4 Identify examples of external and internal software qualities 2.5 Identify examples of qualities required of software processes
3 Software engineering principles, the requirement process; the role of methodologies that package tools and techniques to encourage a particular approach to software development.	3.1 Define the important software engineering principles that form the basis of methods, techniques, methodologies and tools 3.2 Describe the definitions and important principles used in phases of software development and in designing software processes 3.3 Describe the importance of modularity as the cornerstone principle supporting software design. 3.4 Describe factors which leads to project failures 3.5 Define the process of capturing user requirements 3.6 Describe modelling notations 3.7 Describe the requirements and specification languages 3.8 Define prototyping
4 The importance of design, the architecture of a software system in terms of its components; the term "specification", the role and importance of specification in the different phases of software engineering.	4.1 Define program design. 4.2 Describe the issues, techniques and characteristics of program design 4.3 Define conceptual and technical design 4.4 Describe the different design styles, techniques and tools 4.5 Describe good design characteristics 4.6 Define a project schedule 4.7 Devise means of understanding customer needs 4.8 Define roles played by different personnel 4.9 Define the different types of costs involved 4.10 Illustrate risk management activities 4.11 Define how to track project progress 4.12 Describe project personnel. 4.13 Define risks management and illustrate the importance of a project plan.
5 The goals of verification, the main approaches to verification; the differences between formal and informal analysis and when to use each.	5.1 Describe testing requirements for concurrent and real-time systems. 5.2 Describe how to apply testing principles for object-oriented software. 5.3 Analyse the use of both formal and informal analysis techniques. 5.4 Describe the basic techniques of code inspections and walkthroughs. 5.5 Describe how to approach debugging systematically. 5.6 Describe the problems involved in verifying software qualities other than functional correctness and performance. 5.7 Analyse some of the metrics used to measure complexity, reliability, and

Business & Computing Examinations (BCE)

	performance. 5.8 Explain why traditional statistical models that are effective for measuring reliability in traditional engineering fields are difficult to apply to software. 5.9 Describe program testing process. 5.10 Analyse the objective of software testing. 5.11 Describe why software fail 5.12 Describe the different types of software faults 5.13 Discuss who should perform software tests
6 The phases of the traditional software life cycle and the goals of software processes.	6.1 Define software process model tools and techniques for process modelling. 6.2 Analyse software development products, processes and resources 6.3 Identify the several models of the software development process 6.4 Describe the characteristics and limitations of the waterfall software process model. 6.5 Explain how to apply well-known methodologies: structured-analysis/structured-design (SA/SD) and Jackson system development method (JSD). 6.6 Describe the basic principles and phases of the unified process. 6.7 Analyse the importance and role of configuration management in the software life cycle. 6.8 Define the problems of legacy software; the processes and tools that focus on the activities in the maintenance of legacy software.
7 Problems encountered in managing software engineering projects, the key tasks of a project manager; challenges they face, how productivity can be measured and the tools used for planning and monitoring.	7.1 Describe programming standards, procedures and guidelines 7.2 Define control structures 7.3 Evaluate the use of algorithms and data structures 7.4 Evaluate the importance of program re-use 7.5 Describe the problems inherent in organising, controlling, and measuring intellectual activities. 7.6 Describe the common methods for measuring software productivity (lines of code and function points), and their limitations. 7.7 Describe the tools that managers use to plan and monitor projects. 7.8 Describe how to apply Work Breakdown Structures, GANTT and PERT charts in project management. 7.9 Describe typical and effective structures for organising members of a team; their limitations and strengths. 7.10 Describe the capability maturity model

Business & Computing Examinations (BCE)

	for measuring the effectiveness of software organisations.
8 The role and uses of UML design tool and UML CASE tools in software engineering.	8.1 Define objects and describe what Object-Orientation (OO) is 8.2 Define OO characteristics 8.3 Define objects and classes 8.4 Describe the OO development process 8.5 Define CASES 8.6 Define the role of editors, linkers, generators, interpreters, debuggers, analysers, tracking tools, reverse engineering tools, and management tools in different phases of the software lifecycle. 8.7 Analyse the reasons and directions for the evolution of CASE tools.
9 The impacts of software engineering on society and ethical issues raised by software engineering.	9.1 Describe the principles of ethics adopted to guide software engineers in making ethical decisions 9.2 Analyse the influence of the Internet on software engineering 9.3 Describe the role of software as an enabling technology 9.4 Describe how the Internet is providing new possibilities for software engineering.
Methods of Evaluation: A 3-hour written essay examination paper with 5 questions, each carrying 20 marks. Candidates are required to answer all questions. Candidates also undertake project/coursework in Software Engineering with a weighting of 100%.	

Recommended Learning Resources: Software Engineering

Text Books	• Software Engineering: International Edition, 3/E by Shari Lawrence Pfleeger Joanne M Atlee ISBN-10: 0131984616 • Software Engineering: (Update), 8/E Ian Sommerville, *University of St. Andrews, United Kingdom* ISBN-10: 0321313798
Study Manuals 	BCE produced study packs
CD ROM 	Power-point slides
Software 	None

Business & Computing Examinations (BCE)