Professional Graduate Diploma in IT Syllabus

Software Engineering 2

Rationale:

In order to develop, maintain in operation, and evolve software systems that are of high quality it is imperative the Computing professionals have an understanding of software development and its evolution as an engineering discipline. This understanding must be based on the theoretical foundations of software engineering and demonstrated through a critical application of software engineering theory to real world problems in practical applications.

Aims:

To gain a thorough understanding of the relationship between the processes used in the engineering of software systems, the software products produced, and of the theory, laws, and models, that provide a rational basis for the practice of software engineering.

Objectives:

To demonstrate a critical understanding of software systems engineering theory in the form of laws and models, and of associated methods, tools, and techniques;

To critically apply the above to practical situations found throughout the Software Life Cycle from software requirements engineering, system specification, design, implementation, validation and verification, through to maintenance and evolution; and to recognise the potential for software reuse throughout the life cycle;

To appreciate the importance of software project management, software economics, estimation and planning as well as the management of software project teams and their productivity;

To appreciate the role of empirical studies in Software Engineering and the role of the Goal/Question/Metric in software engineering experiments, especially in the context of software quality improvement experiments with respect to both process and product improvements;

To discuss critically recent advances in software engineering: component-based software engineering, model driven software development, the Agile paradigm including Extreme programming, software product lines engineering and community based software development such as Free and Open Source Software development.

To appraise advanced software concepts and their applicability in practice: Design Patterns and Frameworks, Software Refactoring techniques, Software Architectural analysis, Software as a Service.
Prior Knowledge Expected:

Candidates should be familiar with the material covered in the Certificate syllabus and the Software Engineering 1 Diploma syllabus and should have knowledge of Object Oriented software development, both design and implementation and practical experience of developing software systems.

Content:

Advanced topics of software engineering in two main areas: the analysis and improvement of software processes and the analysis and improvement of software products. Indicative process and product topics include the following:


- Various Software Life Cycle Models (Waterfall, V-model, Prototyping, Spiral Model, Incremental Development, Evolutionary Development, Agile models including Extreme programming)

- Software Requirements Engineering including requirements management

- Software Management: project management, cost estimation, planning, personnel management, team building

- Software Evolution: Lehman's Laws of Software Evolution and related models and studies

- Software Maintenance and the related types of maintenance, Impact Analysis, Reverse and Re-engineering of software

- Software reuse, Component based software engineering, Software product lines, Design patterns,

- Software Architecture and software re-factoring, Architectural styles, examples, and applications, Architectural models, Model-driven development

- Software as a service, including web services and dynamic reconfiguration of software systems

- Open Source Software Engineering

- Advanced use of UML including Object-Constraint Language and use of Assertions, pre- and post-conditions

- Software metrics, software complexity measures, measures of software coupling and cohesion, models and associated measures of software quality.
Empirical Software Engineering and Basili’s Goal/Question/Metric

Primary Texts


Other texts


Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, ISBN 0-201-63361-2.


