Philadelphia University

Faculty of Information Technology

Department of Software Engineering

Course Catalogue

2016-2017

CONTENT

CHAPTER 1

INTRODUCTION TO CURRICULUM DESIGN

This catalogue contains a set of module descriptions and some information on the curriculum design and organisation that mostly follow the Accreditation Standards sets by the Higher Education Accreditation Commission (Jordan). It also follows the recommendations of the Software Engineering Curricula 2014 (SE2014). The SE2014 is a joint undertaking of the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-SE) and the Association for Computing Machinery (ACM) that developed curricular guidelines for undergraduate programs in computing.
These modules are offered at the Department of Software Engineering, Faculty of Information Technology/ Philadelphia University, to obtain the four years B.Sc. (honour) degree in Software Engineering (SE).
The information given in this catalogue is extracted for the Program Specifications for the Degree programme. These specifications are published separately.

1.1 Fundamental Concepts

The most important concepts for understanding the module descriptions are as follows:

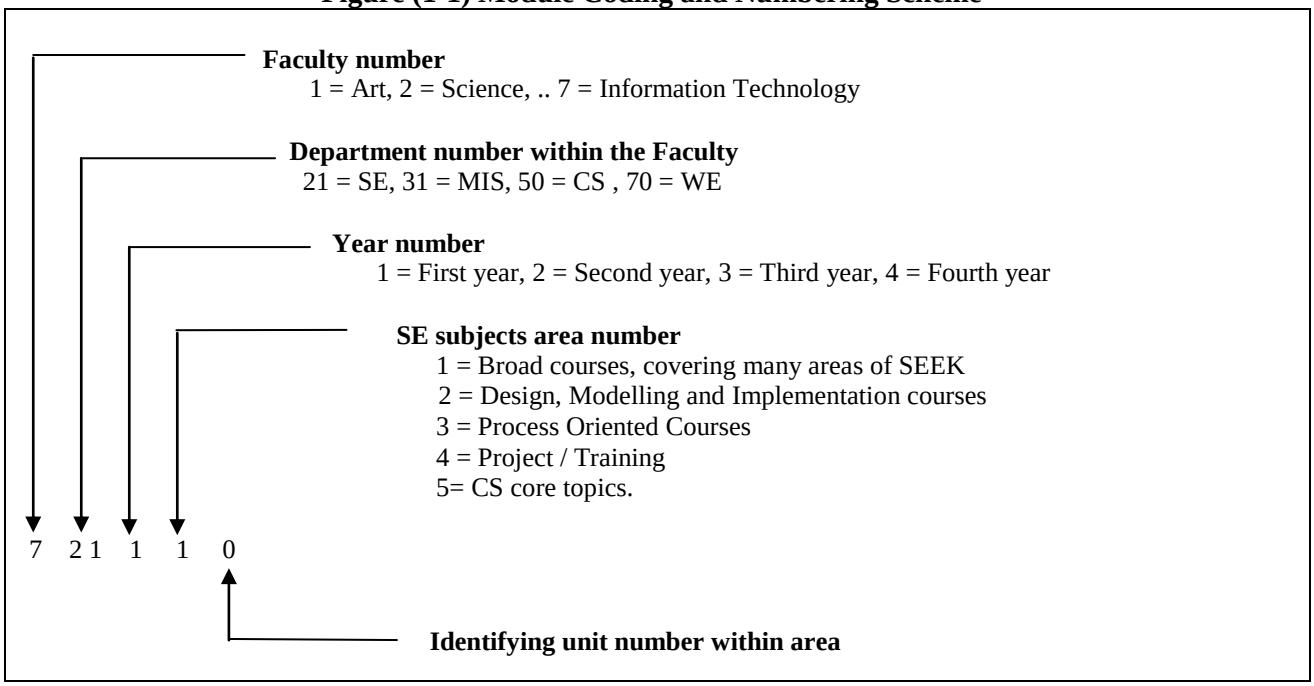
- **The SE Body of Knowledge.** The modules described in this Catalogue are defined in relation to a general taxonomy of that portion of Software Engineering appropriate for an undergraduate curriculum. That taxonomy represents the body of knowledge for Software Engineering. The body of knowledge is organised hierarchically into three levels. The highest level of the hierarchy is the **area**, which represents a particular disciplinary sub-field. The areas are broken down into smaller divisions called **units**, which represent individual thematic modules within an area. Each unit is further subdivided into a set of **topics**, which are the lowest level of the hierarchy.
- **Core and Elective Units**. Given the expanding scope of the computing discipline, it is impossible to insist that every undergraduate learn all the topics that were at one time considered fundamental to the field. The SE2014 report defines a minimal set of core units for which there is a broad consensus that the material is essential to anyone obtaining an undergraduate degree in computer science. Because the core is defined as minimal, the core alone cannot constitute a complete undergraduate curriculum. The undergraduate program must include additional elective units from the body of knowledge. These elective units could be chosen according to the needs of the individual student. Note that, occasionally, timetabling difficulties restricts elective units.

- **Credit Hours.** To give a sense of the time required to cover a particular unit, a time metric should be chosen. The system of study at Philadelphia University is based on the credit hours. The basic measure unit of the curriculum is 3 credit hours module (or course unit). A module, which delivers at least 3 hours per week of lectures or tutorial time, is worth 3 credit hours. Some modules may also provide an extra 1-hour per week for laboratory, but the module is still classified as 3 credit hours. In general, over a 16 weeks semester, a typical module provides minimum 45 hours of contact time. The final week of the semester is used for the examinations. The contact time corresponds to the in-class time required to present the material in a traditional lecture oriented format. Note that this time does not include the instructor's preparation time or the time students spend outside of class. As a general guideline, the time required outside of class is twice the time of the in-class time. Thus, a unit that is listed as requiring 3 credit hours will typically entails a total of 9 hours (3 in class and 6 outside). It is also important to keep in mind that the time associated with each unit represents the minimum number of hours required for adequate coverage, and that it is always appropriate to spend more time than the listed minimum.

1.2 Format of the Module Coding Adopted

Each module in the SE programme is identified by a code and a title. For example, **"721110 - Introduction to Software Engineering"** represents a module offered by Faculty of Information Technology, Department of Software Engineering in the first year, its subject area is "Broad course covering many areas of SEEK" (Software Education Knowledge).. Figure (1-1) illustrates the scheme of module coding and numbering, where the "Introduction to Software Engineering" course is presented as an example.

Figure (1-1) Module Coding and Numbering Scheme



**** SE=Software Engineering, MIS=Management Information Systems, CS=Computer Science, WE=Web Engineering.**

Table (1-1) displays the list of the accredited Software Engineering Knowledge Areas (KA)

Table (1-1) The Knowledge Areas (KA) of Software Engineering

No. of KA	Name of KA
1.	Computational Science and Algorithms (CA
2.	Programming Languages (PL))
3.	Main Computer Components (MCCO)
4.	Software Engineering (SwE)
5+6	Information Sciences and Applications (ISA)
7.	Supplementary Courses (SC)
9.	Graduation Project (GP) / Practical Training (PT)

CHAPTER 2

CURRICULUM DESIGN, ORGANISATION, AND CONTENT

2.1 Outlines of the Degree Programme

Within the general area of Software Engineering (SE), the modules recognise several major subject themes. This represents fundamental material on programming, algorithms and software engineering, the structure and operation of computer systems including a high-level view of processing, memory, data communication and input/output devices, plus operating systems and user interfaces. This includes the theoretical foundations of computing, including programming languages and formal analysis of algorithms and machines.
Details of each module are set out in Chapter (3).

2.2 Requirements for the Degree Programme

The SE programme is covered with different requirements. For obtaining the full award, students should complete 46 modules, 44 X 3-credit hours courses, 1 X 1-credit hour course, and 1 X 2-credit hours course (i.e. a total of 132 credit hours) summarised as follows:

Students should complete 46 modules (132 credit hours) summarised as follows:

-	9 modules (University requirements)	(27 credit hours)	(20.45 %)
-	9 modules (Faculty requirements)	(27 credit hours)	(20.45 %)
-	15 modules (Departmental Compulsories)	(45 credit hours)	(34.09%)
-	2 modules (Departmental Electives)	(6 credit hours)	(4.55 %)
-	9 modules (Supportive modules)	(27 credit hours)	(20.45 %)

The Faculty requirements and University requirements include some computer-oriented modules that account to the Department requirements. (See Chapter (3), Table (3-1) for the titles of these modules).

2.3 Design, Organisation, and Content of Curriculum

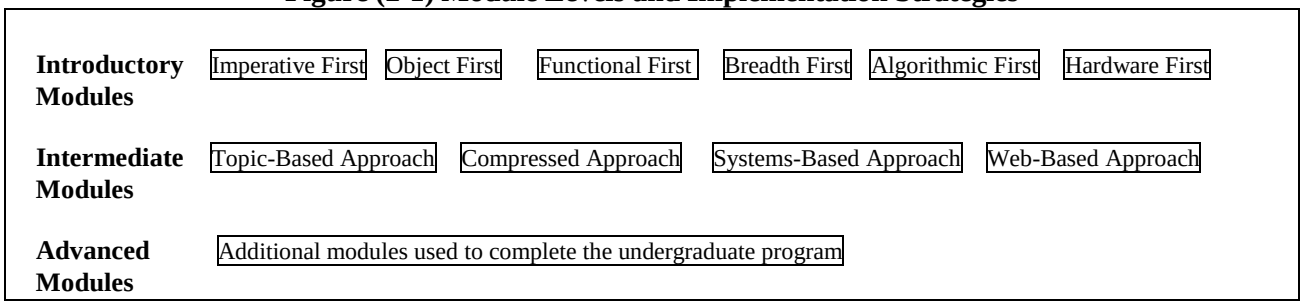
- ***Organisation of Modules:*** The modules are organised into three levels according to the year at which they occur in the curriculum:
  1- Level 1: **Introductory** modules,
  2- Level 2: **Intermediate** modules,
  3- Level 3: **Advanced** modules.

Modules designated as **Introductory** are offered in the first and second years of the Department curriculum. Modules listed as **Intermediate** are usually offered in the second or third year and build a foundation for further study in the field. Modules designated as **Advanced** tend to be taken in later years (third and fourth) and focus on those topics that require significant preparation in the earlier coursework. For these modules, the Department wishes to orient such modules to its own areas of expertise**.**

While these distinctions are easy to understand in their own right, it is important to recognise that there is no necessary relationship between the notions of core and elective - which apply to units in the body of knowledge - and the level of the module. The introductory and intermediate modules concentrate on core material, and the advanced modules include some core material and elective modules.

The point of organising the modules into three levels: **Introductory**, **Intermediate**, and **Advanced** is to provide natural boundaries for defining implementation strategies. The CC2001 report defined many strategies. Figure (2-1) shows these strategies and their relationship in the curriculum.

Figure (2-1) Module Levels and Implementation Strategies



- For Introductory Modules, the Department adopted the **Imperative-First (or Procedural-First) strategy**. The imperative language is C. Then C# is adopted to introduce Object Oriented concepts.
- For Intermediate Modules, the Department adopted **Topic-Based strategy** to preparing for specific areas.
- Some Advanced Modules are selected to attend the departmental objectives and the areas of expertise.

The SE programme is organised to cover some specified knowledge areas selected from the general areas listed in Table (1-1). Table (2-1) shows the areas covered by the specialisation Modules (including those computer-oriented modules taken from the Faculty and University requirements) and the number of modules in each of them. Note that the ratios in Table (2-1) are calculated according to the total number of modules (i.e. 44).

Table (2-1) Distribution of Specialized Courses over the Knowledge Areas (KA) of Software Engineering

	Area	Compulsory Modules		Elective Modules		Total No. of Modules
		No.	(No. /44) %	No.	(No./44) %	
1-	Computational Science and Algorithms (CA)	**3**	**06.81**	**0**	**0.00**	**3**
2-	Programming Languages (PL)	**6**	**13.62**	**0**	**0.00**	**6**
3-	Main Computer Components (CA)	**4**	**09.09**	**0**	**0.00**	**4**
4-	Software Engineering (SwE)	**10**	**22.72**	**4**	**09.09**	**14**

5-	Information Science and Applications (ISA)	**6**	**13.62**	**0**	**0.00**	**6**
6-	Supplementary Courses (SP)	**3**	**06.81**	**0**	**0.00**	**3**
7-	Graduate project (GP) / Practical Training (PT)	**1**	**02.72**	**0**	**0.00**	**1**
	Total	**33**	**75.00**	**any 2**	**04.54**	**37**

- ***The Study Plan.*** The whole modules of the curriculum offered by the SE Department are shown in Appendix A of this Catalogue.

- ***The Guidance Plan***. The Department guides students in their registration and selection of modules during the four years. The Department organizes a guidance plan that is shown in Table (2-2), where UR, FR, DR, and SR indicate University Requirements, Faculty Requirements, Department Requirements, and Supportive Requirements, respectively.

Table (2-2) Guidance Plan for the SE Curriculum

Guidance Plan
(132 Credit Hours)
(2016 - 2017)

Year	Semester	Module Number	Module Title	Prerequisite	Types of Requirements
First	First (18 Credit Hours)	0110101	Arabic Language Skills (1)	----	UR
		0130101	English Language Skills (1)	----	UR
		---	UE (1)	----	UR
		0750113	Programming Fundamentals (1)	----	FR
		0731110	Introduction to Systems and Information Technology	----	FR
		0250101	Differentiation and Integration (1)	----	SR
	Second (18 Credit Hours)	0130102	English Language Skills (2)	0130101	FR
		0111101	National Education	----	UR
		0770110	Introduction to internet and web technology	----	FR
		0750114	Programming Fundamentals (2)	0750113	FR
		0721110	Introduction to Software Engineering	0750113+ 0731110	DR
		0250104	Discrete Structures	----	SR
Second	First (18 Credit Hours)	0721220	Object Oriented Programming	0750114	FR
		0721230	Software Requirements	0721110	DR
		0250231	Introduction to Statistics and Probabilities	----	SR
		0750231	Design Of Logic Circuits	0731110	SR
		0731213	Introduction to World Wide Web Programming	0750114	FR
		----	UE(2)	----	UR
	Second (18 Credit Hours)	0721222	Software Modelling	0721110	DR
		0721221	Object Oriented Data Structures	0721220 + 0250104	DR
		0750272	Numerical Analysis	0250101+0750114	SR
		0750215	Visual Programming	0721220+0731221	FR
		0731221	Database Fundamentals	0721220	SR
		0721240	Computing Ethics	0731110	FR
	First (18 Credit Hours)	0721320	Software Architecture	0721222	DR
		0721330	Software production	0721222	DR
		0721350	Computer Organization and Architecture	0750231	DR
		0731340	Fundamentals of Computer Networks	0721221	SR
		0750322	Design and Analysis of Algorithms	0721221+0250231	SR
		----	UE(3)	----	UR

	Second (15 Credit Hours)	0721322 0721331 0721324 0750333 ----	Software Analysis And Design Software Project Management Advanced Object Oriented Programming Principles of Operating Systems UE(4)	0721230 + 0721320 0721330 0721220 0721350 ----	DR DR DR SR UR
Fourth	**First (13 Credit Hours)**	0721420 0721430 ---- 0721438 0721448 ----	Software Construction and Development Software Testing DE(1) Practical Training Research Project(1) UE(5)	0721322+721324(متزامن) 0721322 ---- 750215+721320 Dept. Agree +90 hours ----	DR DR DR DR DR UR
	Second (14 Credit Hours)	072142 0721421 0721449 ---- 0111100	Graphical User Interface Design Software Re-Engineering Research Project(2) DE(2) Military Sciences(Or UE Non- Jordanians Students)	Dept.Agree+90+hours+0750099 0721420 0721448 ---- ----	DR DR DR DR UR

(UR) University Req. **(UE) University Elective** **(FR) Faculty Req.**
(DR) Dept. Req. **(DE) Department Elective** **(SR) Supplementary Req.**

CHAPTER 3

FULL DESCRIPTION OF MODULES

This chapter presents the full description of the Department modules and those modules from the Faculty and University requirements that are computer-oriented modules.

3.1 Module Descriptor

The Department organised a format for the module descriptor that includes much information on the module. This sub-section presents the components of the adopted module descriptor that are shown in Figure (3-1). The University Quality Assurance Catalogue explains in details the components of the module descriptor.

Figure (3-1) Components of the Module Description

Module Number, Module Title
Providing Department:
Module Coordinator(s):
Year:
Credit:
Prerequisites: Required modules or background
Aims:
Teaching Methods:
Learning Outcomes:
Assessment of Learning Outcomes:
Contribution to Programme Learning Outcome:
Syllabus: Bulleted list providing an outline of the topics covered.
Modes of Assessment:
Textbook and Supporting Materials:
Instructor:

3.2 Introductory Modules

Table (3-1) presents the Introductory (Level 1) modules whose full descriptions are given below.

Table (3-1) Introductory Modules in SE Department

Module Number	Module Title	Prerequisite

250104	Discrete Structures	None
250101	Differentiation and Integration (1)	None
250231	Introduction to Statistics and Probabilities	None
750113	Programming Fundamentals (1)	None
750114	Programming Fundamentals (2)	750113
731110	Introduction to Systems and Information Technology	None
721110	Introduction to Software Engineering	750113 + 731110
750231	Design of Logic Circuits	731110
731221	Database Fundamentals	721220
731340	Fundamentals of Computer Networks	721221
731213	Introduction to World Wide Web Programming	750114
770110	Introduction to internet and web technology	None

0250104 Discrete Structures

Level: 1

Prerequisite: **None**

Aims:
This course is an introduction to Discrete Mathematics for students from the IT majors, covering main topics in number theory, propositional logic, proof techniques, sets and relations, counting techniques, and graph theory, together with selected applications in computer algorithms.

Teaching Methods: 38 hours Lectures (2-3 per week) + 10 hours Tutorial

Synopsis:

Logic: logic operators AND, OR, IFF, XOR, truth table, tautology, equivalence. Normal forms, predicates and quantifiers. Sets: set operations, set identities, power set, cardinality, cross product, power set. Modulo operation, divisibility, GCD and LCM, the Euclidean algorithm. Combinatorics: the addition and multiplication principles, the Pigeonhole principle. Inclusion-exclusion principles, permutations and combinations, permutations on multisets. Recurrence relation: solving first order homogeneous sequences. Methods of proof: mathematical induction. Relations: properties of relations, representation by digraphs, zero-one matrices, transitive closures. Equivalence relations, partial order relations, total order, Hasse diagrams. Graph Theory: complete graphs, complete bipartite, representations by adjacency matrix, incidence matrix, distance matrix. Trees, minimal spanning trees, Euler circuit, the Chinese postman problem. Coloring algorithms, planar graphs, maps and dual graphs.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Textbook

Amin Witno, Discrete Structures in Five Chapters, CreateSpace 2010

0250101 Differentiation and Integration

Level: 1

Prerequisite: **None**

Aims:
This course deals with the following main topics: differentiation of algebraic and transcendental functions, an introduction to analytic geometry, applications of differentiation, and a brief introduction to integration.

Teaching Methods: 38 hours Lectures (2-3 per week) + 10 hours Tutorial

Synopsis:

Functions: Representations of Functions. The Vertical Line Test. Symmetry. Linear Function. Polynomials. Piecewise Defined Functions. Rational Functions. Root Function. Trigonometric Functions. Combinations of Functions: Sum, Difference, Product, Quotient, Composition. Inverse Functions: Functions. Horizontal Line Test. Inverse Trigonometric Functions. Exponential and Logarithmic Functions. Hyperbolic Functions. Limits and Continuity: An Introduction to Limits. Calculating Limits using the Limit Laws. Limits at Infinity and Infinite Limits. Limits Involving ($\sin\theta/\theta$). Continuous Functions. The Derivative: The Derivative as a Function. Differentiation Rules and Higher Derivatives. The Chain Rule. Implicit Differentiation. Tangent Line. Applications of Differentiation: L'Hospital's Rule. Rolle's Theorem; Mean-Value Theorem. Analysis of Functions: Increase, Decrease, and Concavity. Relative Extrema; Graphing Polynomials. Absolute Maxima and Minima. Integration: Anti-derivatives. Indefinite Integrals. Integration by Substitution. The Definite Integral. The Fundamen.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Textbook
Howard Anton, Irl C. Bivens and Stephen Davis, Calculus: Early Transcendentals, 10th Edition, John Wiley & Sons, Inc. 2013. − James Stewart, Calculus: Early Transcendentals, Brooks/ Cole.

250231, Introduction to Probability and Statistics

3 hours per week, 3 credit hours, prerequisite: **none**

Aims: This module aims to help students grasp basic statistical techniques and concepts, and to present real-life opportunities for applying them.

Teaching Method: 30 hours Lectures (2 per week) + 15 hours Tutorials (1 per week)

Synopsis: Descriptive statistics and probability distribution; Sampling distribution Estimation for the mean, variance and proportions; Testing for the mean, variance and proportions; Regression and correlation; One-way analysis of variance.

Assessment: Two 1-hour midterm exams (15% each); Assignments/Quizzes (10%); Tutorial Contribution (10%); 2-hours Final Exam (50%).

Textbooks:

1- D.C. Montgomery and .G.C. Runger, Applied Statistics and Probability For Engineers, 2nd Edition, Wiley, 2002
2- William, Probability and Statistics in Engineering and Management, Wiley, 2002

750113 Programming Fundamentals (1)

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **None**

Aims:

This module aims to introduce computer programming and emphasis in problem solving on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, documentation.
The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers.
.

Teaching Methods *Duration*: 16 weeks, 80 hours in total
Lectures: 32 hours (2 hours per week), *Tutorials*: 16 hours (1 per week),
Laboratories: 32 hours, 2 per week

Synopsis: problem solving strategies, algorithmic language to describe such problem solving, introduces the principles of procedural programming, data types, control structures, data structures and functions, data representation on the machine level. Various problems are considered to be solved using C-like procedural programming language.

Assessment: Two 1-hour midterm exams (15% each); lab (30%); one 2-hours Final Examination (40%)

Textbook:

- P. Deitel & H. Deitel, **C++ How to program,** Pearson Education Limited, 2013.
- Guttag, John. **Introduction to Computation and Programming Using Python**. Spring 2013 edition. MIT Press, 2013. ISBN: 9780262519632. - MIT

750114 Programming Fundamentals (2)

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **750113**

Aims:

This module aims to introduce computer programming and emphasis in problem solving on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, documentation.
The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers
.
Teaching Methods *Duration*: 16 weeks, 80 hours in total
Lectures: 32 hours (2 hours per week), *Tutorials*: 16 hours (1 per week),
Laboratories: 32 hours, 2 per week

Synopsis: Functions definition, Parameters definition and passing , One dimensional array, Two dimensional array, use of main operations of a sequential file: open, reset, rewrite, read, write, eof, Introduction to Class and object, Generics, components reuse, component programming
Various problems are considered to be solved using C-like procedural programming language.
.

Assessment: Two 1-hour midterm exams (15% each); lab (30%); One 2-hours Final Examination (40%)

Textbook

D.S. Malik, Thomson, C++ Programming: From Problem Analysis to Program Design, Sixth Edition, Course Technology, 2011

0731110, Introduction to Information Systems and Technology

3 hours per week, 3 credit hours, prerequisite: None

Course (module) description:
This course provides an introduction to information systems and information technology, information systems development concepts, and application software. It identifies the basic types of business information systems, the major steps of the systems development process and some of the strategies employed to lower costs or improve service. It explains how information is used in organizations and how IT enables improvement in quality, timeliness, and competitive advantage. It also defines the competitive advantages, types of roles, functions, and careers available in IS.

Course (module) objectives:
Students should be acquainted with handling and managing data and information in business organizations and to understand the meaning of "Information Systems and technology and their effects on organizations and the different types of business information systems and the development life cycle.
Students must learn about different Computer Hardware and Software and different types of computer networks. Students should know how to deal with e-commerce

Synopsis: Information theory, dynamic systems, concepts and applications in business organizations, information theory and applications, information systems, information systems in management, management information systems, information technology and computer information systems.

Course/ module components:
1.Books (title , author (s), publisher, year of publication)
Information systems today : managing in the digital world, Joe Valacich, Christoph Schneider,
Harlow: Pearson Education Limited, 2014.
Information Systems Essentials, Editors: Stephen Haag, Maeve Cumming; Published: McGraw-Hill/Irwin, Inc, 2009, Third edition.
- **Support material (s):** slides
- **Study guide (s) (if applicable).**

- **Homework and laboratory guide (s) if (applicable).**

Learning outcomes:
- Knowledge and understanding

1. Know and understand a wide range of principles and fundamentals of Information Systems and Information Technology.
2. The application of IS and IT.

- Cognitive skills (thinking and analysis).

Basic analytical steps of Information Systems and defining the specifications of the IT required in business contexts**.**

- Communication skills (personal and academic).

1. Plan and undertake a small individual project in IS and IT fields.
2. Use the scientific literature effectively and make discriminating use of Web resources.
3. Present seminars in IS and IT fields.

- Practical and subject specific skills (Transferable Skills).

1. Use appropriate computer-based tools.
2. Work effectively with and for others.
3. Strike the balance between self-reliance and seeking help when necessary in new situations.
4. Get knowledge about self learning on the long run.

Assessment instruments:
- Short reports, presentations, Short research projects, Quizzes, and/or Home works (20%)
- First exam (20%)
- Second exam (20%)
- Final exam (40%)

721110, Introduction to Software Engineering
Credit Hours: 3 hours per week (48 hours in total)

Level**:** 1
Prerequisite**:** 0750113 + 0731110
Teaching Methods**:** Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to:
- Teach students the skills needed to execute commercial projects.
- To provide students with the necessary conceptual background for undertaken advanced studies in software engineering, through specialized software engineering courses.

Synopsis: Software Engineering; Software Process; Software Development Life Cycle.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Software Engineering: A practitioner's Approach, R.S. Pressman, Mc Graw Hill, 2010.
2. A Concise Introduction to Software Engineering, Pankaj Jalote, Springer, 2008

750231, Logic Circuit Design

Providing Department: Computer Science, Faculty of IT
Module Coordinator(s):
Year: 2
Credit: 3 credit hours
Prerequisite: 0731110

Aims: This module introduces you to the design and implementation of digital circuits. Topics include: combinational and sequential circuit analysis and design, digital circuit design optimization methods using random logic gates, multiplexers, decoders, registers, counters and programmable logic arrays. Laboratory experiments will be used to reinforce the theoretical concepts discussed in lectures. The lab experiments will involve the design and implementation of digital circuits. Emphasis is on the use computer aided tools in the design, simulation, and testing of digital circuits.

Teaching Methods: 41 hours Lectures (2-3 per week) + 4 hours Tutorials (1 per 3 weeks) + 3 hours Laboratory (1 per 4 weeks)

Learning Outcomes:
A student completing this module should be able to:

1. Define the problem (Inputs and Outputs), write its functions. (A, B, C)
2. Minimize functions using any type of minimizing algorithms (Boolean Algebra, Karnaugh map or Tabulation Method). (A, B)
3. Implement functions using digital circuit (Combinational or Sequential). (A, B)
4. Have knowledge in analyzing and designing procedures of Combinational and Sequential circuits. (B, C)
5. Have knowledge in designing and analyzing circuits with Flip-Flops, Counters and Registers. (B, C)
6. Work effectively with others. (D)
7. Use simulation software, for testing the designed circuit. (C, D)

Assessment of Learning Outcomes
Learning outcomes (1), (2), and (3) are assessed by examinations, tutorial and in the laboratory. Learning outcomes (4), (5), and (6) is assessed by course work/workshops. Learning outcomes (7) is assessed in the laboratory.

Contribution to Programme Learning Outcomes:
A1, A3, A5, B1, B3, C6, D3, D6

Synopsis: Introduction to Digital logic Design; Binary Systems and Codes: Binary Numbers, Octal and Hexadecimal Numbers, Number Base Conversions, Arithmetic Operation with different Bases, Complements, Signed Binary Numbers, Binary Codes: BCD, Gray, ASCII and EBCDIC; Binary Logic and Logic Gates: AND, OR and NOT; Boolean Algebra and Logic Gates: Basic Definition, Basic Theorems, Boolean Functions; Standard Forms: Minterm and Maxterm, Simplification of Boolean Functions using SOP and POS; Logic Operations: NAND, NOR, Exclusive-OR and Equivalence, Integrated Circuits; Gate-Level Minimization: The Map Method, Two- and Three-Variable Map, Four-Variable Map, Product of Sums Simplification, Don’t-Care Conditions, NAND and NOR Implementation, The Tabulation Method,

Simplification of Boolean Functions using Tabulation Method; Analysis and Synthesis of Combinational Circuits: Combinational Circuits, Analysis and Design Procedure, Binary Adders-Subtractor, Decoders and Multiplexers; Analysis and Synthesis of Sequential Circuits: Sequential Circuits, Latches, Flip-Flops: RS, D, JK and T, Analysis of Clocked Sequential Circuits, Design Procedure; Registers and Counters: Registers, Shift Registers, Synchronous Counters, Ripple Counters; Sequential Circuits with Programmable Logic Devices: Introduction, Random-Access Memory, Memory Decoding, Read-Only Memory, Programmable Logic Array.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); one 2-hours Final Examination (40%)

Textbook and supporting material:
1- Morris Mano, Digital Logic, Prentice-Hall, 2012
2- Morris Mano, Charles R. Kime, Logic and computer design fundamentals, Pearson Prentice Hall, 2004
3- Basavaraj,B., Digital fundamentals, New Delhi: Vikas Publishing House, 1999.
4- Kandel Langholz, Digital Logic Design, Prentice Hall, 1988.
5- Rafiquzzaman & Chandra, Modern Computer Architecture, West Pub. Comp., 1988.

731221, Database Fundamentals

3 hours per week, 3 credit hours, prerequisite: **721220**
***Teaching Methods*:** 26 hours Lectures (average 2 per week) + 16 hours Laboratory (1 per week) + 6 hours Tutorials (1 each fortnight)

Aims: This module aims to give the students the main concepts of database, design the database, database models, normalization techniques, query languages, object oriented database, query optimization and database and the web. Further the students have to practice and write some applications regarding the database.

Learning Outcomes:
On completion of this module, student should be able to:
1- Define the general concepts, objectives and the database models
2- Understand the importance of data, and the difference between file management and databases. (A)
3- Understand the design of database management system architectures and environments. (A)
4- Know the principals of database design. (A)
5- Design a database as free-standing applications
6- Apply conceptual design methodologies, in particular conceptual design using Extended Entity Relationship modelling. (A, B, C, D)
7- Apply the relational model and mappings from conceptual designs, in particular normalizations. (A, B, C, D)
8- Explain physical and performance related design considerations. (A)
9- Understand transaction processing. (A)
10- Discuss and apply SQL and the Oracle DBMS. (A, C, D)
11- Work effectively with others.
12- Invoke the database applications with the World-Wide Web browser

Assessment of Learning Outcomes:
Learning outcomes (1) through (7) are assessed by examinations. Learning outcomes (3), (4), and (8) are assessed by projects design and implementation.

Contribution to Programme Learning Outcomes:
A2, A3, A4, A5, B1, B2, B3, C1, C2, C6, D1, D3

Synopsis: Introduction to Data base and DBMS; Database Models; Database Design; Relational Algebra and Relational Calculus; Query Languages (SQL); DB normalization; Database Integrity and Security; Indexing Techniques; Query Optimization; Distributed Data Base; Object-Oriented Database

Textbooks and Supporting Material:
Modern database management , Jeffrey A. Hoffer, V. Ramesh, Heikki Topi, Boston: Pearson, 2016, 12th ed. global edition

Modes of Assessment: Two 1-hour midterm exams (20% each); Lab work (10%); Assignments (10%); Final Examination: written exam (40%)

731340 Fundamentals of Computer Networks

3 hours per week, 3 credit hours, prerequisite: **721221**

Teaching Methods: 40 hours Lectures (2-3 per week) + 8 hours Tutorials

Aims: The aims of this module are to introduce the basic principles of computer network and how the network is working. Type of the topologies and the technologies are also given. The module, however, does not focus on the detailed study of mathematical aspects. Bottom up is the approach used to teach this module.

Learning Outcomes:
On completion of this module, students will be able to
1. Discriminate and appraise different switching technologies
2. Enumerate, differentiate and appraise different WAN and LAN networks
3. Breakdown communication networks into their components
4. Explain the abstract concepts related to layered communication system architecture
5. Select an appropriate communication network architecture
6. Select and devise an appropriate network interconnection solution
7. Outline the communication software implementation issues
8. Configure the network devices like switch and router

Contribution to Programme Learning Outcomes:
A2, A3, A5, B1, B2, C1, C2, C6, D3, D5

Synopsis*:* Networks Models, OSI Model, TCP/IP, Transmission Media and Transmission modes, Multiplexing, Switching Technology: Packet Switching, Virtual Circuit Switching, Cell Switching, Switch Technologies, LAN and WAN.

Textbook and Supporting Material:
1- Data Communications and Networking, ,Behrouz A. Forouzan, Mc GrawHill**,** 4th edition 20*12*
2- Computer Networks and Internets, Douglas E. Comer, Prentice Hall Pub., 2010, 4th Edition
3- Business Data Communications, W. Stallings, Prentice Hall, 5th edition, 2005
Modes of Assessment: Two 1-hour midterm exams (20% each) + Lab work and Coursework (15%) + Tutorial Contribution (5%) + Final (unseen) exam (40%)

……..

0731213, Introduction to Web Programming

3 hours per week, 3 credit hours, prerequisite: 0750114

Course description

This course is intended to give the student advanced issues in website design and implementation. At the course completion, students will have the know-how of designing and implementing web-based applications, completely database-driven web sites.

The course involves two main parts:

- Advanced client-side programming.
- Advanced server-side programming.

Course objective

On successfully completing the module, the students are expected to have gained good knowledge of:

- Implementing advanced server-side programming
- Improving personal productivity concepts through web authoring.

Synopsis: HTML: Basics and Programming; Script Languages; Web Servers: Basics and Programming: Introduction to Web Servers, PHP, Working with Data Types and Operators, String conversion and type juggling, Type casting, Building Functions and Control Structures, Manipulating Arrays, Working with Databases and MYSQL, Regular Expression and Validation.

Course components:

- Books (title, author (s), publisher, year of publication)

Don Gosselin, Diana Kokoska, Robert Easterbrooks, PHP Programming with MySQL. Boston: Course Technology/Cengage L:earning, 2011.

Paul Deitel, Harvey Deitel, Abbey Deitel ,Internet and World Wide Web : how to program. Harlow: Pearson Education Limited, 2012.

- Support material (s)

The world's largest web development site: www.w3schools.com

- Study guide (s) (if applicable).

- Homework and laboratory guide (s) if (applicable).

Teaching methods

Lectures, discussion, groups, tutorials, problem solving, etc.

Learning outcomes:

- Knowledge and understanding.
- Cognitive skills (thinking and analysis).
- Communication skills (personal and academic).
- Practical and subject specific skills (Transferable Skills).

Assessment instruments:

- Short reports, Assignments, Projects, Quizzes, and/or Home works (20%)
- First exam (20%)
- Second exam (20%)

- Final exam (40%)

……

Course Title: Introduction to Internet and Web Technologies

Course code: 0770110

Internet, Internet services, internet applications and tools. World Wide Web, WWW applications and tools, mark-up language, e-mail, web browsers, Web-based research and information resources, World Wide Web Consortium (W3C) ,creating web pages, search engines, FTP.

- **Books (title , author (s), publisher, year of publication)**
  - Web Technology Theory And Practice Paperback – 2012 by M Srinivasan (Author)
  - Web Programming And Internet Technologies: An E-Commerce Approach 1 Pap/Cdr Edition by Porter Scobey (Author), Pawan Lingras (Author) 2012

……

……

3.3 Intermediate Modules

The Intermediate (Level 2) modules are listed in Table (3-2) and their full descriptions are given below.

Table (3-2) Intermediate Modules in Software Engineering Department

Module Number	Module Title	Prerequisite
721220	Object Oriented Programming	750114

721221	Object Oriented Data Structures	721220 +250104
750215	Visual Programming	721220+0731221
721320	Software Architecture	721222
721322	Software Analysis and Design	721230 + 721320
721350	Computer Organization and Architecture	750231
750333	Principles of Operating Systems	721350
721240	Computing Ethics	731110
721230	Software Requirements	721110
721222	Software Modelling	721110
750322	Design and Analysis of Algorithms	721221+250231

Object-Oriented Programming – 721220

Credit Hours: 3 hours per week (48 hours in total)

Level**:** 3

Prerequisite**:** 721220

Teaching Methods**:** Lectures 48 hours, 3 per week, Tutorials(10 from 48 hours), Lab (15 hours)

Aims: This course aims to:
develop an understanding of the principles of the object-oriented paradigm; to provide familiarity with approaches to object-oriented modelling and design; to provide a familiarity with the syntax, class hierarchy, environment and simple application construction for an object-oriented programming language. The module emphasizes developing fundamental programming skills in the context of a language that supports the object-oriented.

Synopsis:, Classes, objects, Methods, visibility, Properties, Static methods and variables, Scope, Overloading, Data Abstraction, Encapsulation, Constructors, Composition, Inheritance, Polymorphism, Abstract classes and methods, Interfaces, and Exception handling.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1- Deitel & Deitel, C# 2010 for Programmers, Publisher: Prentice Hall, 2011, ISBN-10: 0132618206 | ISBN-13: 9780132618205.
2- Purdum, Jack, " Beginning C# 3.0 : an introduction to object oriented programming ", Wiley Publishing, Inc., 2007

721221, **Object Oriented Data Structures**

Credit Hours: 3 hours per week (48 hours in total)

***Level*:** 2
***Teaching Methods*:** Lectures (40 hours), Tutorials (8hours)

Aims:
The objectives of this module are to:
1- Provide a clear introduction to the concepts and the importance of standard data structures.
2- Teach students how to use existing data structures (through suitable APIs), adapt/modify or design new data structures to solve simple and more complex problems**.**
3- Enhance the programming skills of students.

Synopsis: Data Structures, Data types, Abstract Data types, List, Stack, Queue, Tree, Binary Search Tree, Heap, Hash Table, Graph

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:
Data Structures Outside-In With Java, Sesh Venugopal, Prentice Hall, 2006

750215, Visual Programming

Providing Department: Computer Information Systems, Faculty of IT
***Module Coordinator(s)*:**
Year: 2
Credit: 3 credit hours
Prerequisite: **721220**

Aims: This module aims to provide students capabilities to design and implement the applications using visual programming through Microsoft Visual Studio .Net and VC# to develop different types of applications using .Net platform.

Teaching Methods: 32 hours Lectures (2 per week) + 12 hours Tutorials (on average 1 per week) + 16 hours Laboratory (1 per week) + 4 hours Seminar

Learning Outcomes:
On completing this module you should:
1. Have a clear understanding of what comprises a correct program in C# through .Net frame components (A)
2. Have a clear understanding of the object-oriented terminology used to describe features of C# and VC# project with their visual components. (A, C)
3. Have an informal understanding of the operational semantics of object-oriented programs in terms of creation of objects and messages passing between difference interfaces. (A)
4. Be able to design, code, and test C# project, which meet requirements expressed in English. (B, C, D)
5. Be able to understand the documentation for, and make use of, the MSDN library. (A, C)
6. Have a good understanding of the different focus at various stages of the development process. (A, C, D)
7. Have knowledge of design GUI with visual components guidelines. (A, B)
8. Be able to apply the guidelines in learning outcome (7) to a real design problem and justify how they have been used. (A, B)
9. Be able to write a project in C# and VC#, which implements the design in learning outcome (8). (C).

10. Be able to work effectively alone or as a member of a small group working on some programming tasks. (C, D)

Assessment of Learning Outcomes:
Learning outcomes (1), (6), and (8) are assessed by examination and laboratory; learning outcomes (2), (3), and (7) are assessed by tutorial and examination; learning outcomes (4), (5), (9) and (10) are assessed in the laboratory.

Contribution to Programme Learning Outcomes:
A2, A3, A4, B3, C5, C6, D1, D2, D4, D5

Synopsis: **Introducing the Microsoft .NET Platform:** .NET Platform, .NET and Windows DNA, .NET Architecture Hierarchy, .NET Platform features, Multilanguage Development, Platform and Processor Independence, .NET Components, Common Type System CTS, Common Language Specification CLS , .NET Base Class Library (BCL); **Visual Studio.NET IDE:** Visual Studio.NET, Components of VS.NET, Design Window, Code Window, Server Explorer, Toolbox, Docking Windows, Properties Explorer, Solution Explorer, Object Browser, Dynamic Help, Task List Explorer, Features of VS.NET, XML Editor, Creating a Project, Add Reference, Build the Project, Debugging a Project; **Introducing C# Programming:** Data Types, Value Types, Reference Types, Control Structures (if, if-else, switch, for, while, do while, break, continue, return, goto), Understanding Properties and Indexers Accessing Lists (Array) with Indexers, Events, Exception Handling, Using OOP (Object, Class, Constructor/destructor, Inheritance, Polymorphism, Encapsulation); **Windows Forms:** Windows Forms, Adding Controls, Adding an Event Handler, Adding Controls at Runtime, Attaching an Event Handler at Runtime, Writing a Simple Text Editor, Creating a Menu, Adding a New Form, Creating a Multiple Document Interface, Creating a Dialog Form, Using Form Inheritance, Adding a *TabControl,*, Anchoring Controls, Changing the Startup Form, Connecting the Dialog, Using the ListView and TreeView, Controls, Building an *ImageList,* Adding a *ListView,* Using the Details View, Attaching a Context Menu, Adding a *TreeView,* Implementing Drag and Drop, Creating Controls, Creating a User Control, Adding a Property, Adding Functionality, Writing a Custom Control, Testing the Control, Enhancing the Control, Sub classing Controls; **Graphics and Multimedia:** Graphics Contexts and Graphics Objects, Color Control, Font Control, Drawing Lines, Rectangles and Ovals, Drawing Arcs, Drawing Polygons and Polylines, Advanced Graphics Capabilities, Introduction to Multimedia, Loading Displaying and Scaling Images, Animating a Series of Images, Windows Media Player, Microsoft Agent; **ADO.NET:** ADO.NET Architecture, Understanding the *Connection*Object, Building the *Connection* String, Understanding the *Command*Object, Understanding *DataReaders*, Understanding *DataSets* and *DataAdapters,* DataTable, *DataColumn, DataRow,* Differences between *DataReader* Model and *DataSet* Model, Understanding the *DataView*Object ,Working with System.Data.OleDb, Using *DataReaders,* Using *DataSets,* Working with SQL.NET, Using Stored Procedures, Working with Odbc.NET, Using DSN Connection; **Multithreading:** Thread States: Life Cycle of a Thread, Thread Priorities and Thread Scheduling, Thread Synchronization and Class Monitor, Producer/Consumer Relationship without Thread, Synchronization, Producer/Consumer Relationship with Thread Synchronization, Producer/Consumer Relationship: Circular Buffer; **Networking:** Introduction, Establishing a Simple Server (Using Stream Sockets), Establishing a Simple Client (Using Stream Sockets), Client/Server Interaction with Stream-Socket Connections, Connectionless Client/Server Interaction with Datagrams, one Server multi-Clients system; **ASP.NET:** Introducing the ASP.NET Architecture, ASP.NET Server Controls, Working with User, Controls, Custom Controls, Understanding the Web.config File, Using the Global.asax Page,

Modes of Assessment: Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)
)

Textbooks and reference books:

1- H. M. Deitel & J. Deitel, " C# How to Program", Prentice Hall, 2014 fifth edition
2- A.Turtschi et.al. " Mastering Visual C# .Net", Sybex 2002
3- Eric Gunnerson, "A Programmer's Introduction to C#", Apress 2000
4- Anders Hejlsberg et.al. " C# Language Reference", Microsoft Corporation 2000

5- Erric Buttow et al. “C#, your visual blueprint for building .Net application”, Hungry Minds 2002
6- Charles Carroll, “Programming C#”, O’Reily & Associates 2000
Karh Watson “Beginning C#” Wrox Press 2001.

721320, Software Architecture

Course Hours: 3 hours per week (48 hours in total)

Prerequisite: 721222

Level: **3**

Teaching Method: Lectures: 40 hours, Tutorial: 8 hours

Aims:
Successful design of complex software systems requires the ability to describe, evaluate, and create systems at an architectural level of abstraction. This course introduces architectural design of complex software systems. The course considers commonly-used software system structures, techniques for designing and implementing these structures, models and formal notations for characterizing and reasoning about architectures, tools for generating specific instances of an architecture, and case studies of actual system architectures. It teaches the skills and background you need to evaluate the architectures of existing systems and to design new systems in principled ways using well-founded architectural paradigms.

Synopsis: `Architecture in Context, Architecture Styles, Styles and Greenfield Design, Software Connectors, Modelling and Notations, Visualizing Software Architectures, Implementing Architectures.`

Two 1-hour midterm exams (20% each); Course work (20%); Assignments (20%); Final Exam (40%)

Textbooks:

1. *Title: Software Architecture: Foundations, Theory, and Practice*

   Author(s)/Editor(s): R. N. Taylor, N. Medvidovic, and E. M. Dashofy
   Publisher: John Wiley & Sons, 2010.
   ISBN-10: 0470167742
   ISBN-13: 978-0470167748

2. Software Architecture in Practice
   Author(s)/Editor(s): Len Bass, Paul Clements and Rick Kazman
   Publisher: Addison-Weslt, 2007

721322, Software Analysis and Design

Course Hours: 3 hours per week (48 hours in total)

Prerequisite: 721230 + 721320

Level: **3**

Teaching Method: 32 hours Lectures +16 hours Tutorials.

Aims: This course completes the student knowledge on Software Design. This course introduces the major design goals (correctness, reusability, robustness, flexibility). Then the course focuses on basic concepts of software architecture, component technologies, architectural design principles and design patterns. The objective of this course is to introduce and detail the factors and the practices that tend to produce good quality software designs.

Synopsis: **Software Design**: purpose, motivation, design levels, **Software Design Principles(**Flexibility, Reusability, Efficiency), **Object-oriented analysis (Unified Process), Design Patterns, Component Technologies,**

Assessment: Two 1-hour midterm exams (20% each) + Assignments (15%) + Tutorial contributions (5%) + 2-hours final exam (40%).

Textbooks:

1. Introduction to Software Engineering Design: Processes, Principles and Patterns withUML2, Christopher Fox, Addison Wesley, 2007
2. Object Oriented Modelling and Design with UML, Michael Blaha, James Rumbaugh, Person Editions, 2005
3. Software Design: from programming to architecture, Braude Eric, John Wiley & sons, 2004.
4. The Art of Software Architecture: Design Methods and Techniques**,** Stephen T. Albin , John Wiley Sons, 2003.

……..

Computer Organization and Architecture – 721350
Course Hours: 3 hours per week (48 hours in total)
Prerequisite: 750231
Level: 3
Teaching Method: 32 hours Lectures +16 hours Tutorials.
Aims:
This course aims to:

- Provides the basic knowledge about the various digital components used in the organization and design of digital computers.
- Show the general steps that a designer must go through in order to design an elementary basic computer.
- Dealing with the organization and architecture of the central processing unit.
- Presenting the organization and architecture of input-output and memory.

Synopsis: Register Transfer and Micro operations Concepts, Basic Computer Organization and design, Basic Computer Programming, Central Processing Unit Architecture, Pipelining Concepts, Organization and Architecture of Input-Output and Memory.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. **William Stallings, Computer Organization and Architecture (10th Edition), 2016.**
2. **P. Godes, D. A. Godes, Computer Architecture and Organization, 2014.**

……..

750333, Principles of Operating Systems

Providing Department: Computer Science, Faculty of IT
Module Coordinator:
Year: 3
Credit: 3 credit hours
Prerequisite: 750332
***Prerequisite for*:** 750334

Aims:
The aims of this module are to introduce the basic principles of computer systems organization and operation; to show how hardware is controlled by program at the hardware/software interface; to outline the basic OS resource management functions: memory, file, device (I/O), process management, and OS security/protection. Two concrete examples of operating systems are used to illustrate how principles and techniques are deployed in practice.

Teaching Method: 40 hours Lectures (2-3 per week) + 8 hours Tutorials (1 each fortnight)

Learning Outcomes:
On completing the module, students should:
1- Have knowledge and understanding of the overall structure and functionality of a modern operating system and of its interactions with the underlying computer hardware and overlying user-program. (A)
2- Have knowledge and understanding of the operation of the following major components of an operating system: the I/O device manager; the memory manager; the process manager; the file manager; OS security/protection manager (A)
3- Have the ability to design and implement (an emulation of) a prototypical process manager. (B, C)
4- Be aware of how fundamental techniques in (1) and (2) are applied in practice in two distinct modern operating systems. (A)

Assessment of Learning Outcomes:
Learning outcomes (1) and (2) are assessed by examination. Learning outcome (3) is assessed via course project. Learning outcome (4) is not formally assessed.

Contribution to Programme Learning Outcomes
A3, B3, C5.

Synopsis*:* Operating System overview; Operating System Structures: System components, Operating system services, System calls, System structures, Virtual machine; Processes: Process concept, Process scheduling, Operation on process, Cooperative process, Inter process communication; Threads: Thread overview, Benefits, User and kernel threads, Multithreading model, Solaris 2 threads; CPU Scheduling: Basic concept, Scheduling criteria, Scheduling algorithm, Thread scheduling, Algorithm evaluation; Process synchronization and mutual exclusion: Critical section problem, Two task solution, Synchronization hardware, Semaphore, Classical synchronization problem; Deadlock and starvation: System model, Deadlock characterization, Method for handling deadlock, Deadlock prevention, Deadlock avoidance, Deadlock detection, Recovery from deadlock; Memory management: Background, Swapping, Paging, Virtual memory, Background, Demand paging, Page replacement, Allocation of frame, Thrashing; File system implementation and management: File concept, Access method, Directory structure, Protection, File system structure, Allocation method, Free space management, Directory implementation, Efficiency and

performance, I/O management and disk scheduling, Application I/O interface, Kernel I/O subsystem, I/O request handling, Disk structure, Disk scheduling, Disk management, Swap space management, Disk reliability, Stable storage implementation

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Textbooks and Supporting Material:
1- Operating System Concepts, 9th Edition International Student Version
Abraham Silberschatz, Peter B. Galvin, Greg Gagne
ISBN: 978-1-118-09375-7 Addison-Wiley , **2013**

2- A. Silberschatz and Peter Galvin, Applied Operating Systems Concepts, First edition, John Wiley & sons, Inc, 2000
3- J. Bacon, Concurrent Systems: Database and Distributed Systems, 2nd Edition, (ISBN 0-201-177-676), Addison Wesley, 1998.
4-A. S. Tanenbaum, Modern Operating Systems, Prentice Hall, 1992

……..

0721240, Computing Ethics
Credit Hours: 3 hours per week (48 hours in total)
Level**:** 2
Prerequisite**:**
Teaching Methods**:** Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to:

- Introduce the ethical foundations of good professional practice in computing.
- Understand the basic concepts of ethics, moral, law, ergonomics and profession
- Be able to recognize and distinguish different kinds of ethical arguments, Know why professions have codes of conduct.
- Have a basic knowledge of Intellectual Property Rights (IPR) in relation to Copyright and Patents.
- Be aware of the requirements for professionalism in respect of the work of the professional societies and their codes of conduct and practice
- Be able to explain the nature of privacy and how it is protected by different Acts.

Synopsis: Information Technology Changes , Impacts of IT Changes , Introduction to Ethics, Ethics Philosophical Issues, Privacy, Intellectual Property Rights, Computer Crimes, Work environment , Evaluating and Controlling the technology

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Sara Baase, A Gift of Fire: Social, Legal and Ethical Issues for Computer and the Internet, 4nd th., 2012

……..

……..

721230, Software Requirements
Credit Hours: 3 hours per week (48 hours in total)
Level**:** 2
Prerequisite**:** 721110
Teaching Methods**:** Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to:

- Introduce the process, activities, concepts, and techniques needed in the eliciting, analyzing, documenting, validating, and managing requirements for complex systems.
- Explain how requirements engineering fits into a broader systems development process.
- Provide an understanding of the main challenges in requirements Engineering.

Synopsis: Requirements engineering Discipline; Requirement Engineering Process; System Requirements, Software Requirements; Functional Requirements, Non Functional Requirements, Quality Requirements; Domain understanding, Requirements Elicitation, Requirements Evaluation, Negotiation, Specification, Requirements Documentation, Requirement Verification and Validation; Goal oriented and Agent oriented Requirements Engineering.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Klaus Pohl, Chris Rupp, "Requirements Engineering Fundamentals", Rocky Nook Publisher. 2015.
2. Axel van Lamsweerde, "Requirements Engineering: From System Goals to UML Models to Software Specifications", Wiley, 2009.

……..

……..

721222, Software Modeling
Credit Hours: 3 hours per week (48 hours in total) + 1h (Lab practice)

Level**:** 2
Prerequisite**:** 0721210
Teaching Methods**:** Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to:

- Introduce early in the Software Engineering curricula the concept of modeling, its motivations and benefits (lectures).
- Study several modeling frameworks in Software Engineering (lectures).
- Study UML constructs.
- Provide good modeling practices through various real case studies (tutorials, workshops, and laboratory).

Synopsis: Software Modeling; structural |Modeling; Interaction Modeling; Behavioral Modeling.

Assessment: Two 1-hour term exams (20% each), Laboratory works and Quizes (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures; Hassan Gomaa, Cambridge University Press, 2011.
2. Object Oriented Modeling and Design with UML, M. Blaha, J. Rumbaugh, second edition, 2005, Pearson, Prentice Hall

750322, Design and Analysis of Algorithms

Providing Department: Computer Science, Faculty of IT
Module Coordinator(s):
Year: 3
Credit: 3 credit hours
Prerequisite: 750272 + 721221

Aims:
The aim of this module is to learn how to develop efficient algorithms for simple computational tasks and reasoning about the correctness of them. Through the complexity measures, different range of behaviours of algorithms and the notion of tractable and intractable problems will be understood. The module introduces formal techniques to support the design and analysis of algorithms, focusing on both the underlying mathematical theory and practical considerations of efficiency. Topics include asymptotic complexity bounds, techniques of analysis, and algorithmic strategies.

Teaching Methods: 38 hours Lectures (2 per week (including two 1-hour midterm exams)) + 10 hours Tutorials (average 1 hour per week)

Learning Outcomes:
When completing this module, you should be able to:

1. understand basic ideas about algorithms (A)
2. develop efficient algorithms for simple computational tasks (B)
3. reason about the correctness of algorithms (B)

4. understand the concepts of time and space complexity, worst case, average case and best case complexities and the big-O notation (A)
5. compute complexity measures of algorithms, including recursive algorithms using recurrence relations (B)
6. understand the range of behaviours of algorithms and the notion of tractable and intractable problems (A, B)
7. know and understand a wide range of searching and sorting algorithms (A, B)

Assessment of Learning Outcomes:
All learning outcomes are assessed by examinations and tutorials. Learning outcomes (4), (5), and (6) are assessed by examinations and coursework.

Contribution to Programme Learning Outcomes:
A1, A2, B1, B2, B3

Synopsis: Introduction, Algorithm definition, Algorithm Analysis; **Mathematical Induction;** Summation Techniques; Recurrence Relations; **Design & Analysis of Algorithms: Divide and conquer**, Greedy Algorithm, Dynamic Programming, Backtracking, Branch-Bound; Lower Bound Theory; Sorting and Searching; NP-Complete Problems: Basic Concepts, NP-Hard & NP-Complete Problem

Modes of Assessment:
Two 1-hour midterm exams (15% each); Tutorial contributions (5%), Coursework (15%); Final written Examination (50%)

Textbooks and Supporting Material:
1- Jon Kleinberg, Eva Tardos, Algorithm design**,** Boston: Pearson Education Limited, 2014.
2- Alwan, Raad F., Design and Analysis of Algorithms, Dar Majdalawi Publication & Distribution, Amman, 2010.
3- Sara Baase, Computer Algorithms: Introduction to Design and Analysis, Third Edition, Addison-Wesley, 2000.
4- Udi Manber, Introduction to Algorithms: a Creative Approach, Addison-Wesley, 1997.
5- T. Cormen, et.al., Introduction to Algorithms, 1999.
6- R. Sedgewick, Algorithms in C++, 2002.

3.4 Advanced Modules

In this sub-section, the full descriptions of Level 3 modules are presented. Table (3-3) shows these modules and their descriptions are given below.

Table (3-3) Advanced Modules in SE Department

Module Number	Module Title	Prerequisite
721420	Software Construction and Development	721322+721324
721430	Software Testing	721322
721421	Software Re-Engineering	721420
721331	Software Project Management	721330
721330	Software Production	721222
721324	Advanced Object Oriented Programming	721220
721423	Graphical User Interface Design	721320+750215
721438	Practical Training	90h +Department Agreement
721448	Research Project1	90h +Department Agreement
721449	Research Project2	721448

721420, Software Construction and Development

Credit Hours: 3 hours per week (48 hours in total)
Level**:** 2
Teaching Methods**:** Lectures (40 hours), Tutorials (8hours)

Aims:
This course aims to:

- The needed strategies for mapping the architecture and design of applications into source code

- Develop code with errors avoidance techniques
- To discuss the latest technologies for building software applications and knowing which of these technologies to use depending on the requirement specifications of our software problem.
- Preparing the student to pursue their career in software construction by introducing to them the needed software skills to achieve this target.

Synopsis: Implementation Strategies; Mapping design outcomes into Code/ derivation of code from design models; Mapping the Object Model to a Database Schema; Construction tools technology/ Techniques and tools supporting code development API use, Code reuse and libraries; Choosing the appropriate programming language; Defensive Programming; Code-Tuning Strategies

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd Edition, 2010; Bernd Bruegge, Adjunct, Carnegie Mellon University; Allen H. Dutoit, Technical University of Munich

---

0721430, Software Testing
Credit Hours: 3 hours per week (48 hours in total)
Level: 4
Prerequisite: 0721322
Teaching Method: Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to address topics in the verification and validation (V&V) of software. This course aims to teach in depth study of verification and validation strategies and techniques as they apply to the development of quality software. Topics include test planning and management, testing tools, technical reviews, formal methods and the economics of software testing.

Synopsis:
Software testing; Test data & test cases & Testing Levels; Testing Strategies; Software quality metric; Testing Tools, Assessing and improving the validation process; Functional & Structural testing; Integration and system testing; Equivalence Class Testing; Performance and Load Testing; Web based Testing; Regression Testing

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Software Verification and Validation for Practitioners and Managers, by Steven R. Raktitin, ed. Artech House, 2nd Edition, ISBN 1-58053-296-9, 2001.
2. Managing the Testing Process, Rex Black , John Wiley & Sons , August 2009 , 0470404159

Software Re-Engineering – 721421

Credit Hours: 3 hours per week (48 hours in total)

Level**:** 3

Prerequisite**:** 721220

Teaching Methods**:** Lectures (48 hours), Tutorials, (10 from 48 hours)

Aims: This course aims to:
provide students with knowledge about: software re-engineering, different re-engineering techniques used in program understanding and software maintenance, and aware of the challenges inherent in the re-engineering of software systems.
.

Synopsis:, Software evolution, Software maintenance; Software reengineering, Reverse engineering, Code refactoring, Code slicing, Code migration, Program comprehension, Restructuring programs, Program translation, Data reverse engineering, Legacy Systems.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. A. Afshar Alam, Tendai Padenga, Application Software Reengineering, Pearson Education India, 2010

2. Salvatore Valenti, Successful Software Reengineering, IRM Press, 2002

……

0721331, Software Project Management

Credit Hours: 3 hours per week (48 hours in total)
Level: 3
Prerequisite: 0721330
Teaching Methods: Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to provide students with knowledge and skills for wide range of principles and tools available to the software engineer and software manager, such as planning, organization,

and monitoring of all software life-cycle phases. Also, solve a wide range of problems related to the software management. Moreover, students could plan and undertake a major individual project, and prepare and deliver coherent and structured verbal and written technical report.

Synopsis:

Management concepts; General project management, Classic management models; Project management roles, Enterprise/Organizational management structure Software management types; application of computing in a business context; Principles and tools available to the software manager; Project planning; Project personnel and organization; Project control; Software configuration management; Software Quality

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. B. Hughes and M. Cotterell. Software Project management. Fifth edition, McGraw-Hill, 2012.
2. Pro Web Project management. Justin Emond and Chris Stenis, 2011, APress

---

721330, Software Production

Credit Hours: 3 hours per week (48 hours in total)

Level**:** 3

Prerequisite**:** 721222

Teaching Method: Lectures (40 hours), Tutorials (8hours)

Aims**:** This course aims to provide students with knowledge and skills for designing, creating, analyzing and applying software development processes. It introduces and details the commonly used software process models (Prescriptive software process models). It also covers the modeling aspect of software processes (Descriptive software process models).

Synopsis*:* Basic software processes concepts (process, activity, work product, role, agent, product flow, process script,…); Concepts of Descriptive and Prescriptive process models; Families of prescriptive process models: Waterfall, Prototyping, Iterative and Incremental approaches, Evolutionary approaches, Spiral software process, RUP, Agile software processes (XP, Scrum,…); Software Process Modeling languages (MVP-L, SPEM).

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Software Process Definition and Management, J. Munch et al, Springer Verlag Berlin Heidelberg, 2012
2. Software Engineering: A Practitioner Approach. R. S Pressman, M. Bruce, Mc Graw Hill, 2015

Advanced Object Oriented Programming – 721324

Credit Hours: 3 hours per week (48 hours in total)

Level**:** 3

***Prerequisite*:** 721220

***Teaching Methods*:** Lectures (48 hours), 3 per week, Tutorials/Lab(10 from 48 hours)

Aims: This course aims to:
provide students with knowledge about: manipulate several core data structures in python, advanced issues related to writing classes and methods, understand the basic ideas behind class hierarchies, polymorphism, and programming to various interfaces, error handling, basic programming of paradigms including object-oriented, imperative, functional programming, use an existing library to implement a graphical user interface and develop multithreading programming, develop Python scripts for publishing on the Web.

Synopsis:, Object-oriented, Paradigms, Sequences, Containers, Dictionary, Functional programming, Error handling, Class, API libraries, Event-Driven, Inheritance, polymorphism.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

Guttag, John V. (Author)

New Delhi: PHI Learning Private Limited, 2014
ISBN : 978-81-203-4866-0
Introduction to computation and programming using python

721423, Graphical User Interface
Credit Hours: 3 hours per week (48 hours in total)
Level: 3
Prerequisite: 0721320+0761220
Teaching Method: Lectures (40 hours), Tutorials (8hours)
Aims: This course aims to provide students with knowledge and skills for designing, creating, analyzing and applying systematic methodologies for design and evaluating GUI. Writing of GUI specifications and implementation of GUI applications in the any programming language is also covered. The course also aims to provide students with the principles for predicting the usability of human computer interaction.

Synopsis: human-computer interaction; Human aspects; Technology aspects; User Interface development cycle; Usability and Design; basic GUI principles; Design practice; Designing Websites using the Design Process; User Support; Screen design; Evaluation Techniques

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:
B.Shneiderman,Designing the User Interface:Strategies for Effective Human Computer Interaction,Addison-wesley,1998
Soren Lauesen , User Interface Design: A Software Engineering Perspective, Addison Wesley, 2005

721438, Practical Training

3 hours per week, 3 credit hours, prerequisite: **Department Agreement**
(Students can take this module on completing 90 credit hours at least).

Aims: The main aim of this module is that students will have practice in different industrial, commercial, administrative enterprises or companies. By this module, students may apply, in the real world, what they have learned during the first three years of their study in the University. The module also aims to teach students how to be self-confident when they face problems in their practical life.

Duration: At least 9 weeks (18 training hours per week at least). This may be distributed onto two semesters at most.

Regulations for Training: Students who register on practical training module should not register on modules with total credit hours more than 15 hours per week including the training module itself. Students must, therefore, be full-time trainees for at least 2 days per week. Students should arrange their timetable for other modules in a way that enables them to enrol in the pre-specified enterprise or company at least two days per week during the semester period.

Assessment: A committee from the Department supervises the students along their training period, where one supervisor is assigned on one group of students. The student should submit a technical report to this committee in 2 weeks time after completing the training session. In addition, the trainer body presents a report to the committee. The grade "pass" is given to students who complete the training requirements successfully and discuss their reports with the supervision committee.

……

721448, Research Project (1)
Credit Hours: 1 hour per week (16 hours in total)
Level: 4
Prerequisite: Exceeding 90 credit hours + Department Agreement
Teaching Methods: Lectures (16 hours), Tutorials/Lab (7 hours)

Aims: The aims for the project course are:
1- To learn real project definition, analysis, design, implementation, testing, writing, and presenting
2- To manage and execute a substantial project in a limited time.
3- To identify and learn whatever new skills are needed to complete the project.

4- To apply design and engineering skills in the accomplishment of a single task.

Synopsis: Proposed treatment; Work Plan; Requirement elicitation; Requirement modelling analysis; SRS document; Prototype; Software architecture

Assessment: Two 1-hour term exams (20% each), Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:
C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.
Bernd Bruegge, Allen Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java, Prentice Hall, 2003
Michael R. Blaha, James R Rumbaugh, Object-Oriented Modeling and Design with UML, Prentice Hall, 2005

721449, Research Project (2)
Credit Hours: 2 hour per week (32 hours in total)
Level: 4
Prerequisite: 721448

Teaching Methods: Lectures (16 hours), Tutorials/Lab (7 hours)

Aims: The aims for the project course are:
5- To learn real project definition, analysis, design, implementation, testing, writing, and presenting
6- To manage and execute a substantial project in a limited time.
7- To identify and learn whatever new skills are needed to complete the project.
8- To apply design and engineering skills in the accomplishment of a single task.

Synopsis: Work Plan; SRS document; Low level (detailed) Design; Code Construction; Testing technique; Writing Techniques.

Assessment: Two 1-hour term exams (20% each), Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:
C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.
Bernd Bruegge, Allen Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java, Prentice Hall, 2003
Michael R. Blaha, James R Rumbaugh, Object-Oriented Modeling and Design with UML, Prentice Hall, 2005

3.5 Elective Modules

Each student should select 2 modules out of a list of 4 modules according to his/her interest. The Department has a list of elective modules, which can be updated according to the staff expertise and the most recent trends in the field of SE. The current list of such modules is shown in Table (3-4), where some modules are marked with (R) to indicate that these modules are research-oriented according to the staff expertise.

Table (3-4) Elective Modules in Computer Science Department

Module Number	Module Title	Prerequisites
0721443	Secure Software Construction	0721430
0721439	Special Topics In Software Engineering	721322
0721445	Introduction to Cloud Computing	0721322
0721422	Web Software Engineering	0721420+0731213

0721443, Secure Software Construction
Credit Hours: 3 hours per week (48 hours in total)
Level**:** 4
Prerequisite: **0721430**
Teaching Method: Lectures (40 hours), Tutorials (8hours)

Aims: This course aims to teach security issues in different phases of software development (requirements, architecture, design, coding, testing). It also covers threats, vulnerabilities and attacks for Web and Mobile applications.
Keywords:
Risk Analysis, Risk Assessment, Systemic threats, Threat Modelling, Secure software development life cycles, Securing Web and Mobile Applications.

Textbooks:

1. Enterprise Software Security: A Confluence of disciplines, Kenneth R. Van Wyk, Dan S. Peters, Diana L. Burley, Addison-Wesley Professional, 2014.
2- Secure Software Design, Theodor Richardson, Charles Thies, Jhones and Barlett Learning Editors, 2013.
3- Core Software Security: Security at the Source, James Ransome, Anmol Misra, Auerbach Publications, 2013.
4. - Secure and Resilient Software Development , Mark S. Merkow, Lakshmikanth Raghavan, Auerbach Publications, 2010.

721439, Special Topics in Software Engineering

Credit Hours: 3 hours per week (48 hours in total)

Level: 4

Prerequisites: 721322

Teaching Methods: Lectures, Tutorials

Aims:
This course is intended to address a special topic selected among emergent topics of software engineering. The selected topic may concern new software engineering paradigms, or development methodology, or software process, or techniques, or languages or software tools.

Synopsis: key words specific to the selected topic**.**

Assessment: Two 1-hour term exams (20% each), Assignments (20%), Final Examination: 2-hours written exam (40%).

Textbook: Recent books that cover the selected topic.

721445, Introduction to Cloud Computing
Credit Hours: 3 hours per week (48 hours in total)
***Level*:** 4
***Prerequisite*:** 721322
***Teaching Methods*:** Lectures (36 hours), Tutorials/Lab (12 hours)

Aims: This course aims to provide students with a basic knowledge about the Cloud computing paradigm, its relevant concepts and associated technologies. The Cloud computing paradigm is introduced from the software engineering perspective.

Synopsis: Cloud Computing paradigm, Cloud Computing architecture, Platform as a Service, Software as a Service, Service-oriented computing, map-reduce paradigm, Schema-free databases, multi-tenancy.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Gautam Shroff, "Enterprise Cloud Computing: Technology, Architecture, Applications", Cambridge University Press, 2014.
2. Armando Fox, David Patterson, "Engineering Software as a Service: An Agile Approach Using Cloud Computing", Strawberry Canyon LLC Publisher, 2013.

0721422, Web Software Engineering

Credit Hours: 3 hours per week (48 hours in total)
***Level*:** 2
***Teaching Methods*:** Lectures (40 hours), Tutorials (8hours)

Aims: The World Wide Web has become a major delivery platform for information resources. Many applications continue to be developed in an ad-hoc way, contributing to problems of usability, maintainability, quality and reliability. This course examines systematic, disciplined and quantifiable approaches to developing of high-quality, reliable and usable web applications. The course introduces the methodologies, techniques and tools that support their design, development, evolution, and evaluation.
The goals of the course are as follows:

- To be able to analyze and design comprehensive systems for the creation, dissemination, storage, retrieval, and use of electronic records and documents.
- To learn and use some of the client-side and server-side languages used to manipulate information on the World Wide Web – i.e. ASP.NET, and Javascript.
- To learn techniques and evaluation metrics for ensuring the proper operability, maintenance and security of a web application.

Synopsis: Web Engineering; Requirements Engineering for Web Applications; Modeling Web Applications; Web Application Architectures; Technology-aware Web Application Design; Usability of Web Applications; Technologies for Web Applications; Web Project Management; The Web Application Development Process; Security for Web Applications

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Engineering Web applications, by Sven Casteleyn et. al. Springer 2009.
2. Web Engineering: A Practitioner's Approach by Roger Pressman and David Lowe, McGraw-Hill, 2009.
3. HTML and CSS: Comprehensive 7th edition, by by Denise M. Woods and William J. Dorin. Publisher: Cengage Learning; (2012) ISBN-10: 1133526144
4. Internet & World Wide Web How to Program, 5/e

   Paul J. Deitel, Harvey M. Deitel, Abbey Deitel, Pearson Education 2012

APPENDIX A

THE PREREQUISITE RELATIONSHIPS BETWEEN MODULES

APPENDIX B
STUDY PLAN
OF
SOFTWARE ENGINEERING PROGRAMME