CONTENT

CHAPTER 1: INTRODUCTION TO CURRICULUM DESIGN

1.1 Fundamental Concepts ............................................. 2
2.1 Format of the Module Coding Adopted ....................... 3

CHAPTER 2: CURRICULUM DESIGN, ORGANISATION, AND CONTENT

2.1 Outlines of the Degree Programme .............................. 4
2.2 Requirements for the Degree Programme ..................... 4
2.3 Design, Organisation, and Content of Curriculum ........... 4

CHAPTER 3: FULL DESCRIPTION OF MODULES

3.1 Module Descriptor .................................................. 7
3.2 Introductory Modules ............................................. 7
3.3 Intermediate Modules ........................................... 16
3.4 Advanced Modules ............................................... 22
3.5 Elective Modules .................................................. 31

APPENDIX A: THE PREREQUISITE RELATIONSHIPS BETWEEN MODULES ............................................. 36

APPENDIX B: STUDY PLAN OF COMPUTER SCIENCE PROGRAMME ............................................. 37
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 report of the Software Engineering Curricula 2004 project (SE2004). The SE2004 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 SE2004 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 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 entail 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, "721221 Object Oriented Data Structures" represents a module offered by Faculty of Information Technology, Department of Software Engineering in the second year, in the area of Programming Fundamentals, and the module title...
is Object Oriented Data Structures. Figure (1-1) illustrates the scheme of module coding and numbering, where the Object Oriented Data Structures module is presented as an example.

**Figure (1-1) Module Coding and Numbering Scheme**

- **Faculty number**
  1 = Art, 2 = Science, .. 7 = Information Technology

- **Department number within the Faculty**
  21 = SE, 31 = MIS, 50 = CS and 61 = CIS**

- **Year number**
  1 = First year, 2 = Second year, 3 = Third year, 4 = Fourth year

- **SE subjects area number**
  1 = Broad courses, covering many areas of SEEK
  2 = Design, Modelling and Implementation courses
  3 = Process Oriented Courses
  4 = Project / Training

**Identifying unit number within area**

---

**SE=Software Engineering, MIS=Management Information Systems, CS=Computer Sciences, CIS=Computerized Information Systems**
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) (19.56 %)
- 8 modules (Faculty requirements) (24 credit hours) (17.39 %)
- 17 modules (Departmental Compulsorries) (45 credit hours) (36.95 %)
- 2 modules (Departmental Electives) (6 credit hours) (4.34 %)
- 10 modules (Supportive modules) (30 credit hours) (21.73 %)

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

<table>
<thead>
<tr>
<th>Introductory Modules</th>
<th>Imperative First</th>
<th>Object First</th>
<th>Functional First</th>
<th>Breadth First</th>
<th>Algorithmic First</th>
<th>Hardware First</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Modules</td>
<td>Topic-Based Approach</td>
<td>Compressed Approach</td>
<td>Systems-Based Approach</td>
<td>Web-Based Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Modules</td>
<td>Additional modules used to complete the undergraduate program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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 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) Specialisation Areas

<table>
<thead>
<tr>
<th>Area</th>
<th>Compulsory Modules</th>
<th>Elective Modules</th>
<th>Total No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No.</td>
<td>(No. /46) %</td>
<td>No.</td>
</tr>
<tr>
<td>1 - Algorithms and Computing Science</td>
<td>3</td>
<td>6.52 %</td>
<td>0</td>
</tr>
<tr>
<td>2 - Programming</td>
<td>5</td>
<td>10.86 %</td>
<td>0</td>
</tr>
<tr>
<td>3 - Computing Architecture</td>
<td>4</td>
<td>8.69 %</td>
<td>0</td>
</tr>
<tr>
<td>4 - Software Engineering</td>
<td>12</td>
<td>20.08%</td>
<td>4</td>
</tr>
<tr>
<td>5 - Applications and Sciences of Information</td>
<td>4</td>
<td>8.69%</td>
<td>0</td>
</tr>
<tr>
<td>6 - Statistics and Numerical Analysis</td>
<td>3</td>
<td>6.52 %</td>
<td>0</td>
</tr>
<tr>
<td>7 - Practical Modules</td>
<td>1</td>
<td>2.17 %</td>
<td>0</td>
</tr>
<tr>
<td>8 - Training</td>
<td>1</td>
<td>2.17 %</td>
<td>0</td>
</tr>
<tr>
<td>9 - Project</td>
<td>2</td>
<td>4.34 %</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>35</td>
<td>70.04%</td>
<td>any 2</td>
</tr>
</tbody>
</table>

- **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>
<thead>
<tr>
<th>Year</th>
<th>Semester</th>
<th>Module Number</th>
<th>Module Title</th>
<th>Prerequisite</th>
<th>Types of Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>First</td>
<td>0110101</td>
<td>Arabic Skills (1)</td>
<td></td>
<td>UR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0130101</td>
<td>English Skills (1)</td>
<td></td>
<td>UR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750113</td>
<td>Programming Fundamentals (1)</td>
<td></td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0731110</td>
<td>Introduction to Systems and Information Technology</td>
<td></td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750101</td>
<td>Differentiation and Integration (1)</td>
<td></td>
<td>SR</td>
</tr>
<tr>
<td>Second</td>
<td>First</td>
<td>0130102</td>
<td>English Skills (2)</td>
<td>0130101</td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9111101</td>
<td>National Education</td>
<td></td>
<td>UR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750114</td>
<td>Programming Fundamentals (2)</td>
<td>0750113</td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721110</td>
<td>Introduction to Software Engineering</td>
<td>0750113+0731110</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750104</td>
<td>Discrete Structures</td>
<td></td>
<td>SR</td>
</tr>
<tr>
<td>Second</td>
<td>Second</td>
<td>0721222</td>
<td>Object Oriented Programming</td>
<td></td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721230</td>
<td>Software Requirements</td>
<td>0721110</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0250231</td>
<td>Introduction to Statistics and Probabilities</td>
<td></td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750231</td>
<td>Design Of Logic Circuits</td>
<td>0731110</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0731213</td>
<td>Introduction to World Wide Web Programming</td>
<td>0750114</td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721240</td>
<td>Computing Ethics</td>
<td>0731110</td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721222</td>
<td>Software Modelling</td>
<td>0721222</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721221</td>
<td>Object Oriented Data Structures</td>
<td>0721220 + 0250104</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750272</td>
<td>Numerical Analysis</td>
<td>0250104+0750114</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0761220</td>
<td>Visual Programming</td>
<td>0721220</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0761235</td>
<td>Database Fundamentals</td>
<td>0721220</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721240</td>
<td>Computing Ethics</td>
<td>0731110</td>
<td>FR</td>
</tr>
<tr>
<td>Fourth</td>
<td>First</td>
<td>0721320</td>
<td>Software Architecture</td>
<td>0721222</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721330</td>
<td>Software production</td>
<td>0721222</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750332</td>
<td>Computer Architecture</td>
<td>0750231</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0761340</td>
<td>Fundamentals of Computer Networks</td>
<td>0721221</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750322</td>
<td>Design and Analysis of Algorithms</td>
<td>0721221+0250231</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721240</td>
<td>Computing Ethics</td>
<td>0731110</td>
<td>FR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721322</td>
<td>Software Design</td>
<td>0721320 + 0721232</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721331</td>
<td>Software Project Management</td>
<td>0721330</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721324</td>
<td>Advanced Object Oriented Programming</td>
<td>0721220</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750333</td>
<td>Principles of Operating Systems</td>
<td>0750332</td>
<td>SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721322</td>
<td>Software Design</td>
<td>0721320 + 0721320</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721331</td>
<td>Software Project Management</td>
<td>0721330</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721324</td>
<td>Advanced Object Oriented Programming</td>
<td>0721220</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0750333</td>
<td>Principles of Operating Systems</td>
<td>0750332</td>
<td>SR</td>
</tr>
<tr>
<td>Fourth</td>
<td>Second</td>
<td>0721420</td>
<td>Software Construction and Development</td>
<td>0721322</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721430</td>
<td>Software Testing</td>
<td>0721322</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721437</td>
<td>Practical Training</td>
<td>Dept.Agree+90 hours</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721423</td>
<td>Graphical User Interface Design</td>
<td>761220+721320</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721448</td>
<td>Research Project(1)</td>
<td>Dept.Agree +90 hours</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721420</td>
<td>Software Re-Engineering</td>
<td>0721420</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721422</td>
<td>Web Software Engineering</td>
<td>0721240+0731213</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0721449</td>
<td>Research Project(2)</td>
<td>0721448</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0111100</td>
<td>Military Science(Or UE Non-Jordanians Students)</td>
<td></td>
<td>UR</td>
</tr>
</tbody>
</table>

(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](image)

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

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

<table>
<thead>
<tr>
<th>Module Number</th>
<th>Module Title</th>
<th>Prerequisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>250104</td>
<td>Discrete Structures</td>
<td>None</td>
</tr>
<tr>
<td>250101</td>
<td>Differentiation and Integration</td>
<td>None</td>
</tr>
<tr>
<td>750231</td>
<td>Introduction to Statistics and Probabilities</td>
<td>None</td>
</tr>
<tr>
<td>750113</td>
<td>Programming Fundamental (1)</td>
<td>None</td>
</tr>
<tr>
<td>750114</td>
<td>Programming Fundamental (2)</td>
<td>750113</td>
</tr>
<tr>
<td>731110</td>
<td>Introduction to Systems and Information Technology</td>
<td>None</td>
</tr>
<tr>
<td>721110</td>
<td>Introduction to Software Engineering</td>
<td>750113, 731110</td>
</tr>
</tbody>
</table>
721110, Introduction to Software Engineering

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

Level: 1

Prerequisite: 750113, 731110

Aims:
This module aims to provide students a concise introduction to software engineering. It gives a survey of the core concepts, principles and techniques used in software engineering. This module gives an introduction to methods for analysis, design, testing, and implementation of small and medium size software systems. Simple and realistic case studies will be used along all the software process steps.

Teaching Methods: Lectures: 36 hours, Tutorials: 12 hours


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

Textbook and supporting material:
1- Jalote, Pankaj A Concise Introduction to Software Engineering, Springer Verlag, 2008
   http://www.cse.iitd.ac.in/~jalote/ConciseIntroToSE

   http://www.software-engin.com

210104, Discrete Structures

3 hours per week, 3 credit hours, prerequisite: none
Aims:
This module will introduce the student to the basic language and ideas of discrete mathematics that occur in all branches of information technology. It will also begin the process of training the student to argue correctly, both informally and formally, about these structures. The student will begin to learn the use of abstract analysis to solve concrete problems.

Teaching Methods: 32 hours Lectures (2 per week) + 16 hours Tutorials (1 per week)

Synopses: Arithmetic: The standard discrete number systems and the arithmetical operations on them with their properties; Sets and Functions: Standard set and function notation and terminology. Boolean operations on sets. Injective and surjective functions. Composition of functions; Logic: The connectives (or, and, not, implies, if and only if), Formulae of propositional logic, Truth tables, Tautologies and logical equivalence, Normal forms, The quantifiers (for all, there exists); Binary Relations: Definitions and examples, Properties of relations, Digraphs and representations of relations, Equivalence relations and Partitions, Combining relations and closure operators, Order relations, Recurrence Relations: Construction an solutions; Induction: The principle of mathematical induction, with many examples. Structural induction; Combinatory: Inclusion Exclusion principle, Binomial coefficients and permutations, Pascal's triangle. Summing series involving binomial coefficients.

Modes of Assessment:
Two 1-hour midterm exams (15% each); Coursework (15%); Tutorial Contribution (5%); Final (unseen) 2-hour examination (50%)

Textbooks and Supporting Material:

There is not a book, which covers exactly the material in this module. The above book covers a large part of the module but also contains additional material, some of which is covered in later modules.

There are many books on discrete mathematics, which have useful features. For example

761211, Windows Programming

3 hours per week, 3 credit hours, prerequisite: 721120

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

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 (15% each); Assignment 15%; Tutorial Contribution (5%); 2-hours Final Exam (50%: 35% Written Exam + 15% Practical Exam)

**Textbooks and reference books:**
4- Anders Hejlsberg et.al. “C# Language Reference”, Microsoft Corporation 2000

----------------------------------------------------------------------------------------


**Multimedia Software Packages:**
Flash, Macromedia, Photoshop

----------------------------------------------------------------------------------------
731270, World Wide Web: Concepts and Programming

3 hours per week, 3 credit hours, prerequisite: 750112

Aims:
This module aims to give students an introduction and general concepts of the Internet and Intranet technology, the World Wide Web, TCP/IP and Web design languages (HTML, CSS, JavaScript, and ASP). It also involves the necessary background that student needs to develop different tasks of programming aspects concerning the foregoing objectives. Sufficient study levels are supposed to be studied and learned by the students within the course for the sake of applying the different fields of education, learning, economical, E-Business and other approaches.

Teaching Methods: 32 hours Lectures (2 per week) + 8 hours Tutorials (1 per 2 weeks) + 24 hours Laboratory (1-2 per week)

Synopsis: Internet and Intranet Technology: Concepts, protocols, Services, and architecture, TCP/IP Architecture and Protocols (Client & Server), DNS, Internet Service Providers (ISP), Internet Services: USENET News, E-Mail, FTP, and Telnet; The Web: Basic Concepts, WWW and Web Servers, Links: Hyperlinks & Hypermedia, Web pages and home pages, Browsers & Search Engines; Introduction to Markup Languages; Editing HTML, HTML Tags: Headers, HTML Tags: Text Styling and Formatting, and linking; HTML Tags: Images and Image maps; Basic HTML Lists and Tables; Basic HTML Forms and Frames; Frames and Cascading Style Sheets; Cascading Style Sheets and Introduction to Client Scripting; Simple JavaScript Programs; JavaScript: Control Structures, if, if/else, While, for, and switch. JavaScript: Break and Continue statements; JavaScript: Functions, Arrays.

Modes of Assessment: Two 1-hour midterm exams (15% each); Lab work (15%); Tutorial contribution (5%); 2-hours Final Exam (50%).

Textbooks and reference books:
2- Douglas Comer, Computer Networks & Internets, Prentice Hall, 2003
6- Ellie Quigley, “JavaScript by Examples”, Prentice Hall, 2004
8- Susan Anderson-Freed, “Weaving a Website: Programming in HTML, Java Script, Perl and Java”, Prentice Hall, 2002

Website(s):
1. www.w3schools.com
2. www.webteacher.org
3. www.microsoft.com
4. www.whatis.com
5. www.idocs.org
6. www.w3.org
7. www.webdeveloper.com
8. www.javascriptmall.com
10. www.Deitel.com
721221, Object-Oriented Data Structures

3 hours per week, 3 credit hours, prerequisite: 721120 + 210104

Aims:
This is a programming-intensive module where students learn the fundamentals of designing data structures for use in complex programs. Data structures course is an essential area of study for computer scientists and for anyone who will ever undertake any serious programming task. This course deals with the fundamentals of organizing and manipulating data efficiently using clean conceptual models. Students study many of the important conceptual data types, their realization through implementation, and analysis of their efficiency. Implementations in this course are carried out in the Java programming language, but the principles are more generally applicable to most modern programming environments. Topics include recursion, the underlying philosophy of object-oriented programming, fundamental data structures (including stacks, queues, linked lists, hash tables, trees, and graphs), and the basics of algorithmic analysis.

Teaching Methods: 32 hours Lectures (2 per week) + 16 hours Tutorial (1 per week)

Synopsis: Introduction to Software Engineering, Introduction to data structures: data structures and algorithms; Data Design and Implementation; Algorithm complexity; List ADT: static implementation, single linked list; List ADT: dynamic implementation, single linked list; Lists: doubly linked list and circular linked list; Stacks: Static implementation and dynamic implementation; Queues: Static implementation and dynamic implementation, circular queue; Programming with Recursion; Trees: Binary search tree; Trees: binary expression tree, and heap tree; Priority Queues and Heaps; Graph ADT; Sorting: Bubble sort, selection sort, insertion sort, Quick sort, Heap sort; Searching: Sequential search, Binary Search; Hashing: hash function, Separate chaining, open addressing

Modes of Assessment:
Two 1-hour midterm exams (15% each); Coursework (15%); Tutorial Contribution (5%); Final (unseen) Exam (50%)

Textbooks and Supporting Material:
1- Nell Dale, Daniel T. Joyce and Chip Weems, Object-Oriented Data Structures using Java, Jones and Bartlett Publishers, 2001  
4- Mark Allen Weiss, Data Structures and Algorithm Analysis in C++, Addison-Wesley, 1999

----------------------------------------------------------------------------------------------------------------------------------

731150, Introduction to Information Systems

3 hours per week, 3 credit hours, prerequisite: 750112

Aims: This module aims to provide students with some concepts of information systems and some applications in business and management systems. This is a major introductory course presents problems in business environment and solutions with computer-based tools. It focuses on systems and information systems concepts and techniques. Students will learn the most effective ways to use information systems. Case studies are examined to highlight new technology and applications like multimedia.

Teaching Methods: 20 hours Lectures (1-2 hours per week) + 25 hours Class workshop and labs/E-Learning (1-2 per week) + 3 hours Workshops
**Modes of Assessment:** Two midterm exams (15% each); Homework (10%); Workshop Contribution (10%); 2-hours Final Exam (50%).

**Textbooks and reference books:**
4- Leonard M. Jessup and Josef S. Valacich, Information Systems Foundations, 1999, Que E&T

---

### 3.3 Intermediate Modules

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

<table>
<thead>
<tr>
<th>Module Number</th>
<th>Module Title</th>
<th>Prerequisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>750332</td>
<td>Computer Architecture</td>
<td>750231</td>
</tr>
<tr>
<td>750333</td>
<td>Principles of Operating Systems</td>
<td>750332</td>
</tr>
<tr>
<td>721220</td>
<td>Object Oriented Programming</td>
<td>750114</td>
</tr>
<tr>
<td>721240</td>
<td>Computing Ethics</td>
<td>731150</td>
</tr>
<tr>
<td>721222</td>
<td>Software Modelling</td>
<td>721110</td>
</tr>
<tr>
<td>721221</td>
<td>Object Oriented Data Structures</td>
<td>721220</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 250104</td>
</tr>
<tr>
<td>761220</td>
<td>Visual Programming</td>
<td>721220</td>
</tr>
<tr>
<td>721230</td>
<td>Software Requirements</td>
<td>721110</td>
</tr>
<tr>
<td>721322</td>
<td>Software Design</td>
<td>721230 + 721320</td>
</tr>
<tr>
<td>721320</td>
<td>Software Architecture</td>
<td>721222</td>
</tr>
<tr>
<td>750322</td>
<td>Design and Analysis of Algorithms</td>
<td>721221</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 250231</td>
</tr>
</tbody>
</table>

**721220, Object-Oriented Programming**

**Course Hours:** 4 hours per week (1 hour for practice), 3 credit hours

**Prerequisite:** 750114

**Teaching Method:** 30 hours lectures (2 hours per week) + 15 hours Tutorials (1 per week) + 15 hours Laboratory (1 per week)
Aims: The module 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 paradigm. 

Synopsis: Introduction: Classes, Objects, Methods, and Properties, A method deeper Look: static Methods, static Variables, Scope of Declarations, and Method Overloading, Declaring and Creating Arrays, Array of Object, and Generic Collection, Classes and Objects A Deeper Look: Data Abstraction and Encapsulation, Controlling Access to Members, static Class Members, Referring to the Current Object’s, Overloaded, Constructors, and Composition, Inheritance : Base Classes and Derived Classes, protected Members, Relationship between Base Classes and Derived Classes, Constructors in Derived Classes, and Class object, Polymorphism: Polymorphic Behavior, Abstract Classes and Methods, Using Interfaces, Inheritance : Base Classes and Derived Classes, protected Members, Relationship between Base Classes and Derived Classes, Constructors in Derived Classes, and Class object, Polymorphism: Polymorphic Behavior, Abstract Classes and Methods, Using Interfaces, Exception Handling.

Assessment: Two midterm exams (20% each); Laboratory (20%); Tutorial contribution (20%); Final exam (40%).

Textbooks:

721221, Object-Oriented Data Structures

Course Hours: 4 hours per week (1 hour for practice), 3 credit hours

Prerequisite: 721220, 250104

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

Aims: This is a programming-intensive module where students learn the fundamentals of designing data structures for use in complex programs. Data structures module is an essential area of study for computer scientists and for anyone who will ever undertake any serious programming task. This module deals with the fundamentals of organizing and manipulating data efficiently using clean conceptual models. Students study many of the important conceptual data types, their
realization through implementation, and analysis of their efficiency. Implementations in this module are carried out in the Java programming language, but the principles are more generally applicable to most modern programming environments.

Topics include recursion, the underlying philosophy of object-oriented programming, fundamental data structures (including stacks, queues, linked lists, hash tables, trees, and graphs), and the basics of algorithmic analysis.

**Synopsis:** Data Design and Implementation, Abstract Data Types, Lists, Stacks and Queues, Linked Lists, Programming with Recursion, Binary Search Trees, Priority Queues and Heaps, Graphs, Introduction to Algorithms analysis, Sorting and Search Algorithms.

**Assessment:** Two 1-hour midterm exams (20% each); Assignments (20%); 2-hours final exam (40%).

**Textbooks:**

---

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.


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

2. Software Architecture in Practice
   Author(s)/Editor(s): Len Bass, Paul Clements and Rick Kazman
   Publisher: Addison-Wesley, 2007
721322, Software 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

750232, Computer Organization and Architecture

3 hours per week, 3 credit hours, prerequisite: 731150

Aims: The module will emphasize on the following knowledge areas: assembly level machine organization, memory system organization and architecture, interfacing and communication, functional organization, and alternative architectures.

Teaching Method: 32 hours Lectures (2 per week) + 12 hours Tutorials (0-1 per week) + 4 hours Seminars/Presentations

Synopsis: Review of Basic Computer Architecture and Microprocessors; Von Neumann architecture: principles, instruction sets, instruction format, addressing modes, assembly/machine language programming,
CISC versus RISC architectures, subroutine call and return mechanism; Control unit: hardwired, micro-programmed; Storage system and their technology: memory hierarchy, main memory organization and operations, cycle time, bandwidth and interleaving; cache memory: addressing mapping, block size, replacement and store policy; virtual memory: page table, TLB; I/O fundamentals: handshaking, buffering, programmed I/O, interrupts-driven I/O; Buses: types, bus protocols, arbitration, Direct Access Memory; Pipelining: principles, Instruction pipelines, Pipelines difficulties and solutions; Introduction to SIMD, MIMD.

**Modes of Assessment:**
Two midterm exams (15% each); Course work (10%); Seminars (5%); Tutorial Contribution (5%); Final Exam (50%)

**Textbook and Supporting Material:**

---

**750333, Principles of Operating Systems**

3 hours per week, 3 credit hours, prerequisite: 750232

**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)

**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 (15% each); Assignments (10%); Lab work (5%); Tutorial contribution (5%); 2-hours Final Examination (50%)

**Textbooks and Supporting Material:**

721240, Computing Ethics

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

Prerequisite: 731110

Level: 2

Aims: This module aims to give students an informed awareness of the principal issues of professional ethics and responsibility (ergonomics and ethics) in the analysis, design, implementation and use of computers, information systems and Information Technology (IT) products. This will help students in recognition of ethical problems when they occur. Also it will enable students to deal effectively with ethical, social and professional issues now and in their future careers.

Teaching Methods: 36 hours Lectures (2-3 per week) + 9 hours Projects (class work) (average 1 per week) + 3 hours Seminars (1 per month)

Synopsis: Introduction to Ethics; Professional and Professionalism; Code of Ethics and Social Issues; Computer/IT professionals; Computer Security; Privacy and Internet Issues; Information Systems and Ethics; Associations of IT professionals; Ethics and the Internet; Ethical Challenges of e-Business; Ethical Challenges of e-Business; Continuous Professional Development; Intellectual Property Rights; Jordanian Codes for Intellectual Property Rights; Seminars and Project Discussion.

Modes of Assessment: Two 1-hour midterm exams (20% each); Assessment by individual essay (10%); Workshop Assessment (10%); 2 hour written final Exam (40%)

Textbooks and Supporting Material:
5. مجموعة تشريعات الملكية الفكرية الأردنية

Website(s):
ACM, IEEE and BCS Web Sites.
www.cyberethics.cbi.msstste.edu
www.aiitp.org
www.acm.org
www.prenhall.com
www.jcs.rg.jo
210231, 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:

761340, Fundamentals of Computer and Telecommunication Networks

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

Aims: This module aims to introduce the principles of telecommunication networks (Fixed and Mobile) Architectures and Switching Technologies.

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

Synopsis: Network introduction, LAN, WAN, and Telecommunication Networks (Fixed and Mobile) Architectures; Switching Technologies: Layers, Services and Protocols concepts; IP technology.

Assessment: Two 1-hour midterm exams (15% each) + Assignments (15%) + Tutorial contributions (5%) + 2-hours final exam (50%).

Textbooks:
Johan Zuidweg, Next Generation Intelligent Networks, Artech House, ISBN
721230, Software Requirements

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

Level: 2

Prerequisite: 721110

Teaching Methods: Lectures (30 hours) (2 hours per week), Tutorials (15 hours) (1 per week).

Aims: The aim of this module is to be able to systematically establish, define, and manage the requirements for a computer-based system. The principal problem area in software development and production are the requirements specification and the management of customer requirements. Improving the processes of discovering, documenting, and managing system requirements is critical for future business success. This course is intended to address Requirements Engineering, which is the term used to cover all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system. The course will focus on techniques for eliciting requirements, In-depth study of methods, tools, notations, and validation techniques for the analysis, specification and management of software requirements.

Synopsis: Basic concepts and principles of software requirements engineering, its tools and techniques, requirements inception and elicitation, and requirements analysis and specification - modeling techniques, methods for modeling software systems; various approaches to requirements analysis are examined: requirements verification, and validation, object-oriented, and formal approaches, requirements management.

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


- Also recommended:
  - Ian Alexander and Ljerka Beus-Dukic, Discovering Requirements: How to Specify Products and Services, Wiley, 2010,
760261, Database Fundamentals

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

Aims: This module aims to provide students with an overview of database management system architectures and environments, an understanding of basic database design and implementation techniques, and practical experience of designing and building a relational database. The other aim of this module is to make students able to discuss/explain the importance of data, the difference between file management and databases. In addition, it enables students to apply conceptual design methodologies for a database and learn about architectures and environment of database management system (in particular the Ansi-Sparc model). This module requires a practical work, which is assessed by producing individual and group small projects.

Teaching Method: 30 hours lectures (2 hours per week) + 15 hours Laboratory (1 per week).

Synopsis: General introduction and database systems. Architectures: Ansi-Sparc model of databases, components of a database management system, DBMS functions schemas, levels of abstraction and mappings, role of the data dictionary, client-server systems, PC based systems, database servers, distributed systems. General database design: Design framework, mappings between abstractions, integrity, compromises, data vs functional design, non-functional considerations e.g. performance, volumes, user interface etc, security. Conceptual design: Requirement for conceptual design, Extended Entity Relationship model, object-oriented design. Logical design: The relational model, normalization, relational algebra, SQL, mapping conceptual design to relational, integrity, views, embedded SQL, PL/SQL, triggers. Relational databases: Mapping conceptual schema to a relational schema; entity and referential integrity; relational algebra and relational calculus. Database query languages: Overview of database languages; SQL; query optimization. Relational database design: Database design; functional dependency; normal forms; multivalued dependency; join dependency; representation theory. Physical design: Clustering, indexes, performance considerations. Transaction processing: Transactions, Concurrency techniques (locking, 2-phase locking, serialisability), recovery (rollback and commit, 2-phase commit), Transaction Processing Management Systems. Introduction to distributed databases: Distributed data storage; distributed query processing; distributed transaction model; concurrency control; homogeneous and heterogeneous solutions; client-server. Physical database design: Storage and file structure; indexed files; hashed files; signature files; b-trees; files with dense index; files with variable length records; database efficiency and tuning.

Assessment: Two 1-hour midterm exams (15% each) + Labwork and Assignments (20%) + 2-hours Final Exam (50%).

Textbooks:
2- C. J. Date, An Introduction to Database Systems,
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>
<thead>
<tr>
<th>Module Number</th>
<th>Module Title</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>721420</td>
<td>Software Construction and Evolution</td>
<td>721322</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 721324</td>
</tr>
<tr>
<td>721430</td>
<td>Software Testing</td>
<td>721322</td>
</tr>
<tr>
<td>721421</td>
<td>Software Re-Engineering</td>
<td>721420</td>
</tr>
<tr>
<td>721422</td>
<td>Web Software Engineering</td>
<td>721420</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 731213</td>
</tr>
<tr>
<td>721331</td>
<td>Software Project Management</td>
<td>721330</td>
</tr>
<tr>
<td>721330</td>
<td>Software Production</td>
<td>721222</td>
</tr>
<tr>
<td>721324</td>
<td>Advanced Object Oriented Programming</td>
<td>721220</td>
</tr>
<tr>
<td>721423</td>
<td>Graphical User Interface Design</td>
<td>721320</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 761220</td>
</tr>
<tr>
<td>721438</td>
<td>Practical Training</td>
<td>90h +Department</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Agreement</td>
</tr>
<tr>
<td>721448</td>
<td>Research Project 1</td>
<td>90h +Department</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Agreement</td>
</tr>
<tr>
<td>721449</td>
<td>Research Project 2</td>
<td>721448</td>
</tr>
</tbody>
</table>

721420, Software Construction and Development

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

*Level*: 4

*Prerequisites*: 721322 + 721324

*Teaching Methods*: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

*Aims:*
This course presents general principles and techniques for disciplined low-level software coding: mapping design outcomes into code, Selection of appropriated programming language to specific application, code development with errors avoidance techniques, code development with errors tolerance techniques, code development with API, and code development environments programming and GUI builders. An introduction to code testing will close the course This course aims to:
- map design models into source code
- develop code with errors avoidance techniques
- Develop parallel and distributed code
Synopsis: Steps of building a routine, Component-based software construction based on a given architecture, Mapping design outcomes into Code/ derivation of code from design models, Application of Object Model and Database Model, Error handling, exception handling, defensive programming, Error avoidance→ Vulnerable techniques: Program Flow Control breaking (goto, continue, exit, break, …), Recursion, Global data, parameters by value and by references, Construction tools technology/techniques and tools supporting code development, API use, code reuse and libraries, distributed Middleware tools.

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

Textbooks:

0721430, Software Testing

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

Level: 4

Prerequisites: 0721322

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This course will address topics in the software testing addresses issues related to whether the system is correct (with respect to some specification) and the needed software testing techniques to ensure that a system is free of faults and of high quality. Topics include test planning and management, testing tools, technical reviews, formals methods and the economics of software testing. The relationship of testing to other quality assurance activities as well as the integration of verification and validation into the overall software development process are also discussed.

Synopsis: Review of software engineering methods and challenges, the role of verification and validation, the economics of verification and validation, Software reviews and inspections, Software quality metrics, Software testing overview, Functional & Structural testing, Validation based testing techniques, Fault based testing techniques, Boundary value testing, equivalent class testing, syntax testing, path testing, Integration and system testing, Object-oriented testing, Assessing software quality, Testing component based systems such as CORBA and Java beans. Testing tools such as JUnit.

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

Textbooks:
721421, Software Re-Engineering

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

Level: 4

Prerequisite: 721420

Aims: This module will focus on enabling software evolution and maintenance through reengineering.

Teaching Method: Lectures: 36 hours, Tutorials: 12 hours

Synopsis: software evolution, software maintenance; software reengineering, reverse engineering, code refactoring, code slicing, code migration, program comprehension, program transformation, data reverse engineering, Object Oriented Reengineering.

Assessment: Two 1-hour midterm exams (20% each); Tutorial contribution (5%); Project work (15%); 2-hours Final Exam (40%).

Textbook:
1. A. Afshar Alam, Tendai Padenga, Application Software Reengineering, Pearson Education India, 2010
2. Salvatore Valenti, Successful Software Reengineering, IRM Press, 2002

0721422, Web Software Engineering

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

Level: 4

Prerequisites: 721420 + 0731213

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This course aims at applying the software engineering development processes with the Web applications; it introduces a structured methodology utilized in software engineering to Web development projects.

Synopsis:

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

Textbooks:

0721331, Software Project Management

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

Level: 3

Prerequisites: 0721330

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims:
Software management is concerned with knowledge about the planning, organization, and monitoring of all software life-cycle phases. Management is critical to ensure that: software development projects are appropriate to an organization, work in different organizational units is coordinated, software versions and configurations are maintained, resources are available when necessary, project work is divided appropriately, communication is facilitated, and progress is accurately charted.

Synopsis:
Software Process overview, Management related concepts, Project planning methods such as step wise, Project personnel and organization methods, Project control methods, Software configuration management, risk management methods, and software quality.

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

Textbooks:
2. R.S. Pressman. Software Engineering, a practitioner's approach, 8e, 2010
721330, Software Production

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

**Level:** 3

**Prerequisites:** 721222

**Teaching Methods:** Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

**Aims:**
- Description of commonly used software life cycle process models and the content of institutional process standards,
- Definition, implementation, measurement, management, change and improvement of software process, and
- Use of a defined process to perform the technical and managerial activities needed for software development and maintenance.

**Synopsis:**
- Definition of the production related terms such as process, activity, task, etc. Traditional Process Models such as waterfall, Prototyping, Spiral Model, The concurrent development model, Rational Unified Process (RUP), An Agile Process, Extreme Programming (XP), Adaptive Software Development (ASD), Dynamic Systems, Development Method (DSDM), Software Process Analysis Approaches, Object Oriented Development case study, Service Oriented Development, Software Quality, Product Metrics.

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

**Textbooks:**

721324, Advanced Object Oriented Programming

**Course Hours:** 3 hours per week, 3 credit hours

**Prerequisite:** 721220

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

**Aims:**
- The course aims to: to define or explain principles of modularity, encapsulation, information hiding, abstraction, and polymorphism, to use frameworks, classes, and methods from standard libraries in problem solutions, to Use object-oriented design to model problem solutions and
express inheritance, association, aggregation, and composition relationships among classes, to recognize and use basic object-oriented design patterns to structure solutions to problems, to design, implement, and use classes and methods in an object-oriented programming language, employing standard naming conventions and making appropriate use of advanced features such as inheritance, exception handling. (Python for instance)

**Synopsis:** Cornerstones of Computing, Getting Started in Python, Elementary Control Structures, Additional Control Structures, Defining Our Own Classes, Good Software Practices, Input, Output, and Files, Inheritance, Deeper Understanding of the Management of Objects, More Python Containers, Implementing Data Structures, **A case study**, and Event-Driven Programming.

**Assessment:** Two midterm exams (20% each); Laboratory (20%); Tutorial contribution (20%); Final exam (40%).

**Textbooks:**

------------------------------------------------------------------------------------------------------------------------

**721423, Graphical User Interface Design**

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

**Level:** 4

**Prerequisites:** 721320 + 761220

**Teaching Methods:** Lectures (33 hours), Laboratory (15 hours)

**Aims:** HCI (human-computer interaction) is the study of how people interact with computers. It is concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of environment surrounding them. The interaction with the computer systems are done through GUI (Graphical User Interfaces). In order to design, develop and implementation of good interfaces, the knowledge of human-computer interaction principles and GUI programming skills are required. The course aims to provide students with the principles for predicting the usability of human computer interaction, and developing systematic methodologies for design and evaluating them, and improve students' awareness of the issues that determine the usability of an interactive computer system.

**Synopsis:** This course provides an introduction to the study of human-computer interaction and user interface design. HCI is a field that combines material from many different perspectives including computer science, psychology, human factors and graphic design. This course provides also an overview of established and emerging theories, conceptual frameworks and methods of the human aspects of HCI;
Knowledge representation and organisation, mental models, the utility of mental models in HCI, verbal metaphors, virtual interface metaphors, classification of interface metaphors for applications, conceptual models; Technology Aspects: Introducing a range of input and output devices and interaction styles, and discussing some higher level system design issues; Design Practice: Discussing the most popular design and evaluation methods and design support tools that are available to make HCI design user-centered, including principles and methods for user centered design, requirement gathering, task analysis and structured HCI design. Screen Design: An advanced topic that covers a theoretical model to support screen design. Hypertext, Multimedia and the World Wide Web: Covering major research issues in multimedia and the Web. GUI programming: Introduction to GUI, Java review exercises, GUI Components - Swing, Event processing, Mouse Events, Keyboard Events, Window Events.

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

Textbooks:

721440, 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)**

*Course Hours:* 1 credit hour (16 hours)

*Level:* 4

*Prerequisites:* 90 hours + Department agreement

*Aims:* The aims for the project work done in the fourth year are: to manage and execute a substantial project in a limited time, to identify and learn whatever new skills are needed to complete the project, to apply design and engineering skills in the accomplishment of a single task. In this context the skills mentioned may be in the general area of design and engineering in its broadest sense, or may be very specifically related to particular tools. A student works under the supervision of a member of staff, the Supervisor. Most of the projects involve three students working together on the same project; apart from these, all students do different projects. The research project consists of a single project on which the student works over a period of two consecutive semester courses.

At the end of the first research project course students are expected to deliver: SRS (Software Requirement Specification) document, executable prototype, software architecture of their project.

*Synopsis:* Review of Software life cycle, Requirement elicitation, requirement modelling and analysis, SRS document, prototype, software architecture (style, sub-systems, components...)

*Assessment:* SRS document (60%), prototype (20%), Software architecture (20%)

*Textbooks:*
- Department Research Project Guide

**721449, Research Project (2)**

*Course Hours:* 2 credit hours (32 hours)

*Level:* 4

*Prerequisites:* 721448

*Aims:* The aims for the project work done in the fourth year are: to manage and execute a substantial project in a limited time, to identify and learn whatever new skills are needed to complete the project, to apply design and engineering skills in the accomplishment of a single task. In this context the skills mentioned may be in the general area of design and engineering in its broadest sense, or may be very specifically related to particular tools. A student works under the supervision of a member of staff, the Supervisor. Most of the projects involve three students working together on the same project; apart from these, all students do different projects.

The research project consists of a single project on which the student works over a period of two consecutive semester courses.

At the end of the second research project course students are expected to deliver: project design document, project source code, project tests, and final project document.

*Synopsis:* Review of software design methods, Software coding techniques, software testing techniques

*Assessment:* software design document (30%), coding (60 %), software testing (10%)

*Textbooks:*
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>
<thead>
<tr>
<th>Module Number</th>
<th>Module Title</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>721445</td>
<td>CASE Tools Development</td>
<td>721420</td>
</tr>
<tr>
<td>721439</td>
<td>Special Topics In Software Engineering</td>
<td>721322</td>
</tr>
<tr>
<td>721443</td>
<td>Telecommunication Software Design</td>
<td>721322</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 761340</td>
</tr>
<tr>
<td>721447</td>
<td>Advanced Modeling Techniques</td>
<td>721420</td>
</tr>
</tbody>
</table>

721445, CASE Tools Development

3 hours per week, 3 credit hours, prerequisite: 721420

**Aims:** This module focuses on techniques used for the development of Computer Aided Software Engineering Tools: Analysis tools, Projects management tools, Configuration Management tools, Code generation.

**Teaching Method:** 30 hours Lectures (2 hours per week) + 8 hours Seminars (1 per week) + 7 hours tutorials


**Assessment:** Two 1-hour midterm exams (20% each); Tutorial contribution (5%); Project work (15%); 2-hours Final Exam (40%).

**Textbooks:**
1- Ian Somerville, Software Engineering, Addison-Wesley, 2000
2- M.J. Pont, Software Engineering with C++ & CASE Tools, Addison-Wesley, 1996

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.

---

**721443, Telecommunication Software Design**

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

**Level:** 4

**Prerequisites:** 721322 + 761340

**Teaching Methods:** Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

**Aims:** Telecommunications are becoming a dynamic and a core field of information technology. This course is intended to address software engineering issues in the context of the telecommunication field.

The course will focus on methods, techniques, modeling languages, and tools that are currently used to design and implement communication software, communication protocols, telecommunication services and applications. It includes a survey and review of current telecommunication network technologies.

**Synopsis:** Telecommunication and Mobile network technologies (architecture, protocols, services), Communication software, Protocol engineering, Domain Specific Modeling languages for telecommunications, Design Patterns for telecommunications, Formal methods in telecommunications, Communication software development methods.

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

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

-----------------------------------------------------------------------------------------------------------------------------

721447, Advanced Modeling Techniques

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

Level: 4

Prerequisites: 721420

Teaching Methods: Lectures (34 hours), Tutorials (14 hours)

Aims: Modeling is considered as a core field in the modern software engineering discipline (Model Driven Software Engineering). This course is intended to address advanced issues in software modeling. It copes with advanced topics not covered by the first software modeling course (721222).


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

Textbooks:
1. Modeling software behavior; a craftsman's approach, Jorgensen, Paul C, CRC. Press Online, 2009
APPENDIX B

STUDY PLAN

OF

SOFTWARE ENGINEERING PROGRAMME