Philadelphia University

Faculty of Information Technology

Department of Software Engineering

Undergraduate Handbook

Content

I. Introduction

This handbook contains important general information for students undertaking Undergraduate Degree program in the Department of Software Engineering. This handbook is also available on the web.

Your degree program is subject to regulations contained in the **University Students Guide**. This departmental handbook interprets the regulations and your tutors may give advice, but the University Students Guide defines the regulations.

II. Mission Statement

The mission of The Software Engineering Department is derived from the overall IT Faculty and University mission. The Software Engineering Department at Philadelphia University was founded in the year 2000 as one of the first Software Engineering Departments offering honor degree in Software Engineering in Jordan. This undergraduate program addresses the growing need for professionals in this sophisticated field.

The mission of the Software Engineering Department at Philadelphia University is to provide outstanding education to its undergraduate students in accordance with the principles of the University mission, to advance scholarship in key domains of software engineering, and to engage in activities that improve the welfare of society. The Department aims to maintain an environment that promotes innovative thinking; values mutual respect and diversity; encourages and supports scholarship; instills ethical behavior; and engenders life-long learning

The strategies of the Department are set to meet the demands of a rapidly evolving world, and to meet the needs of a developing job market in Information Technology. Graduates of this program will work with the engineering of software, with special attention devoted to large and critical systems. This program addresses both analytic and practical skills required by students to develop robust and efficient computer software systems for manufacturing, industrial, medical, government, and business applications. They will have individual and team hands-on experience with timely, cost-effective and state-of-the-art processes, methods and tools.
The curriculum of this program aims to prepare students for careers in software engineering, software project management, and software development and integration. Software engineering comprises the core principles consistent in software construction and maintenance. This mainly covers the fundamental software processes and life-cycles, mathematical foundations of software engineering, requirements analysis, software engineering methodologies and standard notations, principles of software architecture and reuse, software quality frameworks and validation, software development, and maintenance environments and tools.

III. Important Dates

1. Registration:

Admission criteria are issued by the Higher Education Council, which governs all private universities (55% in the Tawjihi exam). First year students must attend the University and they will be given a full timetable for the introductory activities. Departmental and University registration must be completed at the time specified in the introductory timetable. Returning students must also register in the times specified during introductory week.

2. Timetable

Lectures timetable is published separately from this book. Whilst every attempt is made to timetable reasonable combinations of course units (modules), various constraints make some combinations and outside options impossible. If you have a timetable problem, please consult your personal tutor in the first instance.

IV. Scope and Input Resources

1. Aims and Objectives

Aims: Software Engineering program at Philadelphia University gives you the opportunity to:

- study a body of knowledge relating to Software Engineering, Software reengineering, and maintenance;
- understand the principles of large scale software systems, and the processes that are used to build them;
- have skills in the most widely used approach to software construction – object-orientation (OO), including OO requirement specifications, OO analysis, OO design, OO programming, OO testing and maintenance;
- use tools and techniques for producing application software solutions from informal and semi-formal problem specifications;
- acquire and develop many valuable skills such as the ability to use computer aided software engineering tools to analyze, evaluate, select and synthesise information sources for the purpose of developing a software system;
- develop an appreciation of the cost, quality, and management issues involved in software construction;
- develop an awareness of the role and responsibilities of the professional software engineer;
- acquire skills to think about problems and their solutions using appropriate methods of analysis and design;
- be able to design and communicate ideas about software system solutions at different levels of abstraction and have the opportunity to transfer such skills across a wide range of industrial and commercial domains;
- have an ability to work with other people in a team, communicating computing ideas effectively in speech and in writing;
- have a basis for going on to further study in software engineering, or for finding work in computing-related industries.
- be a graduate that can go on to employment in technical positions in software houses and with large-scale scientific and engineering users;
- be graduate that may seek to pursue research careers.

Objectives (Learning Outcomes). Learning outcomes describe what you should know and be able to do if you make full use of the opportunities for learning that we provide. All these skills are described in the following areas (A, B, C, D). In the individual module syllabi, the categories of learning outcomes (A, B, C, D) and the individual learning outcomes appropriate to the module are identified.

A- Knowledge and Understanding of

A1) the system development lifecycle;
A2) a wide range of principles and tools available to the software developer, such as software process methodologies, choice of algorithm, language, software libraries and user interface technique;
A3) the principles of object-oriented software construction;
A4) the software-development process, including requirements analysis, design, programming, testing and maintenance;
A5) the range of situations in which computer systems are used, the ways in which people interact with them;
A6) professional issues to cover: social, ethical and legal aspects;
A7) communication issues in large, complex software projects;
A8) the principles and techniques of a number of application areas informed by the research directions of the subject, such as software engineering, net-centric, and distributed systems.

B- Intellectual (thinking) skills - able to

B1) model object-oriented software systems;
B2) investigate and improve the specification of a software system;
B3) design and plan software solutions to problems using an object-oriented strategy;
B4) identify a range of solutions and critically evaluate and justify proposed design solutions;
B5) write and test programs using at least one object-oriented programming language;
B6) evaluate systems in terms of general quality attributes and possible trade-offs presented within the given problem;
B7) use and evaluate appropriate tools and techniques
B8) reflect and reason concerning a given information handling problem or opportunity.

C- Practical skills - able to

C1) specify, design and construct CASE tools and application software;
C2) use logic and discrete mathematics to specify software elements;
C3) develop and apply testing strategies for software applications;
C4) develop software applications in a development environment that makes use of commonly supported tools;
C5) identify some of the main risks of software development and use;
C6) use network information services
C7) Prepare and deliver coherent and structured verbal and written technical reports;
C8) use the scientific literature effectively and make discriminating use of Web resources;
C9) analysis of system requirements and the production of system specifications;
C10) use appropriate computer-based design support tools.

D- Transferable skills - able to

D1) effectively participate in team-based activities;
D2) structure and communicate ideas effectively, both orally, in writing, and in cases involving a quantitative dimension;
D3) use IT skills and display mature computer literacy;
D4) work independently and with others;
D5) manage learning and self-development, including time management and the development of organisational skills;
D6) display personal responsibility by working to multiple deadlines in complex activities;
D7) undertake practical training and placements in relevant organisations
D8) appreciate the need for continuing professional development and in recognition of the need for lifelong learning

In order to provide students with the "life long learning" attitude, the teaching method is essentially based on self learning (3 hours in class rooms and 6 hours out of class rooms: coursework, practical works, workshops, seminars, etc.)

2. Staff

A. Academic Staff

- **Qualifications**

The academic staff members are divided into two categories: full-time and part-time. The number of full-time staff members is 10, while the number of part-time staff depends upon the number of students and the needs of the Department.

The academic staff members, who are between 27 and 56 years of age, have relatively adequate experience ranging from 1year to 25 years.

Six academic staff members at the Basic Sciences Department / Faculty of Science assist in teaching the Mathematics and Discrete Structures course units.

- **Specialisations**

  Full-time as well as part-time teaching staff members have various specialisations that can be divided into four categories (Software, Communication and Interaction, Practice, Theory). At present, there are six research teams at the Faculty of IT and young staff members belong to these teams.

B. Non-Academic Staff

Besides the academic staff, the Department has 3 other full time members hold a B.Sc. degree in Computer Science. Those staff members have 3 to 5 years working experience and some of them have been appointed from Philadelphia University graduates who hold bachelor degrees with Grade "Excellent" or "Very Good".

All of the non-academic staff members are qualified as laboratory tutors and assist lecturers in the laboratory hours. In addition, some of them are responsible for maintenance of computer hardware and software in the laboratories.

3. Departmental Learning Resources

- **Code of Practice for Student Computer Usage**

At registration, you will be required to assent to the following departmental code of behavior, which relates to the responsible use of Computer equipment. Misuse of the facilities is regarded as serious disciplinary offences.

This code of practice is supplementary to University regulations concerning the use of computing equipment to which you are required to assent at Registration.

1. Every student is allocated one PC in every laboratory session. But for UNIX laboratory, you have been allocated one or more usernames for your own personal use: you must not use other usernames or permit other people to use your username. You must not use computers to which you have not been granted access, or attempt to access information to which you have not been granted access.
2. You must not deliberately hinder or annoy other computer users.
3. You must not use machines belonging to the Department for commercial purposes without the prior written permission of the Head of Department. You must not sell the results of any work you do using Departmental facilities without the prior written permission of the Head of Department.
4. You must not write or knowingly store, on machines belonging to the Department, software that, if executed, could hinder or annoy other users, except with the prior written permission of the Head of Department.
5. You must not make an unauthorized copy, in any form, of copyright software or data.
6. You must not store personal information, except in a manner permitted by the Data Protection.
7. You must follow all rules, regulations and guidelines imposed by the Faculty of IT and the University in addition to the Department's Code of Practice.

- **Explanatory Notes**

The following notes indicate ways in which the Code of Practice applies to undergraduates for use of computers. It is not intended to be a complete list of possible abuses of the equipment. Each note refers to the corresponding paragraph above.

1. Undergraduate students are not normally granted access to the computers in the network, or to other students' files. You should not attempt to use another student's account even if they have not set a password. Of course, it is still important to set a password for your own privacy and security.
2. This will be interpreted very broadly. It includes
   - Tampering with another user's files.
   - Tampering with another user's screen.
   - Setting up processes which persist after you log out and annoy subsequent users of the machine.
   - Broadcasting of offensive messages.
   - Display or storage of offensive pictures.
   - Abuse of the mail system.

• Occupying a machine to play games while other students need it to do their laboratory work.

3. Clearly, the Head of Department would have to be convinced that any such use of the machines would not conflict with their primary purpose.
4. Note carefully that this means you are not allowed to write or introduce a virus program, even if it is never executed.
5. Note that this does not prevent your taking copies of your laboratory work home, or making copies of non-copyright material, but does prevent your taking random pieces of software away on a floppy. You should assume that all material is copyright unless it specifically states otherwise. If in doubt, ask.
6. Personal information includes names, addresses, mailing lists, etc. You should contact the Data Protection Officer, Mr. Tarek Hassan, if you need to store such information.
7. In fact, you agreed to abide by the University and Faculty rules when you registered. Please direct queries concerning the code of practice to Department Chair.

- **Support for Computer Equipment**

Students are encouraged to own their own machines. Please note, however, that you are NOT REQUIRED to own your own computer. The Department has excellent facilities and undergraduate students are allowed to use the facilities provided in the buildings of the Faculty of Information Technology and the Faculty of Science. Whenever the buildings are open between 08 AM and 07 PM, access is also allowed in this range of time, from Sunday to Thursday during term.

- **Learning Resource Center**

Photocopy facilities are available in the Learning Resource Center, room 103, Tel. 2453. Reference copies of textbooks are available for consultation. Copies of previous weeks' tutorial solutions are also available. The resource center holds non-loan copies of undergraduate textbooks. Lending copies of textbooks are available in the University Library.

- **Photocopying**

Out of the library, photocopy may be done at different Bookshops, on an affordable cost.

- **Printing**

You can take printout (free of charge) in any lab of the Department. Each lab contains at least two printers for this purpose.

- **Departmental Computer Club**

This is organized and run by students. It arranges various activities from time to time. See the notice boards in the Faculty.

- **Administrative Infrastructure**

It is composed of six offices (Dean, 1 Advisory service, Dean Secretary, and Department's Chair, Department Secretary, and Meeting Room).

- **Academic Infrastructure**

It is composed of

- 16 Department classrooms plus some other classrooms shared with other faculties and one lecture theatre equipped with support facilities: computer, data show, overhead projector.
- 3 laboratories (each contains 20 to 25 PCs or Monitors and 1 to 2 printers): Windows NT Laboratories, Internet Laboratories, SunRay1 UNIX Laboratories, and Sun Sparc UNIX Laboratory. The department also shares some other laboratories with other departments.
- 1 Learning Resource Center that contains computers, textbooks and related reference books and journals.
- 19 staff offices where each staff member is supplied with a PC.
- 1 room for staff meeting
- 1 office for the student's guidance and examination committee.

- **Lecture Support Facilities**
  In the Department, there are 4 overhead projectors and 7 data shows used to support modules and seminars presentations.

- **University Computer Centre**
  This centre provides the Department with training and maintenance facilities.

- **Networking Facilities**
  Ethernet*:* The PCs in each laboratory are connected to an Ethernet platform 10/100 Mbps.
  Intranet: All computing facilities of the University are connected to a Gigabit Intranet backbone.
  Internet: The University is connected to the Internet by 2 Mbps lines.

- **Type and Level of Access**
  For communication, computing, or information searching, the Department provides free access to networking facilities at any time for the staff and the students.

- **Library Infrastructure**
  This structure includes the University Main Library, which it provides students and staff members with the required recent text and references books, journals, and CD ROMs. According to its collaboration and co-ordination program, it has relations with more than 120 universities and scientific organisations. It opens from 08 AM to 07 PM. It includes:

  - ***Conventional Library*****,** which contains books and journals. The books room contains more than 1860 different English titles in computing, where more than 12% are edited in years 2000 - 2004. The room of journals contains 30 computing journals that are useful for research and teaching.

  - ***Electronic Library*****,** which contains CD ROMs for the taught programming languages and module support tools. It is connected to approximately 800 universities electronic libraries via the World University Library that is endorsed by the United Nation University. The World University Library has four databases that contain more than 3300 periodicals available online. The online resources in the electronic library include sites that list more than 40000 online books and access to online libraries and encyclopaedias and other databases on the Internet.

  - ***Internet Access Service,*** available in a room containing more than 20 PCs.

- **Bookshops:** contain books, exercises with solutions, solutions to previous examinations and so on.

- **Extracurricular Activities**
  The University provides some entertainment for the students to enrich their talents in their free time. This includes

  - **A Deanship of Student Affairs** that organises the social, cultural, and sport activities for the students in the University. It has also an alumnae office that keeps track of the graduate's information and news.
  - Several spaces for different sports.
  - Several spaces for cultural activities.
  - Several common rooms for meetings, snacks, and cafeterias.
  - Three Internet cafes each one containing 11 PCs.
  - One Students Club.

V. Student Support and Guidance

1. Assistant Dean Office

The Assistant Dean Office (Room IT 604) is mainly for students advisory services. It deals also with all routine undergraduate enquiries. Problems, which cannot be dealt with by the Assistant Dean, will be referred to an appropriate person in the Department or University.

2. Academic Guidance

All new students should have academic (personal) tutors. The new students are grouped into 20 – 30 students groups and each group is assigned to an academic staff member who is their academic tutor. The students remain with the same tutor till their graduation. The tutor deals with all routine undergraduate inquiries, advises for academic registration at the beginning of each semester, and any other raised problems. However, problems, which cannot be dealt with by the tutor, will be referred to the head of the Department, the Dean of the Faculty, or to an appropriate member of academic staff. The academic guidance is available on specified dates in the terms, and any advisory service offered by the Assistant Dean is available daily to all students in the Software Engineering Department (including both Full- and Part-time students).
Time: 11:00 AM to 07:00 PM Sunday to Thursday during term, Venue: Room IT 604 (for all students)

The advisory service offers advice on departmental and University matters and helps with anything that concerns you, whether in your studies, in the Department, in the University or in your life outside the university. Each of the staff in these offices is available with knowledge of the Department and University and who is willing to listen and help with whatever you bring. Note that

- All visits to the advisory service offices are strictly confidential.
- If you have difficulties with material on particular course units you should normally first approach your tutors (or lecturers/project supervisors). You may also consult your tutors on matters that are more general but you can equally well call in at the Assistant Dean Offices.
- If you have health problems, you are welcome to consult an advisor in the Department but may prefer to go directly to your doctor or to the University Clinic.

Feel free to make use of these services at any time on any matter.

3. Students Affair Deanship

Confidential, individual counseling on any matter affecting personal well-being or effectiveness is available at the Philadelphia University Students Affair Deanship. The Deanship sees well over a hundred students a year and gives expert advice on problems such as low motivation, personal decision making, relationships, and anxiety and family difficulties. People there, are willing to help in finding fresh ways of coping with the emotional and personal aspects of problems and seeks to do so in a collaborative, straightforward and empowering way with the individual concerned. Advice is available concerning referral to other services, helping others and dealing with common student problems such as exam anxiety.
The Deanship is open from 8.00 AM to 4.00 PM, from Sunday to Thursday throughout the year and appointments can be made by calling into the office of the Dean of Students affairs. All inquiries will be treated confidentially.

4. Tutoring Arrangements

Some of your course units will have tutorials, where you can discuss topics on a course unit and run through exercises. Usually, the lecturer of the course unit runs the tutorial. There will be an opportunity for you to ask questions on matters you do not understand.
As you have a personal tutor from the beginning of your University life, your tutor is here to help you in your way through University life. He/she will watch your progress and offer help and advice wherever necessary. If you get into difficulties, you should contact your personal tutor or visit the Assistant Dean at the earliest possible opportunity. Do not let things slide until it is difficult to retrieve the situation, especially if you are getting behind with your work. Your personal tutor will also advise on your choice of course units, on departmental or University procedures and will provide references for jobs and other purposes.

Course lecturers are always available to discuss questions or problems with the course unit material. Each lecturer fixes at least six office hours on his timetable, which is fixed on his office door. You can call at these hours. For any reason, if these lecturers could not see you at these office hours, they may arrange an appointment at another time. It is important that any matter that affects your ability to work is notified to the Department - through your personal tutor, through the Assistant Dean or otherwise. The following are examples of matters that may affect your work: illness, personal or family difficulties (including illness in the family) or financial problems. In assessing your performance, the Department has a policy of trying to compensate for difficulties you have encountered whilst studying. We can only do this if we are notified of difficulties and have some idea of their extent.

5. Student Progress

Work and Attendance. The University regulations governing the Work and Attendance of students are given in the Student Guide 2001/2002. Full attendance is required at all lectures, laboratories, and any tutorials, which may be scheduled. Completed laboratory work should be handed in on time. Attendance at laboratories and at many lectures is monitored and attendance registers kept. Please note that the expectation is that students will be required to undertake approximately thirty six hours per week of study i.e. an average of two hours private study will be required for every scheduled hour of lectures, laboratories etc. and some students may require much more time than this. ***Being a student is a full time occupation!*** Absence for holidays is not permitted in term-time. The experience of the Department confirms that lack of attendance leads to study problems and any student with problems should consult his/her subject tutors or personal tutor. In addition, failure to attend can result ultimately in refusal by the University to allow a student to sit in the degree examinations. The duty of the lecturer is to keep continuous review of the work and attendance of the students with whom he is concerned. If the rate of student absences, in a course unit, is greater than 15% (or 20% for student representing the University in sportive or cultural activities) of the completely accredited hours and the student has no acceptable justification, then this student is excluded from that course unit. If the Dean of the faculty accepts the justifications of absence, then this student is mentioned as ***withdrawn*** without refunding the registration fees. A formal process is defined to tackle the problem of any student whose work and attendance appear unsatisfactory. Direct approaches by lecturer to solve the problem are as follows: He may choose to issue an "informal" warning, which has a precisely defined format and permits recovery of the situation. If this is unsatisfactory, a "formal" warning is issued. This is again of a precisely defined format. Failure to recover the situation at this stage leads to an exclusion from the course. A copy of this correspondence is held in a student's file.

6. Interruption of Degree Program

Any interruption (taking at most 2 years) of your degree program requires special permission from Faculty. Regulations state that a B.Sc. degree is a continuous 4-year period of study. Permission will only be granted if satisfactory reasons are given. A written case with supporting evidence must be presented to Faculty. Reasons might include prolonged illness. Consult your tutor for advice.

7. Transfer between Departments

- If you are contemplating any change of Faculty or Department, consult your primary tutor as soon as possible.
- You can change your Department by filling a special form at the beginning of the semester. It is only required that the Tawjihi average imposed in the new faculty or department must be less than or equal to your Tawjihi average. A specialized committee will decide what courses will be retained from your actual Department.

8. Withdrawal from Modules

If you are contemplating withdrawing from a module, please discuss the situation with your personal tutor at the earliest opportunity.

- You can withdraw a module at most during the thirteenth week of the first or second term, and at most during the seventh week of the summer term.
- The minimal number of modules (which is 9) required in each term should be followed.

VI. Organization of Teaching

An individual course of lectures is known as a "**course unit**" or sometimes as a "**module**".
The curriculum contains modules that are from University Requirements (Univ. Reqts.), Faculty Requirements (Fac. Reqts.), and Department Requirements (Dept. Reqts.). Each module has 3 credit hours per week. However, some modules are supported by tutorials and some continuous assessment, such as seminars or laboratory work, usually amounting to 1 hour per week. When you register for course units, you should follow the academic guidance plan that the Department arranges for you. In fact, you can register on any module only if you have taken its prerequisite(s) with the exception that you can register on the module and its prerequisite only if you are in the graduation semester.
In each semester, you can register for at least 12 credit hours and at most 18 credit hours, except for the semester in which you are expected to graduate when you can register for 21 hours. The complete four years academic guidance plan is listed in **Appendix A** of this Handbook. For more information about module numbering and outline module descriptions, see **Appendix B** of this Handbook.

In the **First Year**, you are encouraged to take 18 credit hours in each semester (first and second, the summer term is not taken into account). The fourth digit of each course unit code (see **Appendix B**) tells you the year in which the course is offered. During each 16 weeks semester, students will normally attend 6 modules. Thus, each teaching week contains 18 hours or more of scheduled work. In addition, each scheduled hour typically requires two extra hours of unscheduled work (e.g. writing up lecture notes, preparing for a tutorial, finishing off a laboratory exercise etc.). The selection of a University elective module (one module) depends upon your choice. **Five** of the 12 modules of the first year are from the University requirements, **two** from the Faculty requirements, **three** from the supportive requirements, and **two** from the Department requirements.

In the **Second Year,** the number and size of modules is similar to that of the first year. **Two** of the 12 modules of the second year are from the University requirements, **three** from the Faculty requirements, and **seven** from the Department requirements.

Meanwhile, in the **Third Year,** you should take six modules in the first semster and five modules in the second semester. **Eight** modules are from the compulsory Department Requirements, **one** module from the University requirements and **two** modules form the Faculty requirements. One of the compulsory modules is the **Practical Training module**, which consists of realizing a supervised training in an industrial organization, or using distance online training. You should take this module in the first semester.

In the **Fourth Year,** you should take nine modules in this year. In the first semester, you must select **one** departmental elective module, **three** compulsory modules that are all from the Department requirements, and **one** module from the Fcaulty requirements. In the second semester, you must take the Graduation Project module, **one** departmental elective module, **one** University elective module, and **one** free module from any department in the University.

VII. Course Unit Choices

You may choose a course unit (module) if you have already taken all its prerequisite modules and your personal tutor must supervise this choice.

An initial choice is made before or at Departmental Registration. After that, changes can be made as follows:

- The deadline for changing modules in each semester is one week after lectures start (three days for summer semester). Normally, no changes of modules will be permitted after these dates except for the withdrawal mentioned in point (8) of the previous section.

- In the first instance, you should discuss any plan to change modules with your primary tutor. You must check that the new module you wish to take is a valid option for your degree program and find out if there are likely to be any timetable problems. If there are timetable clashes this will probably prevent you from changing module.

VIII. Assessment and Examinations

1. Criteria for Assessing Examination Work

First class (90 – 100 marks). First class answers demonstrate depth of knowledge or problem solving skills, which is beyond that expected from a careful and conscientious understanding of the lecture material. Answers will show that the student

1. has a comprehensive knowledge of a topic (often beyond that covered directly in the program) with an absence of misunderstandings;
2. is able to apply critical analysis and evaluation;
3. can solve unfamiliar problems not drawn directly from lecture material and can adjust problem solving procedures as appropriate to the problem;
4. can set out reasoning and explanation in a logical, incisive and literate style.

Upper Second class (80 – 89 marks). Upper second class answers provide a clear impression of competence and show that the student

1. has a good knowledge base and understanding of all the principal subject matter in the program;
2. can solve familiar problems with ease and can make progress towards the solution of unfamiliar problems;
3. can set out reasoning and explanation in a clear and coherent manner.

Lower Second class (70 – 79 marks). Lower second class answers will address a reasonable part of the question with reasonable competence but may be partially incomplete or incorrect. The answer will provide evidence that the student:

- has a satisfactory knowledge and understanding of the principal subject matter of the program but limited to lecture material and with some errors and omissions;
- can solve familiar problems through application of standard procedures;
- can set out reasoning and explanation which, whilst lacking in directness and clarity of presentation can nevertheless be followed and readily understood.

Third Class (60 – 69 marks). Third class answers will demonstrate some relevant knowledge but may fail to answer the question directly and/or contain significant omissions or incorrect material. Nevertheless, the answer will provide evidence that the student

- has some basic knowledge and a limited understanding of the key aspects of the lecture material;
- can attempt to solve familiar problems albeit inefficiently and with limited success.

Pass (50 – 59 marks). Answers in this category represent the very minimum acceptable standard. Such answers will contain very little appropriate material, major omissions and will be poorly presented lacking in any coherent argument or understanding. However the answer will suggest that the student

- has some familiarity with the general subject area;
- whilst unable to solve problems can at least formulate a problem from information given in a sensible manner.

2. Assessment Regulations

In general, every module is assessed as follows: 50% is given for two 1-hour midterm exams, coursework and/or seminars, projects, or essays, and 50% for the final exam that may be a written exam only or a written exam plus final laboratory exam (if applicable), final small project, or seminar presentation. The 50% of the final exam is from the University regulations. The minimum pass mark is 50% for any module, whereas the minimum passing accumulated average in each semester is 60%. Students will be warned if they could not obtain average of at least 60%. In this case, students are encouraged to repeat studying those modules with low marks in order to increase their accumulated averages. However, students will be dismissed from the University if this average is not achieved in the third attempt.

For the practical training module, each student should submit a technical report of his/her training, and a team of academic staff members makes several observations on the trainers' work in their place of training. Then according to the observations and the report, they assess the students.

On the other hand, a committee of three staff members, including the supervisor of the project, assesses the graduation project module. The project's assessment includes the supervisor mark (35%) and the

discussion committee mark (65% given as follows: 20% for project presentation, 25% for report writing, and 20% for defendant discussion).

3. Role of Internal and External Examiners

For each module, the Department assigns a module coordinator and an internal examiner who is one of the senior staff members. If many lecturers teach the same module concurrently, they should suggest exam questions (for the first, second and final exams) and run the same exam for all sections. The main coordinator of the module will collect these questions from lecturers and select some of them to be in the exam paper.

On the other hand, external examiners validate the standard of degree program. The external examiners are expected to look at the question papers, inspect a selection of scripts and project reports (particularly those on borderlines). They supply an assessment report to the Department.

4. Appeal Procedures

If you have good reason to question a mark you have been given (in midterm exams or in coursework), you should in the first instance approach the module lecturer. If the problem is not solved, you must submit it to your primary tutor. He will find the appropriate solution with administrative structures.

Problems with final examinations are resolved by submitting complaints or appeals in writing (within three days of the announcement of examination results) to the Examination Committee of the Faculty. The examination committee will consider these cases and checks if there is any mistake in the summation of the marks and so on.

5. Unfair Practices

The University treats attempting to cheat in examinations severely. The penalty is usually more severe than a zero in the paper concerned. More than one student of this Department were dismissed from the University because of this. Plagiarism, or copying of course or lab work, is also a serious academic offense as explained in the University guidelines. In Software Engineering Department these guidelines apply also to laboratory exercises.

6. Department Guidelines on Plagiarism

1. Coursework, laboratory exercises reports and essays submitted for assessment must be your own work, unless in the case of group projects a joint effort is expected and is indicated as such.
2. Unacknowledged direct copying from the work of another person, or the close paraphrasing of somebody else's work, is called plagiarism and is a serious offence, equated with cheating in examinations. This applies to copying both from other students' work and from published sources such as books, reports or journal articles.
3. Use of quotations or data from the work of others is entirely acceptable, and is often very valuable provided that the source of the quotation or data is given. Failure to provide a source or put quotation marks around material that is taken from elsewhere gives the appearance that the comments are ostensibly your own. When quoting word-for-word from the work of another person quotation marks or indenting (setting the quotation in from the margin) must be used and the source of the quoted material must be acknowledged.
4. Paraphrasing, when the original statement is still identifiable and has no acknowledgement, is plagiarism. A close paraphrase of another person's work must have an acknowledgement to the source. It is not acceptable for you to put together unacknowledged passages from the same or from different sources linking these together with a few words or sentences of your own and changing a few words from the original text: this is regarded as over-dependence on other sources, which is a form of plagiarism.
5. Direct quotations from an earlier piece of your own work, if not attributed, suggest that your work is original, when in fact it is not. The direct copying of one's own writings qualifies as plagiarism if the fact that the work has been or is to be presented elsewhere is not acknowledged.
6. Sources of quotations used should be listed in full in a bibliography at the end of your piece of work.
7. Plagiarism is a serious offence and will always result in imposition of a penalty. In deciding upon the penalty the Department will take into account factors such as the year of study, the extend and proportion of the work that has been plagiarized and the apparent intent of the student. The penalties that can be imposed range from a minimum of a zero mark for the work (without allowing resubmission) through caution to disciplinary measures (such as suspension or expulsion).

IX. Teaching Quality Assurance Committee

The Departmental Teaching Quality Assurance and Enhancement Committee is responsible for the quality of teaching in the Department, including the analysis of Course Evaluation Questionnaire responses.

X. Students Feedback and Representation

1. **Staff Student Consultative Committees**
   Student representatives are elected onto the departmental staff student committees at the start of each term. All simultaneous sections of a module have a staff student committee. Each committee meets at least three times each semester and may discuss any matter of concern with the module. The staff members of each committee are the lecturers of the concerned sections.

2. **Departmental and Deanship Meetings**
   The meetings, held by the head of Department and the Dean of the Faculty during term time, has mainly an advisory role, where students may raise their problems that need some concern from these authorized persons. These meetings are held separately for each year students.

3. **Module Evaluation Questionnaires**
   The Department attaches great importance to the opinion of students on the quality of the teaching provided, and every student is asked to complete a Module Evaluation Questionnaire for each module. The questionnaires are anonymous.

XI. Communications

1. **Official Notices**
   Official notices are posted on the notice boards at the Department and at the Faculty. Electronic mail is also used extensively for communication with the Department and University. Each lecturer provides the students with his/her e-mail at the beginning of the term. Most official information including copies of this handbook, the undergraduate syllabus and timetables are available on the University Web pages. This includes directories of staff and students for internal use, completed with photographs.

2. **Electronic Mail**
   Electronic mail is used widely for administrative purposes within the Department. It is frequently useful for communicating between individuals and small groups (e.g. between a tutor and his/her tutorial group), and occasionally for broadcasting important messages to wider groups. It is important that you know how to use email. It will be covered in the introductory laboratory sessions. The code of practice for computer usage covers electronic mail, please note the points below.

3. **Obscene or Offensive Mail**
   DO NOT SEND OBSCENE OR OFFENSIVE MAIL. If you receive mail, which you regard as offensive or obscene, you may wish to complain to a member of staff so that appropriate disciplinary action can be taken against the offender.

4. **Group Mailing**
   You are strongly discouraged from sending email to groups of people. The newsgroups should be used for this purpose.

5. **Miscellaneous Hints**
   - Be brief in your communications.
   - Compose your message as if ALL of your recipients were physically present.
   - Limit the distribution of messages to the people who are likely to be interested.
   - Keep a copy of the mail you send out, for future reference. Learn to use folders to keep useful messages.

- Read all your incoming mail before replying to any of it. There may be other relevant messages for you to read.
- Be careful when replying to messages. You probably want your reply to go only to original message sender - not to the whole of the distribution list.
- When you reply to a message, it is frequently helpful to include some of the original message to help your recipients to remember and understand the context of the reply.

XII. Curriculum Design, Content and Organization

1. Curriculum Design and Content.

Students should complete 44 modules (132 credit hours) summarised as follows:

- 9 modules (University requirements) (27 credit hours) (20.45 %)
- 7 modules (Faculty requirements) (24 credit hours) (18.18 %)
- 22 modules (Departmental Compulsories) (66 credit hours) (50 %)
- 2 modules (Departmental Electives) (6 credit hours) (4.55 %)
- 4 modules (Supportive modules) (9 credit hours) (6.81 %)

The Department covers the Software Engineering program from the areas listed below:

1. **Programming Fundamentals**
2. **Languages / Algorithms**
3. **Architecture / Operating Systems**
4. **Networks / Distributed Computing**
5. **Information Systems / Intelligent Systems**
6. **Information Management**
7. **Human Computer Interaction / Applications**
8. **Professional Practice / Software Engineering**
9. **Project / Training**

Table (1) gives the number of covered modules in each area. Note that the ratios are calculated according to **44**, which is the total number of modules that each student should study. In this Table, the total number of the compulsory modules is shown as **34** rather than **24.** This is because some modules from University and Faculty requirements are included as they are strongly related to the compulsory modules. **Table (2)** illustrates the taught modules in each area.

Table (1) Areas of Specialization and Number of Modules

	Area	Compulsory Modules		Elective Modules		Total No. of Modules
		No.	(No. /44) %	No.	(No./44) %	
1-	Programming Fundamentals	**6**	13.63%	**0**	---	**6**
2-	Languages / Algorithms	**2**	4.54 %	**0**	---	**2**
3-	Architecture / Operating Systems	**2**	4.54%	**1**	2.27 %	**3**
4-	Networks / Distributed Computing	**2**	4.54%	**1**	2.27 %	**3**
5-	Information Systems / Intelligent Systems	**2**	4.54%	**2**	4.54%	**4**
6-	Information Management	**2**	4.54%	**0**	---	**2**
7-	Human Computer Interaction / Applications	**3**	6.81 %	**3**	6.81 %	**6**
8-	Professional Practice/ Software Engineering	**11**	25.00 %	**1**	2.27 %	**12**
9-	Project / Training	**2**	4.54 %	**0**	---	**2**
	Total	**32**	**72.73 %**	**any 2**	**4.55 %**	**40 (90.91%)**

2. **Curriculum Organization.** The curriculum is organised as it is shown in the study plan in **Appendix C**.

3. **Curriculum Characteristics**
   - ***Objectives of the Main University-Requirement Modules.*** These requirements are to broaden the student's base for different topics such as culture, languages, and computer skills.

   - ***Objectives of the Main Faculty-Requirement Modules.*** These requirements are to consolidate mainly the student's background in Mathematics and some other common topics. They constitute the common knowledge required for all students in the Faculty of Information Technology.

   - ***Objectives of the Main Computing Modules in the Curriculum.*** The modules in the curriculum are organized into three types: **introductory, intermediate**, and **advanced** modules. The curriculum is designed according to the **Imperative First Strategy** for the introductory modules. This model also focuses on programming, but emphasises the principles of object-oriented programming and Design from the second semester of the first year. The curriculum of Intermediate modules is designed according to the **Topics-based approach**, which is the most common approach for the intermediate modules. Students take separate modules in each of the core areas enumerated below (programming fundamentals with object-oriented paradigm, Software Engineering, Multimedia, Professional Practices, etc.). For the advanced modules, the Department wishes to orient such modules to its own areas of expertise**.** The advanced and elective modules contain more advanced topics in the areas of Software Engineering, Real Time Systems, and Project and Training.
     Recent methodology in programming such as object-oriented programming, software tools, and current technologies in multimedia systems and network systems are included in the curriculum.

   - ***Objectives of the Training and Graduation Project Modules.*** The objectives of these modules are to allow students to gain practice in problem analysis, design, implementation, report writing, and presentation.

   - ***Elaboration on Content and Emphasis of Practical Components of Modules.*** Most of the modules contain practical work that make students involved in using current software tools and computing technologies. Thus, the practical part of modules accounts for at least 25% of the total number of hours. Many laboratory assignments are given during the semester through which the students can practice what they have learned from the theoretical part of the module, or develop their skills in using most recent software tools and programming languages. For example, the practical work in "Programming Fundamentals" and "Object-Oriented Paradigm" modules emphasize on problem solving and structured and object-oriented programming via C++ language and Java language. However, the practical work in Operating System module is concerned with inter-process communication, while in Net-Centric computing it is concerned with client server applications and simulation of OSI protocols.

4. ***Innovation of Curriculum.*** The curriculum is constantly evolving to cope-up with new technologies and rapidly developing software. The first curriculum was designed in 1994 and updated in 1999, 2000, 2001, 2003, 2005, and 2008. This development is through regular internal monitoring and reviews, and to recent local developments in teaching and learning. For example, the Curriculum 2001/2002 is a clear specialisation in development of software and information systems that are supported by the object-oriented technology. Proceeding in this way provides a curriculum that matches the aims and objectives of the Department and the University. The Scientific Committee with the Syllabus setup committee of the Department usually recommend development and modification of curriculum.

Table (2):the Taught Modules in Each Area of Specialization

A- The Compulsory Specialisation Modules	B- The Elective Specialisation Modules
1. Programming Fundamentals • 210104 Discrete Structures • 750112 Programming Fundamentals • 721120 Object-Oriented Paradigm • 721221 Object-Oriented Data Structures	***1. Programming Fundamentals*** None
2. Languages / Algorithms • 750321 Concepts of Programming Languages • 750322 Algorithms Design and Analysis	***2. Languages / Algorithms*** None
3. Architecture / Operating Systems • 750232 Computer Organization and Architecture • 750333 Principles of Operating Systems	***3. Architecture / Operating Systems*** • 721444 Real Time Systems
4. Networks / Distributed Computing • 721321 Concurrent and Distributed Programming 761340 Fundamentals of Computer Networks	***4. Networks / Distributed Computing*** • 721443 Telecommunication Software Design
5. Information Systems / Intelligent Systems • 731150 Introduction to Information Systems • 750351 Fundamentals of Artificial Intelligence	***5. Information Systems / Intelligent Systems*** • 721340 E-Commerce Systems Development
6. Information Management • 760261 Database Fundamentals	***6. Information Management*** None
7. Human-Computer Interaction/Applications • 761272 Multimedia Systems • 731270 World Wide Web: Concepts and Programming • 721323 Graphical User Interface Design	***7. Human-Computer Interaction / Applications*** None
8. Professional Practice / Software Engineering • 721210 Introduction to Software Engineering • 722282 Computing Ethics • 721222 Software Modeling • 721230 Software Requirements • 721320 Software Architecture • 721330 Software Production • 721322 Software Design • 721420 Software Construction and Evolution • 721430 Software Testing • 721421 Software Reverse Engineering • 721331 Software Project Management	***8. Professional Practice / Software Engineering*** • 722489 CASE Tools Development • 721431 Software Metrics and Measurements • 721442 Software Engineering for Web Applications • 721446 Special Topics in Software Engineering • 721487 Formal Methods in Software Engineering •
9. Project / Training • 721440 Practical Training • 721441 Research Project	***10. Project / Training*** None

XIII. Health and Safety in the University

The University has a Health and Safety Committee, which comprises representatives of all services within the University. It is the responsibility of this committee to investigate complaints and potential hazards, to

examine the cause of all accidents and to carry out periodic inspections of all areas of the Department. At registration, you will be required to assent to the departmental code of behavior, which relates to health and safety.

1. **Buildings**
   The Department comprises two kinds of buildings: the Rooms Building and the IT Laboratories.The buildings are generally open between 08.00 and 19.30 (Sunday – Thursday). In accordance with University policy, smoking is prohibited throughout all buildings.

2. **Emergency Evacuation**
   It is the responsibility of every individual to familiarize themselves with the Department's buildings and be aware of the fire exits.
   - After evacuation of any building, please assemble well away from the building, and do not block any exits.
   - Do not return to any building until authorized to do so.

3. **Fire Action**
   Fire Action notices are located at, or adjacent to, fire alarm actuation points, and all staff and students should make them acquainted with this routine.

4. **Operating the Fire Alarm**
   The manual fire alarm system can be activated by breaking the glass in the red contact boxes sited at strategic points throughout the premises.

5. **Use of Fire Appliances**
   Fire appliances are sited at strategic points throughout the Department to deal with fires. Fires should only be tackled provided there is no personal danger and after the alarm has been set off.

6. **Action when the Alarm Rings**
   On hearing the intermittent alarm, you should prepare yourself to leave the building.
   On hearing the continuous alarm, you should evacuate the building immediately by the nearest exit.

7. **Personal Difficulties**
   Please inform the Department's counselors or your tutor of any difficulties with which the Department can be of assistance.

Appendix A

The Guidance Plan

of

Software Engineering Programme

(2008 – 2009)

Software Engineering Department

Guidance Plan
(132 Credit Hours)
(2008 - 2009)

Year	Semester	Module Number	Module Title	Prerequisite	Types of Requirements
First	First (18 Credit Hours)	110101	Arabic Language Skills (1)	----	UR
		130101	English Language Skills (1)	----	UR
		----	UE	----	UE
		750112	Programming Fundamentals	----	FR
		210101	Mathematics (1)	----	SR
		210104	Discrete Structures	----	SR
	Second (18 Credit Hours)	111101	National Education	----	UR
		----	UE	----	UE
		130102	English Language Skills (2)	130101	FR
		721120	Object Oriented Paradigms	750112	FR
		731150	Introduction to Systems and Information Technology	750112	FR
		----	UE	----	UE
Second	First (18 Credit Hours)	721240	Computing Ethics	731150	FR
		731270	Introduction to Web Programming	750112	FR
		761272	Multimedia Systems	----	FR
		210231	Introduction to Statistics and Probabilities	----	SR
		721210	Introduction to Software Engineering	731150	DR
		721221	Object Oriented Data Structures	721120+210104	DR
	Second (18 Credit Hours)	761211	Windows Programming	721120	FR
		----	UE	----	UE
		721222	Software Modeling	721210	DR
		721230	Software Requirements	721210	DR
		750232	Computer Architecture	731150	DR
		760261	Database Fundamentals	721221	DR
Third	First (18 Credit Hours)	721320	Software Architecture	721222	DR
		721321	Concurrent and Distributed Programming	721221	DR
		721330	Software Production	721222	DR
		750322	Design and Analysis of Algorithms	721221	DR
		750333	Principles of Operating Systems	750232	DR
		750351	Fundamentals of Artificial Intelligence	721221	DR
	Second (15 Credit Hours)	721322	Software Design	721230+721320	DR
		721323	Graphical User Interface Design	721320+761211	DR
		721324	Advanced Object Oriented Programming	721321	DR
		721331	Software Project Management	721330	DR
		761340	Fundamentals of Computer Networks	721221	DR
Fourth	First (15 Credit Hours)	----	UE	----	UE
		721420	Software Construction and Development	721322	DR
		721430	Software Testing	721322	DR
		721440	Practical Training *	Department Agreement	DR
		----	DE	----	DE
	Second (12 Credit Hours)	111100	Military Sciences	----	UR
		721421	Software Reverse Engineering	721420	DR
		721441	Research Project	721440+721420	DR
		----	DE	----	DE

(UR) University Req. **(FR) Faculty Req.** **(DR) Dept. Req.** **(SR) Supportive Req.**

Appendix B

Outlines of Module Descriptions

2009 – 2010

I- The University Requirements and Faculty Requirements

In the University requirements, only the computer-based modules are considered here.

(A) University Requirements

750111, Computer Skills

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

Aims: Introduction to computer systems and practical use of software packages.

Teaching Method: 30 hours Lectures and Laboratory (2 per week) + 15 hours Example sessions (1 per week)

Textbook*:*

1- Joseph S. Akasheh et al, Introduction to Computer Science and Programming in Basic; chapter 1, chapter 10, Second Edition, 1993.
2- Robert T. Grauer et al, Exploring Microsoft Windows 98 and Essential Computing Concepts, Prentice Hall, 1998
3- Robert T. Grauer et al, Exploring Microsoft Office 97 Professional, Volume 1, Prentice Hall, 1998.

Synopsis*:* Introduction to computer systems and practical use of software packages. Introduction, MS-DOS, MS-Windows, WinWord, Excel, PowerPoint, Internet.

Assessment*:* Two 1-hour midterm exams (15% each); Assignments (20%); 2-hours Final Exam (50%)

(B) Faculty Requirements

750112, Programming Fundamentals

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

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

Aims: This module aims to introduce the principles of Top Down problem solving strategy (divide and conquer), algorithm design, and imperative programming mainly at an abstract level. Topics include data definition structures, control structures, and primitive data structures. C++ programming language (in visual environment) is adopted as a vehicle language for implementations.

Textbooks:

1- Friedman Frank and Koffman Elliot B., "*Problem Solving, Abstraction and Design using C++"*, Addison Wesley, Fourth Edition, 2001
2- Deitel & Deitel, C++ How to Program, Prentice-Hall, 2000

Synopsis: Problem Solving, Problem Solving Methodology: Analysis, Design (Algorithm), Coding (program), Testing, Maintenance, Top Down Algorithm design (Sub algorithm : function), Data Definition Structures: Types, constants, variables, Expressions: Arithmetic, Logical, Control Structures: I/O, Assignment, Sequence, Selection (simple, alternated, and multiple), Repetition (While, do while, for), Parameters definition and passing (function depth look), Record **(**non uniform set), Array of 1 and 2 dimensions (uniform set), Strings (use of main operations: Concatenate, Left_N_char, Right_N_Char, Include, Compare, …), File (use of main operations of a sequential file: open, reset, rewrite, read, write, eof), VC++ environment: Editor, compiler, linker, Run, and debugger, Programming with C++: Translating Algorithm structures into C++ structures

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

……..

761272, Multimedia Systems

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

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Tutorials (1 per 2 weeks)

Aims: This module is an introduction to the major topics related to multimedia (desktop publishing, hypermedia, presentation media, graphics, animation, sound, video, and integrated authoring techniques), multimedia devices and development tools. It emphasizes hands-on experience for students to familiarize them with the range of tools used in creating computer-based multimedia.

Textbooks:
1. Vaughan, Tay, Multimedia: Making it work, Berkeley Osborne McGraw-Hill, 4th Edition1998.
2. Stephen McLoughlin, Multimedia: Concepts and Practice, Prentice hall, 2001

Synopsis*:* Introduction to Multimedia: Basic concepts, Applications (video on demand, Videoconferencing, virtual learning, entertainment, games, simulations, virtual reality…), Multimedia Hardware, Multimedia Software Tools (Overview on current available tools), Desktop Publishing, Graphics, Pictures: graphic modes and formats, still pictures and format (JPEG…), User Interface Design and Graphics: Graphic Elements and user interface considerations (Backgrounds, buttons, presentation elements), Production Planning and Design: (Research, content flow, Content acquisition, Multimedia team management using project management software, Budgeting considerations, Element and resource lists), Audio and Sound, Analogue Video (1), Digital video (2), Animation, Authoring, Hypermedia Authoring: Authoring: Web Based Multimedia, Multimedia Compression: Overview on techniques and standards.

Assessment: Two 1-hour midterm exams (15% each); Project work (10%); Assignments (10%); 2-hours Final Exam (50%).

……

731270, World Wide Web: Concepts and Programming

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

Teaching Method: 18 hours Lectures (1-2 hours per week) + 7 hours Tutorials (1 per 2 week2) + 20 hours Laboratory (1-2 hours per week)

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.

Textbooks*:*
1- Deitel & Deitel, “Internet & World Wide Web How to Program”, Prentice Hall, 2000.
2- Comer, “Computer Networks”, Prentice Hall, 1999.
3- HTML for fun and profits, Prentice Hall, 1999.

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, and Telnet; The Web: Basic Concepts, WWW and Web Servers, Links: Hyperlinks & Hypermedia, Web pages and home pages, Browsers & Search Engines; HTML: Basics and Programming; Script Languages; Web Servers: Basics and Programming: Introduction to Web Servers, Active Server Pages (ASP), ASP Examples; Overview of Web and Internet Tools.

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

……

721240, Computing Ethics

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

Teaching Method: 30 hours Lectures (2 hours per week) + 10 hours Seminars (average 1 per week) + 3-5 hours presentations at the end of the semester (depending on the number of students in the class) where students present their work in the essays.

Aims: This module aims to give students an informed awareness of the principal issues of professional ethics and responsibility in the design, implementation and use of computer and information systems. In addition, the module aims to help in recognition of ethical problems when they occur, and to enable students to deal effectively with ethical and professional issues now and in their future careers. The module does not require a laboratory, but one group and one individual essay are required. Students are expected to spend 10 - 20 hours preparing for these essays at outside lecturer times. Students are asked at the end of the semester to present their essays.

Textbooks:

1- Ayres R., The Essence of Professional Issues in Computing. ISBN 0-13-908740-0, Prentice Hall Europe 1999.
2- Dejoie, R. et al., Ethical Issues in Information Systems. (ISBN 0-878-355-626), Boyd & Fraser 1991.
3- Bott F et al, Professional Issues in Software Engineering, 3rd Edition (ISBN 0748409513), Pitman 2000, UCL 1995 .

Synopsis*:* Introduction to the module, Problems of ethical decision-making, Professional Societies and their codes of conduct and practice, Professionals and Professional Behavior, Discussion of Case Studies: Describing Steps to Resolve the Current Situation, Preparing Policies and Strategies to Prevent Recurrence. Introduction to the Crawling Eye case study, Formal laws do not make for ethics, Graduate careers in the 21st century, Building the foundations to future career success, Concurrent engineering, group working and distributed enterprises, The law and contracts, Safety critical systems and legal liability, Introduction to the Killer Robot case study, A business view of contracts, IPR and copyright, IPR and patents, Computer misuse and the law, Data Protection, the Act and its implications.

Assessment: Two (1 hour) midterm exams (15% each); Assessment by individual essay (10%); Assessment by group essay (10%); 2-hours Final Exam (50%).

……

731150, Introduction to Information Systems

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

Teaching Method: 38 hours Lectures (2-3 hours per week) + 7 hours Tutorials (1 per 2 weeks).

Aims: This is a major introduction course that presents problems in business environment and solution with computer-based tools. It focuses on systems and information systems concepts and technologies. Students will learn the most effective ways to use information systems. Case studies are examined to highlight new technology and applications like multimedia.

Textbooks:

1- Gerald M. Weinberg, An Introduction to General System Thinking, Silver Anniversary Edition, 2001.

2- Leonard M. Jessup and Joseph S. Valacich, Information Systems foundations, Que E&T, 1999.

References:

1- James A. O'Brien, Introduction to IS: Essentials for the E-Business Enterprises, 11th ed. 2003, McGraw Hill.
2- Steven Alter, Information Systems a Management Perspective, 2nd ed., 1996, The N/Benjamin/Cummings Publishing Company, inc.
3- Senn, James A., Analysis and Design of Information Systems, 1989, New York, McGraw-Hill.

Synopsis: System 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.

Assessment: Two midterm exams (15% each); Asignments (15%); Tutorial contribution (5%); 2-hours Final Exam (50%).

……

750351, Fundamentals of Artificial Intelligence

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

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

Aims: This module aims to present the basic representation and reasoning paradigms used in AI in both theory and practice with careful attention to the underlying principles of logic, search, and probability. It is also designed to show students practical examples of the use of AI in applications and to encourage further reading. The assignments given to students aim to provide a sound practical introduction to knowledge-based systems and a basic introduction to modern paradigms of knowledge representation and belief networks. The tutorials aim to provide an introduction to the underlying issues in cognitive emulation and to provide an opportunity for practical exercises in logic and probability.

Textbooks:

1- S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, 2001
2- G.F. Luger and W.A. Stubblfield, Artificial Intelligence: Strategies for Complex Problem Solving, 1998
3- P. H. Winston, Artificial Intelligence, Addison Wesley, 1992

Synopsis: Fundamental issues: History of AI, philosophical questions; definitions of intelligent systems; modelling the world; the role of heuristics. Search and constraint satisfaction: Problem spaces; brute-force search; best-first search; two-player games; constraint satisfaction. Knowledge representation and reasoning: Review of propositional and predicate logic; resolution and theorem proving; non-monotonic inference; probabilistic reasoning; Bayes theorem. Advanced search: Genetic algorithms; simulated annealing; local search. Advanced knowledge representation and reasoning: Structured representation; non-monotonic reasoning; reasoning on action and change; temporal and spatial reasoning; uncertainty; knowledge representation for diagnosis, qualitative representation. Machine learning and neural networks: Definition and examples of machine learning; supervised learning; learning decision trees; learning neural networks; learning belief networks; the nearest neighbour algorithm; learning theory; the problem of over fitting; unsupervised learning; reinforcement learning.

Assessment: Two 1-hour midterm exams (20% each) + Assignments (5%) + Tutorial contribution (5%); 2-hours Final Exam (50%).

……

II- The Departments' Major and Supplementary Requirements

(a) Supplementary Requirements

210101, Mathematics (1)

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

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

Aims: This module aims to provide students with some background in different topics in mathematics such as derivatives, applications of derivatives, integrals, applications of integrals, transcendental functions and inverses of functions.

Textbooks:
1- Salas, Hille, and Etgen, Calculus, 9th Edition, Wiley, 2002
2- Howard Anton, Calculus, Wiley, 2002

Synopsis: General Introduction: (Inequalities, functions);Limits and continuity; differentiation: (rate of change, chain rule, implicit differentiation); the mean value theorem:(maxima and minima, applications, - concavity, curve sketching); Integration: (the fundamental theorem of calculus, change of variables, applications (area, motion, solids of revolution); the transcendental functions: (differentiation and integration) .

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

……..

210103, Mathematics for Computing

3 hours per week, 3 credit hours, prerequisite: **210101**

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

Aims: This module aims to provide students with some background in sequences and series, and multi-variable calculus, and systems of linear equations and their solutions.

Textbooks:
1- Salas and Hille, Calculus, 7th Edition, Wiley, 1995
2- Howard Anton, Elementary Linear Algebra, 9th Edition, Wiley, 2000
3- Silverman, R.A., Calculus With Analytic Geometry, Wiley, 1998

Synopsis: Infinite series (convergence, Taylor series, power series), vector Calculus (functions of several variables, partial Derivatives; double and triple integral over a region), vectors, Linear algebra, Linear equations, Gaussian elimination, Eigen values and Eigenvectors, introduction to Linear transformation).

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

……..

210104, Discrete Structures

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

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

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.

Textbook:

1- J.K. Trust, Discrete Mathematics for Computer Scientists, Addison Wesley, 1998 or latest Edition.
2- Mattson, H.F., Discrete Mathematics With Applications, Wiley, 1993
3- Garnier, R. and Taylor, J., Discrete Mathematics for New Technology, Institute of Physics Publishing, 1992
4- Eccles, P.J., An Introduction to Mathematical Reasoning, C.U.P, 1997

Synopsis: Boolean algebra logic, truth tables, methods of pouf, induction, sets, recurrence relations, number theory (natural numbers, counting), binary relations, algorithms, (their classification and complexity).

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

…….

210231, Introduction to Probability and Statistics

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

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

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

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

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%).

……..

(b) Department Requirements

750232, Computer Organization and Architecture

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

Teaching Method: 30 hours lectures (2 hours per week) + 9 hours Tutorials (1 per week), 6 hours seminars (1 per week on average)

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.

Textbooks:

1-David A. Patterson, John L. Hennessy, John L. Hennessey Computer , "Organization and Design: The Hardware/Software Interface",
2- William Stallings, "Computer Architecture and Organization: Design for Performance", Prentice Hall, 2000

Synobsis: Aims of the course, 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.

Assessment: Two midterm exams (15% each); Course work (10%); Tutorial contribution (5%); Seminars (5%); 2-hours Final Exam (50%)

……

750333, Principles of Operating Systems

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

Teaching Method: 37 hours Lectures (2 hours per week) + 8 hours Tutorials (1 per 2 weeks).

Aims: The aim of this module is 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.

Textbooks:
1- J. Bacon, Concurrent Systems: Database and Distributed Systems, 2nd Edition, (ISBN 0-201-177-676), Addison Wesley, 1998.
2- A. S. Tanenbaum, Modern Operating Systems, Prentice Hall, 1992.
3- Abraham Sillberschatz and Peter B. Gavlin, Operating Systems Concepts, Addison Wesley, 1994

Synopsis: Processes: Purpose of the Operating System. OS entities and functions (processes and memory); Process and Threads Management Creation / Scheduling / Termination Communication/Synchronization; The OS Kernel; Memory Systems: Hierarchical Organization. Contiguous storage allocation, Single- and multi- programming, Static and Dynamic partitioning, Segmentation, Paging. File Systems: Directory organization. File types and file organization, File systems implementation. I/O Devices: I/O devices and interfaces; polling and interrupts. The general exception mechanism for interrupts, errors, system calls, memory management. Character and DMA interfaces. Protection and Security: Security (Threats, Authentication), Protection mechanisms (access control lists, capabilities). Case Studies: Unix, Linux and Windows 2000 Overviews.

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

……..

721321, Concurrent and Distributed Programming

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

Teaching Method: 37 hours Lectures (2 hours per week) + 8 hours Seminars (1 per 2 weeks).

Aims: The aim of this module is to study, learn, and understand the main concepts of concurrency. Hardware and software features to support concurrency, language features for concurrent and distributed systems, and concurrent and distributed algorithms.

Textbook:
J. Bacon, Concurrent Systems: Database and Distributed Systems, 2nd Edition, (ISBN 0-201-177-676), Addison Wesley, 1998.

Synopsis: Concurrent model of execution; interleaving; atomic operation; critical sections and mutual exclusion; deadlock; starvation; invariants. Cocurrent and distributed algorithms: producer-consumer; reader-writer problems; dining philosophers. Architectural features to support concurrent and distributed systems. Language features for concurrent and distributed systems. Performance evaluation.

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

……

721120, Object-Oriented Paradigm

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

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 modern software engineering principles and developing fundamental programming skills in the context of a language that supports the object-oriented paradigm (Java for instance).

Textbooks:
1- Barnes D. and all, Objects First with Java: A Practical Introduction using BlueJ, Prentice Hall, 2002
2- C. Thomas Wu: "An Introduction to Object-Oriented Programming with Java", 2nd edition, published by McGraw-Hill, 2001, ISBN 0-07-118195-4.
3- Ben Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, (3rd edition) Addison-Wesley, 1998.

Synopsis: Introduction to Object Oriented Thinking: Object Modelling; Objects and Classes; Object Interaction; Grouping Objects; Using Library Classes; Designing Classes; Inheritance (1): reuse, inheritance; Inheritance (2): polymorphism; Abstract classes; Abstract methods; Interfaces; Multiple inheritance; Designing Applications; A case study; Error handling.

Assessment: Two midterm exams (15% each); Laboratory (15%); Tutorial contribution (5%); Final exam (50%).

……

721210, Introduction to Software Engineering

3 hours per week, 3 credit hours, Second year, First semester, prerequisite: **731150**

Teaching Method: 35 hours Lectures (2-3 hours per week) + 7 hours Tutorials (1 per week), 3 hours Seminars (1 per 2 weeks)

Aims: This module aims to provide students a comprehensive introduction to software engineering. It gives an introduction to basic concepts, principles and techniques used in software engineering. It discusses the nature of software and software projects, software development models, software process maturity, project planning, management, and communication. This module gives an introduction to methods for analysis, design, testing, and implementation of large, complex software systems.

Textbooks:
1- Ian Sommerville**,** Software Engineering 6TH, Addison Wesley Longman, Inc. August 2000
2- D. C. Ince, An introduction to discrete mathematics, Formal systems, and Z. Ed Clarendon Press, Oxford. 1992
3- D. F. D'Souza, Objects, Components and frameworks with UML, Addison Wesley, 1998
4- S. Benett, Object-Oriented System Analysis and Design, Ed Mc Graw Hill, 1999
5- R. S. Pressman Software Engineering: A Practitioner's Approach, 5th Edition, McGraw Hill; 2001

Synopsis: Introduction to Software Engineering: Motivation, Basic concepts (Life cycle, phases, software quality, process, method, methodology, mission and roles of phases, Abstraction, encapsulation, information hiding, etc.); The Software Process: Software Process, Life Cycles, Process Tools; The Software Project Management: Scheduling, Personnel, Estimation, Risk Management, Managing for Results; Software Requirements: Representing Requirements, Prototyping, Documentation, Participants, Requirements Analysis and Specification; Introduction to Formal Specification; Software analysis and design: Structured analysis and design (DFD, ERD, Decision tables), Object oriented analysis and design (Class diagram, State diagram, etc.), User Interface Design; Software Construction: Automatic implementation generation from design model, Programming with error avoidance, Programming with error tolerance, defensive programming, Programming for and with reuse, Programming Database applications with encapsulating blocks; Verification and validation: Relation to software quality - Software metrics, Unit Testing, Integration Testing & OO Testing, Test Planning System Testing, Test Management; Introduction to Maintenance: Introduction to Software configurations management, Introduction to Software Reengineering, Quality Assurance and risk management, Software engineering principles: tool, technique, method, methodology, process. Lifecycle models; sizing, estimation, planning and control; Support techniques: object-oriented paradigm, software database, parallel and distributed programming, Requirements specification; design; implementation, Integration and testing strategies; quality assurance; configuration management; Software maintenance; Reuse and reengineering.

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

……..

721221, Object-Oriented Data Structures

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

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

Textbooks:
1- Nell Dale, Daniel T. Joyce and Chip Weems, Object-Oriented Data Structures using Java,

2- Goodrich and Tamassia, Data Structures and Algorithms in Java, 2nd edition, John Wiley and Sons, 2000, ISBN 0471383678.
3- Arnold, Gosling, and Holmes, The Java Programming Language, 3rd edition, Addison-Wesley, 2000, ISBN 0201704331.

Synopsis: Introduction to Software Engineering, 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 (15% each); Assignments (20%); 2-hours final exam (50%).

……..

721230, Software Requirements

3 hours per week, 3 credit hours, prerequisite: **721210**

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

Aims: The aim of this module is to introduce the basic concepts and principles of software engineering requirements.

Textbooks:
Ian K. Bray, An Introduction to Requirement Engineering, ISBN 020176792-9, 2003

Synopsis: Basic concepts and principles of software requirements engineering, its tools and techniques, and methods for modelling software systems; various approaches to requirements analysis are examined: structured, object-oriented, and formal approaches.

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

……..

721322, Software Design

3 hours per week, 3 credit hours, prerequisite: **721230 + 721320**

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

Aims: This module emphasizes on the basic concepts of software design.

Textbooks:
1. Ian Sommerville, Software Engineering 6TH edition, Addison Wesley Longman, Inc., 2000
2. Hoffer, George, and Valacich, Modern Systems Analysis & Design, 3rd edition, ISBN 0-13-033990-3

Synopsis: Software Systems, Software Cricis, Software analysis and design, Problem analysis Process, Object model, behaviour model, Process model, Communication model; Entity Relation Diagram (ERD), Data flow Diagram (DFD), Data Dictionary (DD), Process Specification, Structured Design; UML, Paradiam, OMT (Tumbaugh), OOA (Booch); Design Patterns; Risk management; Quality Assurance.

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

……..

721330, Software Production

3 hours per week, 3 credit hours, prerequisite: **721222**

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

Aims**:** This module looks at how software quality assurance and configuration management are performed and how software process improvement is maintained in order to assure the highest possible quality.

Textbooks**:**
1- Daniel Galin, Software Quality Assurance: From Theory to Implementation, 2004
2- Jarvis, A. and Vern, C., Inroads to Software Quality, Prentice-Hall, 1997

Synopsis: Software Process; Conventional methodologies; An Agile Process; Specific Software Process; Generic Software Process; Software Qual;ity; Product Management and Metrics; Process Mangement and Metrics.

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

761340, Fundamentals of Computer Networks

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

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

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

Textbooks**:**
Johan Zuidweg, Next Generation Intelligent Networks, Artech House, ISBN

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%).
……..

721320, Software Architecture

3 hours per week, 3 credit hours, prerequisite: **721222**

Teaching Method: 30 hours Lectures (2 hours per week) + 8 hours Tutorials (1 per 2 weeks) + 7 hours seminars (1 per 2 weeks)

Aims**:** This module is concerned with the detailed software design after students learned some software design concepts.

Textbook:
McConnel and Steve, Code Completed: a Practical Handbook of Software Construction, Microsoft Press, 1993.

Synopsis: Design Methods, Design Patterns, Component design, Component interface design, Design Notations and Support tools, and Evaluation.

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

……..

721420, Software Construction and Evolution

3 hours per week, 3 credit hours, prerequisite: **721322**

Teaching Method: 30 hours Lectures (2 hours per week) + 8 hours Tutorials (1 per 2 weeks) + 7 hours seminars (1 per 2 weeks)

***Aims*:** The module examines issues, methods, and techniques associated with constructing software, given a high-level design, and for maintaining software over its lifetime.

***Textbooks*:**
1- Barbara Liskov and John Guttag, *Program Development in Java,* Addison Wesley, 2001, ISBN 0-201-65768-6.
2- J. Bloch, *Effective Java: Programming Language Guide*, Addison Wesley, 2001, ISBN 0-201-31005-8.

Synopsis: Language-oriented issues: Programming style and idioms, defensive programming; Construction technology: Selections of data structures, API design and use, code reuse and libraries, object-oriented issues, exception handling and security, middleware; Software construction tools: Application of abstract machine, automatic generation of code; Evolution processes; Evolution activities.

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

……..

721430, Software Testing

3 hours per week, 3 credit hours, prerequisite: **721322**

Teaching Method: 30 hours Lectures (2 hours per week) + 7 hours Tutorials (1 per 2 weeks) + 7 hours seminars (1 per 2 weeks)

***Aims*:** This module introduces the role of verification and validation in the system life cycle.

Textbook*:*
Robert O. Lewis, Independent Verification and Validation, Publisher John Wiley; Sons, Inc. ISBN 0-471-57011-7

Synopsis: Role of verification and validation (V&V) in the system life cycle;. Techniques and tools; Quality assessment, testing, inspection, proof-of-correctness and relevant V&V standards; The student will be assigned chapters in the required tests to read and use in the projects and assignments.

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

……..

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.

……

721421, Software Reverse Engineering

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

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

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

Textbook:

Synopsis: This module will focus on enabling software maintenance through reengineering; computer-aided techniques to recover information from pre-existing systems; Refactoring, migration, Program transformation, Data reverse engineering, Object Oriented Reengineering.

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

……

721487, Formal Methods in Software Engineering

3 hours per week, 3 credit hours, prerequisite: **721430**

Teaching Method: 30 hours Lectures (2 hours per week) + 8 hours Tutorials (1 per 2 weeks) + 7 hours seminars (1 per 2 weeks)

***Aims*:** This module introduces formal specification and analysis methods and their use in Software Engineering.

***Textbook*:**

M.G. Hinchey and J.P. Bowen (editors), Application of Formal Methods, 1995

Synopsis: This module will examine formal specification and analysis methods and their use in Software Engineering. A secondary focus of the class will be on the many software development activities that benefit from the use of formal methods. Students will study different types of formal specification languages and utilize specific languages to describe software systems. In addition, various tools are available for use that support the discussed formal specification languages and facilitate software development activities, including but not limited to proving program correctness.

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

……

721331, Software Project Management

3 hours per week, 3 credit hours, prerequisite: **721330**

Teaching Method: 30 hours Lectures (2 hours per week) + 8 hours Tutorials (1 per 2 weeks) + 7 hours seminars (1 per 2 weeks)

***Aims*:** The module addresses issues involving the creation, development, and maintenance of software projects.

Textbook*:*
Watts Humphrey, Managing the Software Process, Addison-Wesley, 1995

Synopsis: This module includes project management, risk analysis, project planning, project administration, and configuration management.

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

……..

721441, Research Project

3 credit hours, prerequisite: **721440 + 721420**

General Descriptions:
The graduation project consists of a single project on which the student works over a period of 16 weeks that can be extended to 32 weeks (2 semesters). It is assumed that the student spends a nominal 192 hours (or 384 hours), the equivalent of 12 hours per week, working on this. There are three deliverables: demonstration, discussion, and a written report.
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.

Aims: The aims for the project work done in the fourth year are:
1- To manage and execute a substantial project in a limited time.
2- To identify and learn whatever new skills are needed to complete the project.
3- 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.

Textbook:
C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.

The projects list and notes for guidance in carrying out a project are available in the Graduation Project Committee.

Assessment: Supervisor mark: 35%; Project Examination Committee mark: 65% (demonstration 20%, Report 25%, discussion 20%).

…….

750321, Concepts of Programming Languages

3 hours per week, 3 credit hours, prerequisite: **210104 + 721221**

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

Aims: This module provides students with basic concepts of different programming languages in order to enable students to learn any new programming language easily.

Textbook:
R. Sebesta, *Concepts of Programming Languages*, Addison-Wesley, 5th edition.

Synopsis: Introduction; A survey of programming paradigms, Imperative paradigm: Names, Bindings, Type, checking, and Scopes. Data Types: Primitive data types, Character string types, User-Defined Ordinal Types, Associative arrays, Record Types, Union Types, Set Types, Pointer Types. Statement-Level Control Structures: Compound Statements, Selection Statements, Iterative Statements, Unconditional Branching, Guarded Commands. Subprograms: Fundamentals of subprograms, Design Issues for Subprograms, Local Referencing Environment, Parameter-Passing Methods, Parameters That Are Subprograms Names, Overloaded Subprograms, Generic Subprograms, Separate and Independent Compilation, Coroutins. Abstract Data Type: The Concept of Abstraction, Encapsulation, Introduction to Data Abstraction, Design Issues, Parameterized Abstract Data Type. Support for Object-Oriented Programming: Object-Oriented Programming, Design Issues for Object-Oriented Languages, Support for Object-Oriented Programming in C++. Functional Programming Languages: Introduction, Mathematical Functions, Fundamentals of Functional Programming, LISP, Scheme, Applications of Functional Languages, A comparison of Functional and Imperative Languages. Logic paradigm, Introduction to Predicate Calculus, Predicate Calculus and Proving Theorems, An overview of Logic Programming, Prolog, Smalltalk, Applications of Logic Programming; Scripting Languages.

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

……

750322, Design and Analysis of Algorithms

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

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

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 the underlying mathematical theory and practical considerations of efficiency. Topics include asymptotic complexity bounds, techniques of analysis, and algorithmic strategies. The solved problems are mainly business-oriented.

Textbooks:

1- Sara Baase, Computer Algorithms: Introduction to Design and Analysis, Addison-Wesley, 1998
2- E. Horowitz et al. The Fundamentals of Computer Algorithms, Computer Press, 1992
3- G. Brassard, P.Bratley, Fundamentals of Algorithms, Prentice Hall, 1996

Synopsis: Introduction to Algorithms: Idea of algorithms, algorithms and programs. Basic Algorithmic Analysis: Asymptotic analysis of upper and average complexity bounds; best, average, and worst case behaviors; big-O, little-o, W, and λ notation; standard complexity classes; empirical measurements of performance; time and space tradeoffs in algorithms; using recurrence relations to analyze recursive algorithms. Proof of Correctness. Fundamental Algorithm Design Strategies: divide-and-conquer; greedy; backtracking; branch-and-bound; non-deterministic; numerical approximation. Lower Bound Theory: Sorting and searching, lower bound examples NP-Hard and NP-complete Problems: Basic Concepts, NP-Hard & NP-complete problems, examples.

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

……

760261, Database Fundamentals

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

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

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.

Textbooks:
1- Elmasri R. and Navanthe S. B., Fundamentals of Database Systems, 3rd edition, (ISBN 0-201542633), Addison Wesley, 1999.
2- C. J. Date, An Introduction to Database Systems,

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%).

……

721323, Graphical User Interface Design

3 hours per week, 3 credit hours, prerequisite: **721320 + 761211**

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

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 .Writing of GUI specifications and implementation of GUI applications in the JAVA programming language is also covered.

Textbooks:

1. Alan Dix, Janet Finlay, Gregory Abowd, and Russel Beale, Human-Computer Interaction, 2nd Edition, Prentice Hall, 1998.
2. B.Shneiderman, Designing the User Interface: Strategies for Effective Human Computer Interaction, Addison-wesley,1998
3. Y. Daniel Liang, *Introduction to Java Programming 4th ed.,* Prentice Hall, 2002
4. Dietel and Dietel, Java How to Program, 3rd ed., Prentice Hall, 2000
5. Jenny Preece, Human-Computer Interaction, Addison Wesley, 1994
6. Bruce Eckel*, Thinking in Java*, 2nd ed, Prentice-Hall, 2000
7. Horstmann & Cornell, *Core Java 2,* Volume I – Fundamentals, Prentice-Hall, 1999
8. Horstmann & Cornell, *Core Java 2,* Volume II - Advanced Features, Prentice-Hall, 1999

Synopsis: Introduction. Human Aspects: Introducing a range 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-centred, including principles and methods for user centred 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 midterm exams (15% each); Tutorial / Lab exercises (5%); Quizzes (5%); Project work (10%); 2-hours Final Exam (50%).

……

(c) Elective Modules

721444, Real Time Systems

3 hours per week, 3 credit hours, prerequisite: **750333 + 721322**

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

***Aims*:** This module is a survey of issues in the design and implementation of real time computer systems.

Textbook*:*
Burns A. and Wellings A., *Real-Time Systems and Programming Languages*, 2nd edition, Addison-Wesley, 1997

Synopsis: The module intends to recognize, classify, and formulate the hard and soft timing requirements of a software system, select an appropriate software architecture and combination of scheduling techniques to satisfy a set of timing requirements. Dedicated Real Time languages and Real Time Executives are also introduced.

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

……

721443, Telecommunication Software Design

3 hours per week, 3 credit hours, prerequisite: **721322 + 761340**

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

Aims**:** The module introduces the state art of software concepts, methods, and tools used for the design of advanced telecommunication services (AIN services), and of communication protocols.

Textbook:
Iakovos Venieris, Fabnozio Zizza, Object Oriented Software Technologies in Telecommunications, Thomas Magedanz, 2000

Synopsis: Depth in signals and information theory; Telephony and telecommunications protocols; Related telecommunications systems knowledge.

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

……

721340, Financial and E-Commerce Systems Development

3 hours per week, 3 credit hours, prerequisite: **721320**

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

Aims**:** The module provides breadth knowledge in Financial and E-Commerce fields (Accounting, Finance, E-commerce, and Security) and depth knowledge about the design of Financial and E-commerce systems.

Textbook:

Synopsis: E-commerce systems; Accounting; Finance; depth in security.

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

……

721340, CASE Tools Development

3 hours per week, 3 credit hours, prerequisite: **721320**

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

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.

Textbooks**:**

1- Ian Somerville, Software Engineering, Addison-Wesley, 2000

2- M.J. Pont, Software Engineering with C++ & CASE Tools, Addison-Wesley, 1996.

Synopsis: Introduction, CASE classification, CASE lifecycle, workbenches. Programming workbenches. Analysis and design workbenches. Meta-case workbenches. Environments. Integrated environments, platform and framework services. Software process: modeling, structure and methodologies

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

……

Appendix C

Study Plan
of
Software Engineering Programme

(2008 – 2009)