QFO-AP-FI-MO02	اسم النموذج: Course Syllabus	جامعة فيلادلفيا
رقم الاصدار : (Revision) 1	الجهة المصدرة: كلية تكنولوجيا المعلومات	Philadelphia University
التاريخ: 2017/11/05	الجهة المدققة: عمادة التطوير والجودة	
عدد صفحات النموذج:		

Course Syllabus	
Course Title: Software Mmaintenance and Evolution	**Course code: 750785**
Course Level: *Master*	**Course prerequisite(s) and/or corequisite(s): None**
Lecture Time:	**Credit hours: 3**

Academic Staff Specifics				
Name	**Rank**	**Office Location**	**Office Hours**	**E-mail Address**

Course Description

This course is intended for students already accepted to study the MSc in Computer Science in the Faculty of Information Technology. Students can expect to gain from the course the skills and techniques required to undertake Software Maintenance at MSc level. It aims to present theoretical and a practical techniques, tools, and methodologies that help software developers to achieve this unavoidable task with a maximum of success.. The practical aspect should be emphasized.

Course Objectives

This course aims to offer advanced skills in Software Maintenance: **Software Change** (Introduction to software maintenance methods and tools: Corrective, adaptive , perfective, and preventive maintenance methods and tools), **Change Management** (Software Configurations Management SCM: Concepts, Repository, Changes set, Workspace, System/product model, Composition, Database, Long transaction, Versioning, tools, researches, …), **Program Comprehension – Reverse Engineering (**Code reading: Code reading, top down reading, unfamiliar PL code reading**;** Code Understanding: Software architecture, software understanding tools), **Evolution of Legacy Systems** (Fundamentals and re-engineering by migration: Characteristics o f legacy systems, challenges for their evolution, the re-technologies: reverse-engineering, re-structuring, and re-engineering by migration; Restructuring methods and tools: Error avoidance tools, Architectural tools, reuse tools, Application; Change impact analysis methods and tools: Static Analysis, Dependency Graph, Slicing), **Looking to the future.**

Course Components

Textbooks

1. John Krogsti (Author). Model-based development and evolution of information systems: a quality approach. London: Springer. 2012
2. Tom Mens, Serge Demeyer. Software evolution, Computers, 2008
3. Penny Grubb, Armstrong A. Takang. Software maintenance: concepts and practice, Computers, 2003

In addition to the above, the students will be provided with handouts by the lecturer.

Teaching Methods
Duration: 15 weeks, 45 hours in total. Lectures: 15 hours, 1 per week. Tutorial (case study in classroom): 21 hours, 2 per week. Seminar: 3 (15 mn at the end of each lecture). Laboratories: 15 hours in total, 1-hour per week (free lab). Exams: 6 hours (3h for the mid and 3 for final exam). The last week is reserved to practical works examination.

Learning Outcomes

- ***Knowledge and understanding***
  - be prepared for some of the demands of, and skills required for, work in IT and IT-related industries.
  - have a knowledge and understanding of the importance of taking a long term view of software development at all stages in the software life cycle.
  - have a knowledge and understanding of the processes of change control for software systems
  - have a knowledge and understanding of the specific problems inherent in the maintenance and evolution of legacy software systems, and some of the techniques that can be of use.
  - have a knowledge and understanding of the specific challenges inherent in the maintenance and evolution of data-intensive systems, and some of the techniques that can be of use
  - have a knowledge and understanding of the specific problems inherent in the maintenance and evolution of package-based software systems, and some of the techniques that can be of use.

- ***Cognitive skills (thinking and analysis)***
  - Understand A wide range of principles, methodologies, and tools available to the software maintainer. All these direction informed by research.
  - be able to make use of standard tools and techniques for program comprehension, in order to gain an understanding of an unfamiliar software system.
  - have experience in reading, fixing and modifying software constructed by others, and be able to apply current best practices for software modification
  - be able to make appropriate choices regarding the tools and techniques to apply to software evolution problems, trading off costs and limitations against the expected benefits
  - Understand the professional and ethical responsibilities of the practicing computer professional including understanding the need for quality.
  - Understand the application of computing in a business context.

- ***Communication skills (personal and academic)***
  - Have skills and knowledge necessary to undertake projects.
  - Be able to display an integrated approach to the deployment of communication skills, use IT skills and display mature computer literacy; strike the balance between self-reliance and seeking help when necessary in new situations, and display personal responsibility by working to multiple deadlines in complex activities.

- ***Practical and subject specific skills (Transferable Skills)***
  - Solve a wide range of problems related to software maintenance and evolution.
  - Design, analysis and implementation of different kind of software maintenance.
  - Plan and undertake a major individual project, and prepare and deliver coherent and structured verbal and written technical report.

Assessment Instruments

Allocation of Marks	
Assessment Instruments	**Mark**
Midterm examination	**30%**
Final Exam (written unseen exam)	**40 %**
Reports, research projects	**30%**
Total	**100%**

** Make-up exams will be offered for valid reasons only with consent of the Dean. Make-up exams may be different from regular exams in content and format.*

Practical Submissions
The assignments that have work to be assessed will be given to the students in separate documents including the due date and appropriate reading material.

Documentation and Academic Honesty

Submit your homework covered with a sheet containing your name, number, course title and number, and type and number of the home work (e.g. assignment, and project).

Any completed homework must be handed in the class on the due date. After the deadline "zero" will be awarded. You must keep a duplicate copy of your work because it may be needed while the original is being marked.

You should hand in with your assignments:

- A brief report to explain your findings.
- Your solution of given problem

For the research report, you are required to write a report similar to a scientific research paper. It should include:

- *Ab-stract:* It describes the main synopsis of your paper.
- *Introduction:* It provides background information necessary to understand the research and getting readers interested in your subject. The introduction is where you put your problem definition, summary of contribution, related work, and is likely where the bulk of your sources will appear.
- *Methods (Algorithms and Implementation):* Describe your methods here. Summarize the algorithms (if any) generally, highlight features relevant to your project, and refer readers to your references for further details. Information from sources must be rephrased in own words, "copy-and-paste" from documents, found for example on the Internet, is NOT allowed. It is allowed to use short quotations, or figures, from other documents, but then the source MUST be clearly stated in the reference list (please check copy rights). Papers not fulfilling these rules will be failed.
- *Results and Discussion (Benchmarking and Analysis):* This section is the most important part of your paper. It is here that you demonstrate the work you have accomplished on this project and explain its significance. The quality of your analysis will impact your final grade more than any other component on the paper. You should therefore plan to spend the bulk of your project time not just gathering data, but determining what it ultimately means and deciding how best to showcase these findings.
- *Conclusion:* The conclusion should give your reader the points to "take home" from your paper. It should state clearly what your results demonstrate about the problem you were tackling in the paper. It should also generalize your findings, putting them into a useful context that can be built upon. All generalizations should be supported by your data, however; the discussion should prove these points, so that when the reader gets to the conclusion, the statements are logical and seem self-evident.
- *Bibliography:* Refer to any reference that you used in your assignment. Citations in the body of the paper should refer to a bibliography at the end of the paper.

- **Protection by Copyright**

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. 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.
3. Sources of quotations used should be listed in full in a bibliography at the end of your piece of work.

- **Avoiding Plagiarism**

1. 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.
2. 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.
3. 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.
4. 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 extent 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).

Course/Module Academic Calendar

Week	Basic and support material to be covered	HW

(1)	**Part 1/ Software Change** *Chapter 1. Introduction to software maintenance methods and tools* - Corrective, adaptive maintenance.	PW on Cor& Ad
(2)	- perfective, and preventive maintenance. **Tutorial on relevant research papers**	PW P&P
(3)	*Chapter 2. Changes management/ Software Configurations Management (SCM)* - Concepts, Repository, Changes set, Workspace, System/product model, Composition	PW SCM
(4)	- Long transaction, Versioning, tools, researches. **Tutorial on relevant research papers**	PW Vers.
(5)	**Part 2/ Program Comprehension (Reverse engineering)** *Chapter 3. Code reading* - Code/ top down/ unfamiliar PL code reading. **Tutorial on relevant research papers** *Chapter 4. Code Understanding* - Software architecture/ understanding tools. **Tutorial on relevant research papers**	PW CR PW A&U
(6)	**Part 3/ Evolution of Legacy Systems** *Chapter 5. Fundamentals and re-engineering(by migration)* - Characteristics o f legacy systems, challenges for their evolution, re-technologies: reverse-engin., re-structuring, and re-engineering. **Tutorial on relevant research papers**	PW REE RW REE
(7)	*Chapter 6. Restructuring methods and tools* - Error avoidance /Architectural/Reuse tools. **Tutorial on relevant research papers**	PW RM
(8)	*Chapter 7. Change impact analysis methods and tools* - *Static Analysis*: Dependency Graph, Slicing. **Tutorial on relevant research papers**	PW Slice
(9)	**Mid Exam**	RW DIS
(10)	*Chapter 8. Evolution of Data-Intensive Systems* - Data Reverse Engineering	PW DIS
(11)	- Data Re- engineering. **Tutorial on relevant research papers**	PW DRE
(12)	**Part 3/ Look to the future** *Chapter 9. Novel Trends in Software Evolution* - Bio-inspired software evolution	
(13)	**Tutorial on relevant research papers**	
(14)	**Homework Exam**	
(15)	**Final Exam**	

Expected workload
On average students need to spend 2 hours of study and preparation for each 50-minute lecture/tutorial.

Attendance policy
Absence from lectures and/or tutorials shall not exceed 15%. Students who exceed the 15% limit without a medical or emergency excuse acceptable to and approved by the Dean of the relevant college/faculty shall not be allowed to take the final examination and shall receive a mark of zero for the course. If the excuse is approved by the Dean, the student shall be considered to have withdrawn from the course.

Module References
Students will be expected to give the same attention to these references as given to the Module textbook(s)

1. John Krogsti. Model-based development and evolution of information systems: a quality approach. London: Springer. 2012
2. Donald J. Reifer. Software change management: case studies and practical advice. Microsoft Press, 2011
3. Tom Mens, Serge Demeyer. Software evolution, Computers, 2008
4. Alain April, Alain Abran. Software Maintenance Management: Evaluation and Continuous Improvement, Computers, 2008
5. Gopalaswamy Ramesh, Ramesh Bhattiprolu. Software maintenance: effective practices for geographically distributed environments. McGraw-Hill, 2006
6. Eldad Eilam. Reversing: secrets of reverse engineering. Wiley, 2005
7. R.C. Seacord, D. Plakosh and G.A. Lewis. *Modernising Legacy Systems*, The SEI Series in Software Engineering, Addison-Wesley, 2003
8. Penny Grubb, Armstrong A. Takang. Software maintenance: concepts and practice, Computers, 2003

Research papers in Software Maintenance (of Prof. S. Ghoul, others papers will be provided along with the course):

1. Modelling Variability in Algorithms Design Methods - Divide and Conquer Case, IJSEIA, February 2015
2. Systems Versioning: A Features-Based Meta-modeling Approach. ONLINE SPECIAL JOURNAL ISSUES. published in International Science Index Vol:8 No:06, 2014 at www.waset.org/Publications.

3. A Genetic Methodology for Object Evolution. International Journal Of Software Engineering and Its Applications, ITSEIA, Vol.8, No. 3, 2014, http://www.sersc.org/journals/IJSEIA/vol8_no3_2014.php
4. Systems Variability Modeling: A Textual Model Mixing Class and Feature Concepts. International Journal of Computer Science & Information Technology (IJCSIT) Vol 5, No 5, October 2013, http://airccse.org/journal/ijcsit2013_curr.html
5. Aspect mining using a specification driven program slicing approach, WSEAS Transactions on Information Science and applications, 1(1), July 2004
6. T. Khammaci, ZE. Bouras, S. Ghoul, Program Slicing: Precise Shops Extraction Approaches, Hand book of Software Engineering and Knowledge Engineering, Vol. 1, Fundamentals, S. K. Chang (editor), Word Scientific Publishing, 200
7. R. Conardi and B. Westfechted, Versions Models for Software Configuration Management, ACM Computing Surveys, Vol.30, Issue 2, 1998.
8. MS Bendelloul, S. Ghoul, T. Khammaci, An object-based Decomposition for Assistance to Software Maintenance, Proc. of the 5th Magrrebian Conf. on Software Engineering and Artificial Intelligence, MCSEAI'98, Tunis, Tunisia, 1998. 30
9. MS Bendelloul, S.Ghoul, An Object System for Software Maintenance, Proc. of the 4th African Conference on Computer Science, CARI'98, Dakar, Senegal, 1998.
10. J. Estublier, S. Ghoul. Preliminary Experience with a Configuration Control System for Modular Programs, ACM SIGSOFT (USA), Vol.9, No.3, May 84. pp. 149 – 156.

Website(s):

http://ecourse.philadelphia.edu.jo/login/index.php // our e-course
http://intranet.cs.man.ac.uk/Study_subweb/Ugrad/coursenotes/CS3331/ //Manchester maintenance course