Faculty of Computers and Artificial Intelligence

Software Maintenance & Evolution

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

COURSE CODE : SCS358

L.ABOZAID@FCI-CU.EDU.EG

Software Evolution : TOC

1. Introduction to Software Maintenance & Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Program Comprehension
5. Impact Analysis
6. Refactoring
7. Reengineering
8. Legacy Information Systems
9. Reuse and Domain Engineering

What do you learn in this course?

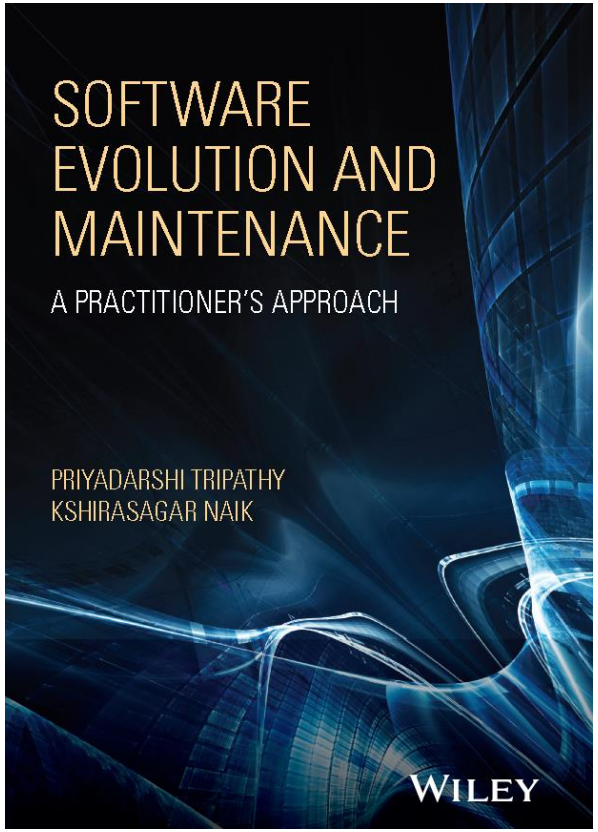
- This course will introduce you to the fundamental developments in evolution models and common maintenance practices for software.
- During the course you will learn the following topics:
  - Laws of software evolution and the means to control them
  - Evolution and maintenance models, including maintenance of commercial off the-shelf systems
  - Impact analysis and change propagation techniques
  - Program comprehension and refactoring
  - Reengineering techniques and processes for migration of legacy information systems
  - Reuse and domain engineering models

Grading System

- **Written Exam (60%)**
- **Class Assignments (40%)**
  - **Midterm (50%)**
  - **Labs (30%)**
  - **Individual Assignments (20%)**

Course Material

- Lecture Slides
- Text book: Software evolution and maintenance : a practitioner's approach; Priyadarshi Tripathy, Kshirasagar Naik. ISBN 978-0-470-60341-3
- Selected papers from the literature



What is Software Evolution?

- In 1965, Mark Halpern introduced the concept of “software evolution” to describe the growth characteristics of software

- A few years later, in 1976, Swanson introduced the term “maintenance” by grouping the maintenance activities into three basic categories: corrective, adaptive, and perfective

Software Evolution

- Evolution of software systems means creating new but related designs from existing ones.
- The objectives include supporting new functionalities, making the system perform better, and making the system run on a different operating system.
- Basically, as time passes, the stakeholders develop more knowledge about the system.
- Therefore, the system evolves in several ways.
- As time passes, not only new usages emerge, but also the users become more knowledgeable.

Software Maintenance

- Maintenance of software systems primarily means fixing bugs but preserving their functionalities.
- Maintenance tasks are very much planned.
  - bug fixing must be done and it is a planned activity.
- Unplanned activities are also necessitated.
  - a new usage of the system may emerge.
- Generally, maintenance does not involve making major changes to the architecture of the system.
- Maintenance means keeping an installed system running with no change to its design

What is Software Evolution?

- "Software evolution" and "Software maintenance" are used interchangeably.
  - However key semantic differences exist between the two.
- Lowell Jay Arthur distinguish the two terms as follows:
  - "Software maintenance means to preserve from failure or decline."
  - "Software evolution means a continuous change from lesser, simpler, or worse state to a higher or better state."
- Keith H. Bennett and Lie Xu use the term:
  - "maintenance for all post-delivery support and evolution to those driven by changes in requirements."

What is Software Evolution?

- Maintenance is considered to be set of planned activities whereas evolution concern whatever happens to a system over time.

Mehdi Jazayer's view on software evolution:

- "Over time what evolves is not the software but our knowledge about a particular type of software."

Bennett and Xu made further distinctions between the two as follows:

- All support activities carried out after delivery of software are put under the category of maintenance.
- All activities carried out to effect changes in requirements are put under the category of *evolution*.

Software Evolution- Revisited

- Lehman and his collaborators from IBM are generally credited with pioneering the research field of software evolution.
- Lehman formulated a set of observations that he called laws of evolution.
- These laws are the results of studies of the evolution of large-scale proprietary or closed source system (CSS).
- The laws concern what Lehman called E-type systems:

> "Monolithic systems produced by a team within an organization that solve a real world problem and have human users."

Software Evolution: Laws of Lehman

- *Continuing change* (1st) – A system will become progressively less satisfying to its user over time, unless it is continually adapted to meet new needs.
- *Increasing complexity* (2nd) – A system will become progressively more complex, unless work is done to explicitly reduce the complexity.
- *Self-regulation* (3rd) – The process of software evolution is self-regulating with respect to the distributions of the products and process artifacts that are produced.
- *Conservation of organizational stability* (4th) – The average effective global activity rate on an evolving system does not change over time, that is the average amount of work that goes into each release is about the same.

Software Evolution: Laws of Lehman

- Conservation of familiarity (5th) – The amount of new content in each successive release of a system tends to stay constant or decrease over time.
- Continuing growth (6th) – The amount of functionality in a system will increase over time, in order to please its users.
- Declining quality (7th) – A system will be perceived as loosing quality over time, unless its design is carefully maintained and adapted to new operational constraints.
- Feedback system (8th) – Successfully evolving a software system requires recognition that the development process is a multi-loop, multi-agent, multi-level feedback system.

Questions

?

Readings

- Book Chapter 1
  - Section 1.1, 1.2