

## **SOFTWARE EVOLUTION (SE-O-01)**

### **AIMS:**

With the advent of new architectures, the need to introduce new functionalities, the improvement in design techniques and/or changing in business objectives and process, there is a strong need to evolve existing software systems maintaining their continuity of use. Such evolution requires various techniques for what is known as '*re-engineer*'.

By re-engineering, we adopt the view of examining, understanding and altering a system with the intent of re-designing and implementing it in a new form. The process of re-engineering a system requires therefore

- the construction of a higher level abstraction of the system, a process known as *reverse engineering*; and
- the development of a new system starting from its higher level of requirement specification (i.e. *forward engineering*).

Recognising the functionality of an existing code is often seen as being the most difficult aspect of the process of re-engineering. To successfully re-engineer a system, we need to identify design decisions, intended use and domain specific details.

This module aims to introduce and critically analyze current techniques for software evolution and provides students with a practical experience using industry strength tool-set (such as FermaT).

### **LEARNING OUTCOMES:**

Upon successful completion of this module, the student will be able to:

- critically evaluate the current rationales for software evolution;
- appreciate reengineering techniques for software migration and abstraction;
- apply transformation rules to migrate time and business critical software;
- develop an integrated approach for software evolution life cycles;
- gain practical experiences on using industry-strength tools (such as FermaT).

### **SYLLABUS CONTENT:**

- Rationales and taxonomies for software evolution.
- Evolution within development life cycles.
- Lehman's laws of evolution.
- Managerial aspects of software evolution.
- Program comprehension.
- Refactoring.
- Aspect oriented software evolution.
- Wide spectrum language (WSL) and software transformation.
- Transformation theory and their implementation.
- Tools (e.g. FermaT) and software migration and abstraction.
- Open source and their evolution.

**PREREQUISITES:** None

**RECOMMENDED ASSESMENT:** Coursework and unseen paper