CO7206 System Re-engineering

Credits: *15* **Convenor:** *Neil Walkinshaw* **Semester:** *1st*

Prerequisites:	*none*		
Assessment:	*Coursework: 100%*	*Exam: 0%*	
Lectures:	*20 hours*		
Surgeries:	*7 hours*	**Private Study:**	*71.5 hours*
Laboratories:	*14 hours*		

Subject Knowledge

Aims To understand the problems and issues associated with legacy software systems and the methods used in reverse engineering, comprehending, maintaining, migrating, evolving and reengineering them.

Learning Outcomes By the end of the module, students should be able to: understand software ageing phenomenon and the issues related to it; understand the challenges in renovating and maintaining legacy software systems and the available methods for dealing with them; make reasoned decisions on which reengineering methods to apply for certain types of legacy system renovation tasks.

Methods Class sessions, tutorials and lab sessions together with course notes, course readings, assignments and class tests.

Assessment Three assignments.

Skills

Aims To develop analytical and problem solving skills in dealing with legacy systems and software integration challenges. To develop on hands experience in reverse engineering and reengineering existing software systems.

Learning Outcomes By the end of the module, students will be able to apply the methods learned to assess the situation of a small-scale legacy system and decide a suitable reengineering strategy for it, in the light of the objectives of the reengineering/renovation effort. They will learn how to reverse engineer and reengineer moderate size legacy software systems using some of the available commercial and research tools.

Methods Class sessions, tutorials, lab sessions and industrial tutorials by industry experts together with course notes, course readings, assignments and class tests.

Assessment Three assignments.

Explanation of Prerequisites

This module assumes that the student has reasonable programming experience in more than one high level language, preferably Java, C#, or C++. It assumes that the student has done some non-trivial programming tasks and has some understanding of the challenge inherent in trying to understand and modify old software or software that was written by other people.

Course Description

Software development is not always a ”green-fields” process. More often than not, new software engineers are hired to maintain and evolve existing systems, not to develop new ones. If a new system is to be developed, it has to be integrated with other existing ”legacy” software systems. Legacy systems are valuable software systems that are still in use but are difficult to maintain, change or migrate because they were developed with technologies of the past and/or because they were not engineered properly. Very often, these systems were developed without proper documentation, version control, or proper design. Many such systems had undergone numerous changes by different people that violate the original system design, if any ever existed. As a result, it is challenging to understand, modify or migrate these systems. Fresh software developers are usually neither equipped with the necessary skills nor have the desire to work with these software ”legacies”. Fresh software development is usually considered superior to software maintenance and reengineering. The year 2000

problem and the deployment of the Euro gave rise to research and practice of software system reverse engineering and reengineering. Reengineering is ”the examination of a subject system to reconstitute it in a new form and the subsequent implementation of the new form.” [Chikofsky, and Cross II, 1990]. Part of any reengineering efforts is a reverse engineering process, which is ”the process of analyzing a subject system with two goals in mind: (1) to identify the system’s components and their interrelationships; and, (2) to create representations of the system in another form or at a higher level of abstraction.” [Chikofsky, and Cross II, 1990].

In the Internet era, it is very important to have the skills to deal with legacy systems because it is not always the case that Web applications will be developed from scratch. In many cases it is required to open the available information systems to Web access or integrate them with other Web applications.

This module is an introduction to the main issues related to software systems ageing and evolution. We will examine some of the available methods and technologies for software reverse engineering and reengineering as well as some of the managerial and planning issues specific to software reengineering projects.

Detailed Syllabus This module will cover the following topics.

Software Deterioration: How and why software systems deteriorate.

Software Maintenance and Reengineering: What are the challenges involved in re-engineering a system.

Software Comprehension: Basic skills of manually analysing and comprehending the functionality of a complex, unfamiliar system.

Automated and Semi-Automated Software Analysis: Learning basic static and dynamic program analysis techniques.

Assessing Software Vulernabilities and Weaknesses: What are the sources of problems in a software system? Which areas need to be re-engineered most urgently?

Regression Testing: Essential testing skills to make sure that re-engineering does not introduce any new bugs.

Restructuring and Migration: Basic strategies to restructure and migrate a software system.

Reading List

[A] Neil Walkinshaw,, *Reverse Engineering Software Behaviour,* Elsevier Advances in Computers, vol 91, 2013.

[A] Oscar Nierstrasz, Stéphane Ducasse, Serge Demeyer,, *Object-Oriented Reengineering Patterns,* Morgan Kaufmann Publishers, 2003.

[A] Elliot Chikofsky and James Cross, *Reverse Engineering and Design Recovery: A Taxonomy,* IEEE Software 7(1):13-17, 1990.

Resources Course notes.

Module Evaluation Course questionnaires, course review.