Reengineering Legacy Systems Towards New Technologies

Djelloul Bouchiha
Center University of Naama, Algeria & EEDIS Lab. UDL-SBA, Algeria

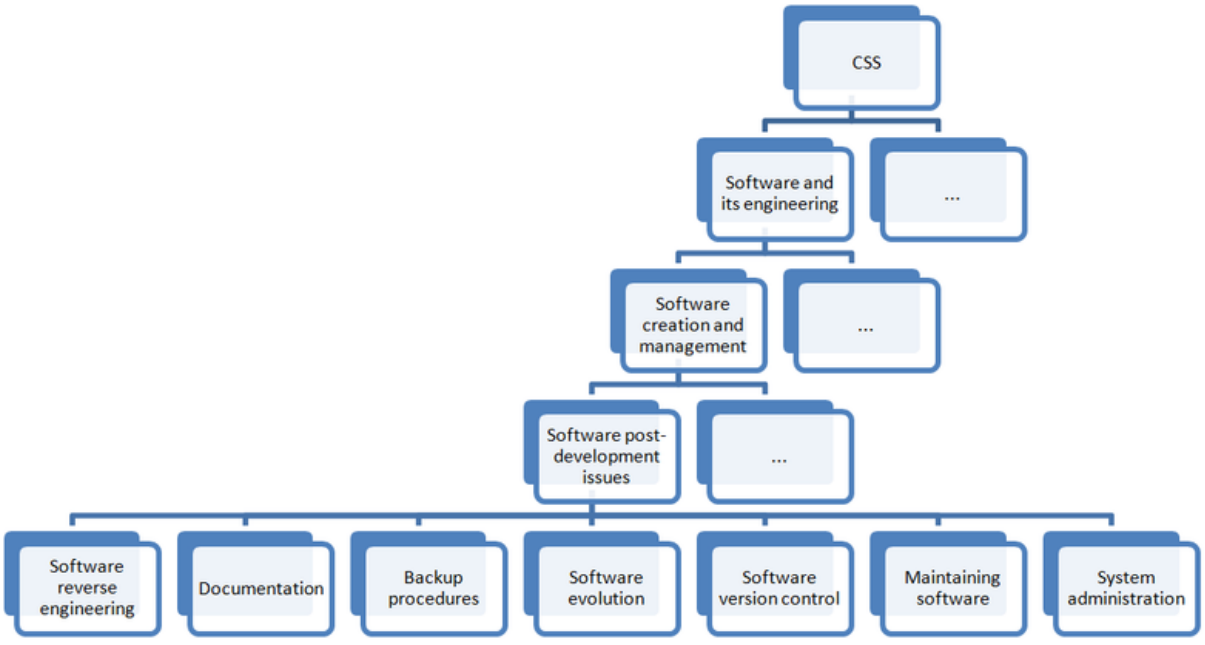
INTRODUCTION

Software reengineering is an important area of the software engineering. The quest to maintain and understand operational legacy systems has always been a challenge for software practitioners. This chapter presents a compilation of notions and techniques covering major areas namely, reverse engineering, program understanding, software maintenance, migration and evolving. Our objective is not to create new terms, but to introduce the terms already in use with different perspectives.

Before starting with the reengineering of legacy systems towards new technologies, we must know: in which research field are we working? Which research community do we belong to? What research issue are we dealing with?

To answer all these questions, we use the ACM Computing Classification System (CCS). It has been built as a hierarchical ontology that can be used in semantic Web applications. It relies on a semantic terminology as the main source of concepts and categories that present the state of the art of the computing discipline. The 2012 ACM CCS (ACM-CCS, 2012) starts with thirteen computing fields. Browsing this hierarchy to attempt our goal gives us the following tree.

Figure 1. Reengineering issues in the ACM CSS classification



DOI: 10.4018/978-1-7998-3479-3.ch084

Leaves of the presented tree in Figure 1 are the sub-issues in relation to the reengineering problem, according to the ACM CSS classification.

The following sections give more details about the reengineering process, notably prerequisites, model, problem statement, taxonomy, techniques and future trends. We also list some specialized conferences and journals. Finally, we give conclusion and advices to the researchers belonging to the software engineering community.

BACKGROUND

This chapter is titled "Reengineering Legacy Systems towards new Technologies". Here, the term "System" is used to say "software system" not the hardware one. "Legacy systems" are systems currently in use, and have been often left to run for a long time without too many significant changes. "New technologies" are the latest (current) technologies.

Reengineering is discussed in this chapter as sub-issue of Software Engineering which is described as the application of engineering techniques to produce software that runs properly and is constructed in an efficient manner. The produced software should be efficient, reliable, modifiable, testable, reusable, maintainable, portable, interoperable and correct. Software engineering process consists of a set of activities, known as the software lifecycle. Principal activities are: requirements analysis, design, implementing and testing the software. Requirements analysis phase consists in determining precisely what the functionality of the system will be. The design phase consists in taking the requirements and devising a representation to allow the requirements to be translated into source code. The implementing activity is the most familiar to students that consists in writing the program source code. Software testing consists in checking the software before it is judged ready to be released to its customer.

Reengineering is the study and analysis of an existing system for purposes of understanding, maintenance, or migration towards new technologies (Chikofsky & Cross, 1990). Based on the software lifecycle described above, the next section presents a general model for the software reengineering process.

REENGINEERING MODEL

We propose, in Figure 2, a general model for software reengineering that depicts the sub-processes for all levels of the reengineering process.

Figure 2. General model for software reengineering