Chapter 1
A Brief Overview of Software Process Models:
Benefits, Limitations, and Application in Practice

Sanjay Misra
Covenant University, Nigeria

Martha Omorodion
Federal University of Technology, Nigeria

Luis Fernández-Sanz
Universidad de Alcalá, Spain

Carmen Pages
Universidad de Alcalá, Spain

ABSTRACT

Software process development in software engineering does not seem to offer a solid view of what they have in reality. Although many models have already been developed, recommended, or even used in the industry, these proposals have still not been able to come to terms with what is available. This chapter evaluates the benefits and limitations of some of the software development models while offering a comparative analysis and data on their real usage. In particular, an attempt has been made to evaluate the problems and strengths of a good variety of software process models and methodologies. Some conclusions and lines of future research are presented.

1. INTRODUCTION

In our fast-paced society, judging from the recent technological advances within the past two decades and the rate at which they become obsolete, one is tempted to ask, "How do they come up with such ideas? What does it take to move from one stage to the next? How does the developer suddenly feel that

DOI: 10.4018/978-1-5225-3923-0.ch001

the next best thing since sliced bread that customers want is a car with GPS (global positioning system) and not just that, it talks to you as well?"

The basis of all these developments fits into a single word, "software." Within the past two decades this abstract concept has consumed a vital part of our society and on a regular basis. We all demand improvements, systems that are more sophisticated than what we already have, features and services that are almost insurmountable to the layman. Yet at the end of the day, we get them through systems we often refer to as "computerized."

Computerized systems range from laptops, air traffic control systems to even architecture and building operations and trading. Achieving all these involve a lot of complex activities. Occasionally, problems arise within our components that are software-based. These problems did not appear out of the blue. In some cases, they were embedded in the systems during the development stages. This is due, in part, to the fact that these systems and their applications are products of complex activities that are really very difficult to develop, integrate and test.

The aim of this paper is to give insight into the various activities technically known as software processes and how they are organized into different models to achieve the sophisticated software-based products that we have today. Also, focus will be placed on highlighting the benefits and limitations of each model and recommendations made for improvements on what is available to ease the problems involved in the production of these software-based systems, while creating better outputs in the process.

Section two of this paper deals with the definition of terms and a list of other software development processes that are not evaluated in this paper; section three shows the comparative analysis of the processes that are evaluated. Section four outlines their benefits and limitations in tables. The recommendations made are quite elementary but are useful for consideration when designing a process model.

1.1 Motivation for This Chapter

While teaching software engineering courses to undergraduates and masters students for the past six years, one of the authors has observed that students are often confused about the different software process models. Each book on software engineering explains these models in its own way. For example, in the well-known book "Software Engineering" by (Sommerville, 2010), the author describes the models under three categories: waterfall, iterative and CBSE. However, we do not feel comfortable considering CBSE as a model because it is essentially based on the development of components and very close to object-oriented development. Furthermore, most of the available models are similar in many ways, which make them difficult to discriminate from one another. For example, spiral, iterative/evolutionary models (waterfall, incremental, spiral models, prototyping, Cleanroom, and object-oriented techniques) show similarities in their processes. Additionally, several new software process models/methodologies based on the basic models have been developed and applied in the industries in the last decade. There is lack of clear-cut explanation and discrimination between these models in the literature except for the basic ones. This contributes to insufficient knowledge in the learning community regarding these models. To the best of our knowledge, no existing research paper available has provided the explanation and discrimination of these models in a clear way. In this paper we evaluate and explain each available model and present them to the learning community in a more comprehensible way to help discriminate between them easily.