Chapter 1
Lifecycles:
Organizing Development Phases

ABSTRACT

This chapter discusses lifecycle model application for software development. It compares the benefits and shortcomings of different models. The authors argue that there is no universal lifecycle model. For agility, this chapter recommends combining prototyping with the other models. The authors suggest this to achieve a common understanding of the key product features and to reduce project risks. The lifecycle model choice determines project economics and time to market. The model also influences product quality and overall project success. However, product success essentially depends on human factors. The authors analyze the applicability of the lifecycle models to large-scale, mission-critical software systems. Finally, this chapter introduces a methodology. It includes a spiral-like lifecycle and a set of formal models and visual tools for software product development. This methodology helps to optimize the software product lifecycle. It fits large-scale, complex heterogeneous software products.

INTRODUCTION

This chapter describes lifecycle model application to software development. It includes a discussion of their benefits and shortcomings. It analyses the applicability of the models to large-scale, mission-critical software systems. In a limited agility situation, this is essential.

DOI: 10.4018/978-1-5225-5589-6.ch001

Some of the models are more straightforward. Others require a number of iterations. The investigation of the models concludes that there is no "silver bullet" for lifecycle models. However, the authors arrive at a set of recommendations to adjust the models for agile software development.

Project success is not solely determined by a lifecycle model or by their combination. It also depends on a number of human factors. The causes may help or hinder a common understanding of the key product features by the client and the developer. The authors give more details on the human-related factors in Chapter 6.

To optimize the software product lifecycle, the authors introduce a methodology. The software lifecycle is mission-critical for agility. This approach includes a spiral-like lifecycle and a set of formal models and visual computer-aided tools.

This chapter is organized as follows. The section called "Simple Lifecycles" discusses the abbreviated and straightforward lifecycle models. They include build-and-fix and waterfall. The section called "Evolutionary Lifecycles" presents an overview of evolutionary models. The section called "Risk-Driven Lifecycles" describes risk-driven software lifecycle models. These are spiral, synchronize and stabilize, and object-oriented. The section called "Customizing Lifecycles" contains an overview of an enhanced software development methodology. It provides lifecycle optimization and elaborates the agile deliverables for mission-critical software products. The conclusion summarizes the chapter results.

SIMPLE LIFECYCLES

One of the lifecycle models is the build-and-fix (see Figure 1). This is a model of an incomplete lifecycle. The build-and-fix is too simple for large-scale projects. Large-scale typically means above 100 KLOC; however, the authors recommend using this model for the projects below one KLOC. The build-and-fix model may be a possible option for a small-scale software solution. This downsized solution lacks agility. Therefore, it only applies for a very small product with clear requirements.

The other model the authors discuss here is rapid prototyping (see Figures. 4 and 5). It is also somewhat limited. The fact that it includes all the basic stages of the lifecycle does not change its limitations. These stages are the analysis and specification of requirements, preliminary and detailed design, implementation, unit testing, integration, product testing, maintenance, and