IRM PRESS
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.irm-press.com
ITB9265

Chapter XIX

A RUP-Based Software Process Supporting Progressive Implementation

Tiago Lima Massoni
Universidade Federal de Pernambuco (UFPE), Brazil

Augusto Cesar Alves Sampaio
Universidade Federal de Pernambuco (UFPE), Brazil

Paulo Henrique Monteiro Borba
Universidade Federal de Pernambuco (UFPE), Brazil

ABSTRACT

This chapter introduces an extension of the Rational Unified Process (RUP) with a method that supports the progressive, and separate, implementation of three different aspects: persistence, distribution, and concurrence control. This complements RUP with a specific implementation method, called Progressive Implementation Method (Pim), and helps to tame the complexity of applications that are persistent, distributed, and concurrent. By gradually and separately implementing, testing, and validating such applications, we obtain two major benefits: the impact caused by the requirements changes during development is reduced and testing and debugging are simplified. In addition, the authors hope to contribute to solving the lack of a specific implementation method in RUP.

INTRODUCTION

Software development has become a more complex activity over the last years. Clients have been increasingly demanding higher productivity, better software quality, and shorter time to market. Additional strain results from new, common requirements

such as distribution and concurrent access. These non-functional issues complicate implementation, testing, and particularly, maintenance activities. Most human and financial resources are driven to maintenance activities (Pressman, 1997).

Industrial software processes, such as the Rational Unified Process (RUP), can be useful in dealing with this complexity. A software process defines a set of software construction, validation, and maintenance activities in order to discipline the overall software development practices in an organization. This is particularly true for RUP, which has been widely adopted by major software organizations (Ambler, 1999). It is a highly comprehensive and detailed process, yet focuses on requirements, analysis, and design activities.

Regarding the coding process, implementation methods are important to address the complexity of design decisions during coding activities, especially for non-functional concerns. In order to simplify those activities, we argue that it is useful to tackle functional and non-functional concerns separately. In fact, whereas architectural and design activities should jointly consider functional and non-functional concerns (Waldo et al., 1997), implementation activities can benefit from the separation of the two concerns.

In this context, an implementation method might help programmers to effectively achieve this separation. Therefore, we have defined the Progressive Implementation Method (Pim) (Borba et al., 1999), supporting a progressive approach for object-oriented implementation in Java (Gosgling et al.,1996) where persistence, distribution, and concurrence control aspects are not initially considered in the coding activities, but are gradually introduced. In this way, we can significantly reduce the impact caused by requirements changes during development and tame the complexity by gradually implementing and testing different aspects of code.

This progressive approach is possible because this method relies on the use of design patterns that provide a certain degree of modularity and separation of concerns (Parnas et al., 1972) in such a way that the different aspects can be implemented separately. However, other techniques and tools for separation of concerns might just as well be used, such as aspect-oriented programming (Kiczales et al., 1997).

The objectives of this chapter are:

- Present the basic concepts of the Progressive Implementation Method (Pim), defining a clear basis for our main goal, i.e., the RUP extension.
- Define the software process resulting from the inclusion of the method into RUP, providing proper implementation guidelines for RUP. Thus, we hope to support the progressive implementation of different aspects in software development projects where disciplined requirements, design, and test activities are essential (demanding a software process).

In the next sections, we present the main concepts of RUP and Pim, which are useful for a better understanding of our solution. We also outline the definition of RUPim, i.e., the proposed extension of RUP, and present some results obtained in simple practical experiments using RUPim. Finally, we present some conclusions, future trends, and related work.