

Chapter 45

A Method to Design a Software Process Architecture in a Multimodel Environment: An Overview

Mery Pesantes

Research Centre in Mathematics (CIMAT, A.C.), Mexico

Jorge Luis Risco Becerra

University of São Paulo – Escola Politécnica, Brazil

Cuauhtémoc Lemus

Research Centre in Mathematics (CIMAT, A.C.), Mexico

ABSTRACT

In the multimodel improvement context, Software Organizations need to incorporate into their processes different practices from several improvement technologies simultaneously (i.e. CMMI, PSP, ISO 15504, and others). Over the last few years, software process architectures have been considered a means to harmonize these technologies. However, it is unclear how to design a software process architecture supporting a multimodel environment. In this chapter, an overview of the method to design a software process architecture is presented, identifying basic concepts, views, phases, activities, and artifacts. In addition, important aspects in the creation of this method are explained. This method will assist process stakeholders in the design, documentation, and maintenance of their software process architecture.

1. INTRODUCTION

Multimodel Software Process Improvement (MSPI) aims to achieve business goals, develop quality products through a mature process applying multiple improvement technologies best practices simultaneously, and reduce time-to-market and production costs (Siviy, Penn, & Stoddard, 2008; Unterkalmsteiner, Gorschek, Islam, Cheng, Permadi, & Feldt, 2012). Therefore, software organizations are analyzing their processes, selecting appropriate improvement technologies and adopting best practices from each technology.

DOI: 10.4018/978-1-5225-3923-0.ch045

Problems have arisen within organizations working under this multimodel environment (Kelemen, Kusters, & Trienekens, 2011), where multiple technologies, which may be used in different ways, address the same need with significant overlap. Therefore, the decision to simultaneously adopt multiple technologies can be complex and can depend as much on how they will be implemented as on their specific features and benefits.

The need to harmonize technologies emerges as a solution toward working simultaneously with multiple improvement technologies (Kirwan, Marino, Morley, & Sivi, 2008a; Lawrence, 2009; Pardo, 2010). Currently, there are many harmonization approaches (Calvache, Pino, García, & Piattini, 2009; Kirwan et al., 2008a), methods and techniques (Halvorsen & Conradi, 2001; Mutafelija & Stromberg, 2003; Wang & King, 2000). Some techniques, such as mapping and comparison, are widely used but many other techniques have not yet been clearly defined, making harmonization of multiple technologies a difficult endeavor for organizations.

Software Process Architectures have been recognized as a means to harmonize multiple technologies within an organization that develops software products (Kirwan et al., 2008a; Kirwan, Marino, Morley, & Sivi, 2008b). Software process architecture in a multimodel environment is defined as “a set of process elements and its relationships that support adding, removing or modifying any improvement technology and allowing it to be derived from standard processes” (Pesantes, Lemus, Mitre, & Mejia, 2012a).

Several methods have been published to address the problem of how to design a process architecture (Borsoi & Becerra, 2008; Dai, Li, Zhao, Yu, & Huang, 2008; Green & Ould, 1996; Maldonado & Velázquez, 2006). However, it is unclear how to design a software process architecture that supports a multimodel software process improvement environment.

This research presents a method to design a software process architecture that supports a multimodel environment. This method considers creating a software process architecture that will receive as input a set of harmonized heterogeneous technologies and obtain as output a set of standard processes. It is based on a statistical thinking approach, analysis method and internal structured analysis technique.

Accordingly, the contents herein are structured as follows: section 2 presents a background of available efforts regarding methods to design a process architecture. Section 3 presents important aspects considered to create the method. Section 4 describes the basic concepts of the method. Section 5 shows the basic constructors of a software process architecture. Section 6 gives a general description of the method. Section 7 describes the method's phases, with their respective activities and artifacts. The last section summarizes conclusions and future works of this research.

1.1 Background

Today, researchers are concerned with understanding and improving the quality of software, which is being used in a variety of areas and applications and becoming more complex as the functionality required to provide services is evolving. As software increases in usage, complexity and size, the cost of building and maintaining it has increased as well. Software exhibits unexpected and undesirable behaviors that may even cause severe problems and damage that affect its quality. Hence, the software process approach has emerged to address these concerns and, recently, the research area of process architecture is emerging with it.

The software process approach is centered on the process through which software is developed. A software process is defined as “the set of partially ordered steps used to develop or enhance a software product” (Feiler & Humphrey, 1993). This approach is based on the assumption that there is a direct

23 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:

www.igi-global.com/chapter/a-method-to-design-a-software-process-architecture-in-a-multimodel-environment/192916

Related Content

India to China – Repurposing Learning Software across Cultures: Positioning an E-Learning Framework of a Technical Library Program for Success

Margaret Strong, Bobby Joy, Madhukar Pulluru, Tenya Dong and Edward Zhou (2012). *Computer Engineering: Concepts, Methodologies, Tools and Applications* (pp. 1099-1114).

www.irma-international.org/chapter/india-china-repurposing-learning-software/62500

DQ Based Methods: Theory and Application to Engineering and Physical Sciences

Stefania Tomasiello (2012). *Handbook of Research on Computational Science and Engineering: Theory and Practice* (pp. 316-346).

www.irma-international.org/chapter/based-methods-theory-application-engineering/60366

Development of Controllers Using Simulink and Contract-Based Design

Pontus Boström, Mikko Huova, Marta (Plaska) Olszewska, Matti Linjama, Mikko Heikkilä, Kaisa Sere and Marina Waldén (2012). *Dependability and Computer Engineering: Concepts for Software-Intensive Systems* (pp. 151-169).

www.irma-international.org/chapter/development-controllers-using-simulink-contract/55328

Threats Classification: State of the Art

Mouna Jouini and Latifa Ben Arfa Rabai (2018). *Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications* (pp. 1851-1876).

www.irma-international.org/chapter/threats-classification/192950

Improvement of RSM Prediction and Optimization by Using Box-Cox Transformation: Separation of Colloidal Contaminants From Mineral Processing Effluents via Electrocoagulation

Mustafa Çrak (2018). *Handbook of Research on Predictive Modeling and Optimization Methods in Science and Engineering* (pp. 156-191).

www.irma-international.org/chapter/improvement-of-rsm-prediction-and-optimization-by-using-box-cox-transformation/206749