Chapter 13
Software Development Governance: A Case Study for Tools Integration

Nagehan Pala Er
ASELSAN Microelectronics, Guidance and Electro-Optics Division, Turkey

Cengiz Erbaş
ASELSAN Microelectronics, Guidance and Electro-Optics Division, Turkey

Bahar Çelikkol Erbaş
TOBB University of Economics and Technology, Turkey

ABSTRACT
Software development governance can be defined as the application of “governance” in software engineering in order to increase the probability of success in the level of individual projects as well as in the level of the organization comprising many interrelated projects. The topic deserves an interdisciplinary perspective, as the general subject of governance has been analyzed quite extensively under the field of Transaction Cost Economics. This interdisciplinary approach enabled the identification of three main modes of governance for software engineering, namely: top-down governance, bottom-up governance and reuse governance, each having unique transaction cost characteristics. To be cost effective, (1) the organizations should adapt the right governance structure for their projects based on their characteristics, and (2) the software development tools should support and be in alignment with the underlying governance structure. In this chapter, we briefly overview the first premise and then outline an approach to address the second premise, specifically tackling the issue of tools integration, for software modeling and configuration management tools. We use Dependency Structure Matrix (DSM) to represent the mapping between governance structures and software modules, propose a configuration management approach for each mode of software governance, and demonstrate a successful integration using Lattix LDM, IBM Rational ClearCase and IBM Rational Rhapsody, three broadly available tools in the software industry.
INTRODUCTION

Business success or failure for a software development organization is determined by its competitiveness in the marketplace in recognizing and fulfilling the customer needs. Independent of the characteristics of the underlying business, the success or failure is influenced by the organization’s ability to accomplish not only individual projects - given particular cost, duration and quality constraints; but also an undetermined number of existing and future projects given their collective performance targets under uncertainty. The amount of uncertainty varies based on the maturity of the domain and the technologies involved. Software development governance, a new field which applies the economics intuition from an interdisciplinary literature, is promising to better understand the issues involved in business success in software engineering.

In essence, the objective of software development governance is to guide the software engineering organization so that it produces value that aligns with the needs of the underlying business (Kofman, Yaeli, Klinger, & Tarr, 2009). Software development governance intends to establish and maintain efficient and effective operations and workable arrangements among the parties involved in software development. It provides an integrated view, as opposed to the dominant functional and structural perspectives for managing the management of this particular domain (Bannerman, 2009; Ambler, 2009; Cataldo & Herbsleb, 2009; Anderson & Carney, 2009). Given these broad objectives, it is not unexpected that the field is building on an interdisciplinary foundation, benefitting primarily from the prior work on the general topic of “governance” in the Transaction Cost Economics (TCE) literature (Erbas & Erbas, 2005).

TCE states that efficient governance structure depends on the characteristics of the transaction, and places a central role to bounded rationality and opportunism when analyzing which governance structure should be preferred for a given situation. Considering the wicked nature of large-scale software development, it is evident that bounded rationality and opportunism play a central role also in software development. Applying this TCE perspective, three primary categories of approaches for governance were identified: one based on top-down (planned), another based on bottom-up (adaptive) processes, and another based on software reuse (Erbas & Erbas, 2009). It was demonstrated that in order to minimize transaction costs for a particular project, software organizations should select the governance structure based on certain project characteristics, such as asset specificity, uncertainty and frequency. It was further suggested that the software development and management tools should support and be in alignment with the selected governance structure.

Software projects in an organizational setting may have interdependencies for many reasons, such as due to shared functionality, delivery schedule, personnel availability, leveraged technology, underlying platforms and/or utilized tools. Among the sources of such interdependencies, uncertainties over module and interface specifications, as well as their impact on the development and reuse of shared functionality and resources, are critical subjects from the software development governance perspective. A successful implementation of governance in software development requires a governance framework with an integrated tools support at many different levels, including requirements management, role assignment, software modeling/code generation, configuration management and policy enforcement. Even though, there is some work performed related to tools integration, such as Kofman, Yaeli, Klinger, and Tarr (2009) for automated role assignment, this is an area unexplored for the most part (Cheng, Jansen, & Remmers, 2009; Lehto & Rautiainen, 2009; Talby & Dubinsky, 2009).

In this chapter, we first provide a brief overview of the prior work in software development governance from (Erbas & Erbas, 2009), and