Chapter 15
Providing Automated Holistic Process and Knowledge Assistance During Software Modernization

Gregor Grambow
Ulm University, Germany

Roy Oberhauser
Aalen University, Germany

Manfred Reichert
Ulm University, Germany

ABSTRACT

Software modernization remains a difficult, highly intellectual, labor-intensive, collaborative, and risky undertaking involving software engineers interacting in knowledge-centric processes. While many tools and several methodologies are available, current modernization projects lack adequate automated and systematic operational process support. This chapter provides an introduction to the topic of automated process and knowledge assistance for software modernization, giving background information on related work in this area, and then expounds on various problems. To address these, a holistic solution approach and guidance framework called the Context-Aware Software Engineering Environment Event-Driven Framework (CoSEEEK) is described, which can support developers on software modernization projects, addressing such aspects as process dynamicity, extrinsic processes, process exception handling, coordination, quality assurance, and knowledge provisioning. Subsequently, future research directions are discussed and a conclusion is drawn.

DOI: 10.4018/978-1-5225-3923-0.ch015

INTRODUCTION

Software applications are continuing to grow in size, with one recent study indicating that applications typically double in size as measured in lines of code every 4-5 years (van Genuchten & Hatton, 2012). Size growth results in increasing complexity and an increase in the total number of defects (Koru, Zhang, El Emam, & Liu, 2009). Over time, legacy applications continue to face additional challenges (Mens et al., 2005), including increasing software maintenance costs, quality assurance issues, and architectural and technological erosion (Ducasse & Pollet, 2009; Tan & Mookerjee, 2005). Software modernization is challenged with modernizing the language and/or technology used in a legacy application while retaining its previous functionality, and as such has a key place within the Software Maintenance knowledge area of the Software Engineering Body of Knowledge (SWEBOK) (Abran & Bourque, 2004). Yet many reengineering efforts fail because they are unsystematically executed using ad-hoc processes (Weiderman, Bergey, Smith, & Tilley, 1997). Various software modernization strategies, processes, methodologies, and techniques have been developed over the years, and the appropriate selection, tailoring, and application to a given modernization situation is dependent on many factors, including organizational constraints, budget, time, and resource constraints, experience, available competencies, risk, urgency, criticality, etc. (ISO/IEC 14764, 2006). Aspects involved in modernization can include business process archaeology, process mining, business process modeling, reverse engineering, software architecture, model-driven software engineering, knowledge management, and static and dynamic code analysis.

Software modernization remains a risky, difficult, and highly intellectual process incorporating various automated and manual tasks performed by specialized knowledge workers. The process lacks sufficient automated and systematic operational process support. The intangibility of the product represented in its source code artifacts poses a significant comprehension challenge to software engineers. To support them in their modernization tasks, various tools such as tools related to Model-Driven Architecture™ (MDA) (Mukerji & Miller, 2003) from the Object Management Group (OMG), and various processes such as the (ISO/IEC 14764, 2006), Service-Oriented Migration and Reuse Technique (SMART) (Lewis, Morris, & Smith, 2005), and eXtreme end-User dRiven Process (XIRUP) ("D31b", 2008) from the MOdel driven MOdernisation of Complex Systems (MOMOCS)[1] project can be used. Mostly, the former are utilized for supporting the project participants' individual tasks while the latter aim to help organize the necessary collaboration and sequencing of work. However, this lack of integration may also be a crucial obstacle for holistic and technically supported process enactment in the software engineering (SE) domain: On the one hand, projects and their processes are planned using abstract process models, on the other hand the concrete tasks are executed by different people working with different software tools and techniques in order to manipulate a large number of different artifacts (e.g., specifications, user documents, source code artifacts, or tests). The tools are crucial for the successful completion of the different tasks, yet their usage is neither directly connected to the process nor to each other. This promotes an ever-growing gap between the planned and the actually executed process. This disparity has negative effects for modernization projects and their produced software: First of all, proactive and reactive software quality techniques are often not systematically and satisfactorily integrated, since the actually executed operation process is unknown. Project planning with management tools (e.g., in visual form such as a Gantt chart) typically does not provide process governance nor process execution support. Since quality assurance is not systematically and automatically integrated, important quality actions (perhaps involving tools for static code analysis, profiling, or test remediation) may not be executed in a timely fashion, and quality deterioration may go undetected.