Event-Driven Exception Handling for Software Engineering Processes

Gregor Grambow[1], Roy Oberhauser[1], and Manfred Reichert[2]

[1] Computer Science Dept., Aalen University
{gregor.grambow,roy.oberhauser}@htw-aalen.de
[2]Institute for Databases and Information Systems, Ulm University, Germany
manfred.reichert@uni-ulm.de

Abstract. In software development projects, process execution typically lacks automated guidance and support, and process models remain rather abstract. The environment is sufficiently dynamic that unforeseen situations can occur due to various events that lead to potential aberrations and process governance issues. To alleviate this problem, a dynamic exception handling approach for software engineering processes is presented that incorporates event detection and processing facilities and semantic classification capabilities with a dynamic process-aware information system. A scenario is used to illustrate how this approach supports exception handling with different levels of available contextual knowledge in concordance with software engineering environment relations to the development process and the inherent dynamicity of such relations.

Keywords: Complex event processing; semantic processing; event-driven business processes; process-aware information systems; process-centered software engineering environments.

1 Introduction

The development of software is a very dynamic and highly intellectual process that strongly depends on a variety of environmental factors as well as individuals and their effective collaboration. In contrast to industrial production processes that are highly repetitive and more predictable, software engineering processes have hitherto hardly been considered for automation. Existing software engineering (SE) process models like VM-XT [1] or the open Unified Process [2] are rather abstract (of necessity for greater applicability) and thus do not really reach the executing persons at the operational level [3]. In sparsely governed processes without automated data assimilation and process extraction, deviations from the planned process, exceptions, or even errors often remain undetected. Even if detected, an automated and effective exception handling is hard to find.

To increase the level of standardization (i.e., usage, repeatability, conformance, etc.) of process execution, automated support for SE processes is desirable. To enable this in a holistic way, an automated solution should be capable of some kind of

F. Daniel et al. (Eds.): BPM 2011 Workshops, Part I, LNBIP 99, pp. 414–426, 2012.

process exception handling so that the occurrence of exceptions does not deteriorate process performance. Further, automated process exception support will only be acceptable if it is not too complex or more cumbersome than manual handling [4]. Automated handling implies automated detection of exceptions that depends on the capabilities of the system managing the processes [5]. However, existing process-aware information systems (PAIS) are still rather limited regarding detection and handling of exceptions [6]. Exceptions can arise for reasons such as constraint violations, deadline expiration, activity failures, or discrepancies between the real world and the modeled process [7]. Especially in the highly dynamic SE process domain, exceptions can arise from various sources, and it can be difficult to distinguish between anticipated and unanticipated exceptions. Even if they are detected, it can be difficult to directly correlate them to a simple exception handler. Due to its high dynamicity, SE has been selected as first application domain, but the generic concept can also be applied to other domains.

Two fictional scenarios from the SE domain illustrate the issues:

- Scenario 1 (Bug fixing): In applying a bug fix to a source code file, the removal of a known defect might unintentionally introduce other problems to that file. E.g., source code complexity might increase if multiple people applied "quick and dirty" fixes. Thus, the understandability and maintainability of that file might drop dramatically and raise the probability of further defects.
- Scenario 2 (Process deviation): In developing new software, the process prescribes the development and execution of a unit test to aid the quality of the produced code. For various reasons, the developer omits these activities and integrates the produced code into the system. This could eventually negatively affect the quality of that system.

These scenarios demonstrate the various challenges an automated process exception handling approach for SE faces: Exceptions can arise relating to various items such as activities, artifacts, or the process itself. Many of these exceptions may be difficult to detect, especially for a PAIS without direct knowledge of the environment. It may also be unclear when exactly to handle the exception and who should be responsible. Generally, the knowledge about the exception can vary greatly, making unified handling difficult and the application of standardized exception handlers unsuitable. Both of the aforementioned scenarios will be used to show the applicability of our approach to SE processes and their exception handling.

The remainder of this paper is organized as follows: Section 2 introduces the novel exception handling approach, followed by Section 3 showing its technical realization. An application scenario is presented in Section 4 and related work is discussed in Section 5. Finally, Section 6 presents the conclusion.

2 Flexible Exception Handling

To respond to the special properties of dynamic SE process execution, this paper proposes an advanced process exception handling approach. It is grounded on two properties: the ability to automatically gather contextual information utilizing special sensors and complex event processing; and second, an enhanced flexibility in the