Flexibility in Process-Aware Information Systems

Manfred Reichert, Stefanie Rinderle-Ma, and Peter Dadam

Institute of Databases and Information Systems, Ulm University, Germany
{manfred.reichert,stefanie.rinderle,peter.dadam}@uni-ulm.de

Abstract. Process-aware information systems (PAIS) must be able to deal with uncertainty, exceptional situations, and environmental changes. Needed business agility is often hindered by the lacking flexibility of existing PAIS. Once a process is implemented, its logic cannot be adapted or refined anymore. This often leads to rigid behavior or gaps between real-world processes and implemented ones. In response to this drawback, adaptive PAIS have emerged, which allow to dynamically adapt or evolve the structure of process models under execution. This paper deals with fundamental challenges related to structural process changes, discusses how existing approaches deal with them, and shows how the various problems have been exterminated in ADEPT2 change framework. We also survey existing approaches fostering flexible process support.

1 Introduction

In many application domains process-aware information systems (PAIS) will be not accepted by users if rigidity comes with them [1234]. Instead, it should be possible to quickly implement new processes, to enable on-the-fly adaptations of running ones, to defer decisions regarding the exact process logic to runtime, and to evolve implemented processes over time. Consequently, process flexibility has been identified as one of the fundamental needs for any PAIS and different enabling technologies have emerged [5678]. They support adaptive processes [91011], declarative models [7], late modeling [1213], and data-driven processes [1415]. Basically, we need to be able to deal with uncertainty, to cope with exceptions, and to evolve processes over time:

- **Ability to deal with uncertainty.** The implemented process is based on a loosely or partially specified model, where the full specification is made during runtime and may be unique to each process instance. Rather than enforcing control through a rigid, or highly prescriptive model, that attempts to capture every aspect, the model is defined in a more declarative or incomplete way that allows individual instances to determine their own processes.

- **Ability to adapt processes.** The implemented process is able to react to exceptions, which may or may not be foreseen and which affect one or a few instances. Generally, it must be possible to adapt the structure and/or state of the process model of a particular instance. Respective adaptations, however, must not affect other instances being executed on this model as well.
— *Ability to evolve processes.* A process model has to be changed when the business process evolves. One challenge concerns the handling of long-running, active instances, which were initiated based on the old model, but now need to comply with the new specification. Potentially, thousands of active instances may be affected.

This paper focuses on structural adaptations of process models at different levels. Adaptations of single process instances (e.g., to add, delete or move activities) become necessary to deal with exceptional situations and often have to be accomplished in an ad-hoc manner [11]. Model changes at the process type level, in turn, have to be continuously conducted to evolve the PAIS [9]. It must be also possible to dynamically migrate running process instances to new model versions. Important challenges are to perform instance migrations on-the-fly, to guarantee compliance of migrated instances with the new model version, and to avoid performance penalties. Our ADEPT2 change framework addresses these challenges and explicitly covers the latter two kinds of flexibility; i.e., the adaptation and evolution of processes. However, through its ability to support late binding of sub-processes and to dynamically evolve or define these sub-processes, ADEPT2 is also able to support late modeling, and thus to deal with certain kinds of uncertainty.

The ultimate ambition of structural process adaptations during runtime is to ensure correctness of the modified instances afterwards. First, structural and behavioral soundness have to be guaranteed already at the model level (i.e., without considering instance states). Second, when performing instance adaptations this must not lead to flaws (e.g., deadlocks); i.e., none of the guarantees ensured by formal checks at build time must be violated due to the runtime adaptation. As example consider Fig. 1 where the model on the left-hand side is structurally modified by arranging parallel activities B and C in sequence afterwards. The instance running on the old model (with B being enabled and C being completed) does not comply with the new model version since its marking cannot be transferred to it (B must be completed before C may start). Such undesired runtime situations are denoted as *dynamic change bug* [16]. To exterminate them adequate correctness criteria are needed; e.g., to decide whether a given process instance is compliant with a modified process model and – if yes – how to adapt instance states when migrating the instance to the new model version.

In the following we deal with different correctness notions for dynamic process changes and discuss the strengths and weaknesses of the approaches relying on

![Fig. 1. Dynamic change bug](image-url)