Dealing with forward and backward jumps in workflow management systems

Manfred Reichert¹, Peter Dadam¹, Thomas Bauer²

1 University of Ulm, Dept. Databases and Information Systems, D-89069 Ulm, Germany; E-mail: {reichert,dadam}@informatik.uni-ulm.de
2 DaimlerChrysler Research and Technology Ulm, Dept. RIC/ED, Postfach 2360, D-89013 Ulm, Germany; E-mail: thomas.tb.bauer@daimlerchrysler.com

Received: 6 October 2002 / Accepted: 8 January 2003
Published online: 27 February 2003 – © Springer-Verlag 2003

Abstract. Workflow management systems (WfMS) offer a promising technology for the realization of process-centered application systems. A deficiency of existing WfMS is their inadequate support for dealing with exceptional deviations from the standard procedure. In the ADEPT project, therefore, we have developed advanced concepts for workflow modeling and execution, which aim at the increase of flexibility in WfMS. On the one hand we allow workflow designers to model exceptional execution paths already at buildtime provided that these deviations are known in advance. On the other hand authorized users may dynamically deviate from the pre-modeled workflow at runtime as well in order to deal with unforeseen events. In this paper, we focus on forward and backward jumps needed in this context. We describe sophisticated modeling concepts for capturing deviations in workflow models already at buildtime, and we show how forward and backward jumps (of different semantics) can be correctly applied in an ad-hoc manner during runtime as well. We work out basic requirements, facilities, and limitations arising in this context. Our experiences with applications from different domains have shown that the developed concepts will form a key part of process flexibility in process-centered information systems.

Keywords: Workflow management – Adaptive workflow – Exception handling – Forward/backward jump

1 Introduction

E-Business has significantly increased the competitive pressure companies must face [4]. To meet this challenge enterprises are developing a growing interest in supporting their business processes more effectively and in streamlining their application systems such that they behave “process-oriented”; i.e., to offer the right tasks at the right point in time to the right persons along with the information and application functions needed. Workflow management systems (WfMS) like MQSeries Workflow, Staffware, or INCOME Workflow offer a promising technology for this [33, 58]. Designed for a distributed environment they increase the number of work processes (workflow; abbr. WF) that can pass through an electronic workplace. For this purpose, the business process logic is extracted from application code. So, instead of a large, monolithic program package we obtain a set of WF activities which represent the application functions. The process logic between them (i.e., control and data flow) is specified in a separate WF schema. Usually, for WF modeling graphical formalisms like Petri Nets [1, 38, 58], Statecharts [32, 60], UML Activity Diagrams [16], or block-structured description languages [13, 36, 41] are used. They allow the WF designer to quickly define and modify WF schemes at a high semantic level, and enable the buildtime components of the WfMS to detect behavioral inconsistencies and errors in a very early implementation stage [46, 47, 52, 58].

Long regarded as technology for the automation of well-structured, repetitive processes, showing only little variations in their possible execution sequences, WF management is in the throes of transformation as more and more non-traditional applications require comprehensive process support as well. In many domains, like hospitals, engineering environments, or E-Commerce, however, process-oriented information systems will not be accepted if rigidity comes with them [4, 8, 14, 18, 26]. Instead users must be able to flexibly deviate from the standard process (e.g., by skipping WF activities or by working on a WF activity ahead of the normal schedule), in particular to handle exceptional situations [45,
Generally, we have to differ between deviations that can be pre-planned and deviations for which this is not possible. Concerning pre-planned deviations, their context as well as the actions necessary to handle them are known beforehand. They, therefore, can be already considered at buildtime in order to achieve a flexible WF execution behavior. As opposed to this, deviations that cannot be pre-planned may become necessary to deal with unforeseen events and must be dynamically handled during WF execution. In practice, both kinds of deviations frequently occur and must therefore be adequately supported by WfMS.

The present work is embedded in the ADEPT project which aims at the flexible support of enterprise-wide business processes [5, 14, 41]. We have developed and implemented advanced concepts for the modeling, execution, and monitoring of workflows as well as for the dynamic change of in-progress WF instances. Our work is based on first-hand knowledge with clinical as well as engineering workflows [8, 14]. We have observed that many exceptions are known in advance and can therefore be considered already at buildtime, which decreases the necessity of “expensive” ad-hoc interventions during runtime. To enable users to cope with unforeseen exceptions as well, additionally, we offer advanced concepts for dynamic WF changes. They are based on the ADEPTFlex calculus which enables authorized users to dynamically change the structure, the state, and the attributes of in-progress WF instances in a consistent manner and at a high semantic level [41].

In this paper we develop advanced concepts for both, the increase of flexibility at buildtime and its enhancement during runtime. Thereby, we focus on the support of forward and backward jumps, which are indispensable to flexibly deal with exceptions in WfMS [14]. While the former enable deviations in forward direction (e.g., to skip unnecessary activities or to work on a particular activity ahead of the normal schedule), backward jumps make it possible to partially roll back the flow to a previous execution state and to re-commence work in this state (e.g., when activity execution fails). We present concepts for both, the pre-modeling of jumps at buildtime and their dynamic application during runtime. To better understand related issues and problems, we consider the viewpoint of the WF designer as well as of the end user. In some respects forward and backward jumps bear resemblance to GOTO statements in programming languages.

However, deviations from standard procedures concern exception handling at a higher semantic level, which is indispensable for WfMS to cover a broad spectrum of processes. (Note that the need for supporting jump operations has been approved by several other research groups as well [1, 27, 36, 42, 50].) Nevertheless, jumps must not be complicated for users or lead to an undefined execution behavior. For this reason, ADEPT imposes several restrictions for their use, which either have to be checked at buildtime (pre-planned jumps) or must be ensured when applying the jump during runtime (ad-hoc jumps). Backward jumps, for example, must always result in a former state of the WF instance in order to guarantee a consistent execution behavior. Forward jumps, in turn, must not lead to activity program invocations with missing input data or to skipping of imperative activities. Finally, jump operations must be properly integrated with respect to authorization and documentation.

Although very important for realizing and adaptive workflows, forward and backward jumps do not cover all exception handling procedures needed in practice. As we have reported in earlier papers [14, 41], ADEPT provides other facilities as well. Examples include the ad-hoc insertion or deletion of WF activities, the late modeling of sub-workflows, and the dynamic change of WF attributes (e.g., activity work assignments). In addition, several research groups have used the ADEPT WfMS for implementing sophisticated exception handling procedures on top of it, like the automatic adaptation of workflows or the dynamic creation of WF instances as response to occurring exceptions [4, 53]. In this context, ECA rules (Event – Condition – Action) can be used to describe the conditions leading to an exception and the actions necessary to handle them [11, 36]. Other exception handling approaches are discussed in Sect. 5.

The outline of this paper is as follows: Sect. 2 furnishes basic information about WF modeling and execution in ADEPT – background information which is necessary for a further understanding of this paper. Section 3 describes how pre-planned as well as ad-hoc forward jumps can be flexibly realized in WfMS. In Sect. 4 we set out how backward jumps have to be handled. We discuss related work in Sect. 5 and conclude with a summary in Sect. 6.

2 Background information

For each business process type to be supported, a corresponding WF schema has to be defined and stored in the WfMS. An example is depicted in Fig. 1. Among other things, the diagrammed WF schema defines WF activities as well as the control and data flow between them. The work presented in this paper uses the ADEPT formalism [39, 41] for WF modeling and execution. On the one hand this WF meta model is expressive enough to adequately model real-world processes [14], on the other hand, the resulting WF models are easy to understand.