Information Systems 40 (2014) 142–152

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Case-based adaptation of workflows

Mirjam Minor [a,*], Ralph Bergmann [b], Sebastian Görg [b]

[a] Goethe University Frankfurt, Department of Business Information Systems, D-60325 Frankfurt, Germany
[b] University of Trier, Department of Business Information Systems, D-54286 Trier, Germany

ARTICLE INFO

Available online 23 December 2012

Keywords:
Workflow management
Adaptation
Case-based reasoning

ABSTRACT

This paper presents on a Case-based Reasoning approach for automated workflow adaptation by reuse of experience. Agile workflow technology allows structural adaptations of workflow instances at build time or at run time. The approach supports the expert in performing such adaptations by an automated method. The method employs *workflow adaptation cases* that record adaptation episodes from the past. The recorded changes can be automatically transferred to a new workflow that is in a similar situation of change. First, the notion of workflow adaptation cases is introduced. The sample *workflow modeling language CFCN* is presented, which has been developed by the University of Trier as a part of the agile workflow management system Cake. Then, the retrieval of adaptation cases is briefly discussed. The *case-based adaptation method* is explained including the so-called *anchor mapping algorithm* which identifies the parts of the target workflow where to apply the changes. A *formative evaluation* in two application domains compares different variants of the anchor mapping algorithm by means of experts assessing the results of the automated adaptation.

1. Introduction

Workflows are a well-established concept to formalize business processes and support their execution by a workflow management system. *Workflows* are "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [41]. While early workflow management systems put their focus mainly on production workflows, additional application fields like financial services, telecommunications, the public sector and even creative areas like media industry or nutrition services are addressed by workflow technology today [32]. With the changing requirements in this wide range of areas, we identified a research gap for more flexibility support of workflows at run time as described in the following.

In traditional workflow management systems, *workflow definitions* (also called workflow templates or process models) are created at build time. The basic constituents of workflow definitions are *tasks* that describe an activity to be performed by an automated service (e.g. within a service-oriented architecture) or a human (e.g. an employee). The procedural rules to control the tasks are usually described by routing constructs like sequences, loops, parallel and alternative branches that form the *control flow* of the tasks. Documents and information are represented by *data objects*; the relationships between data objects and tasks form the *data flow*. *Workflow instances* (in short *workflows*) are derived from the workflow definitions to be enacted, i.e., to be scheduled for execution within a *workflow engine* of the workflow management system.

A significant limitation of traditional workflow technology is the missing flexibility of the workflow concept and the lack of flexibility of workflow engines. Once

* Corresponding author. Tel.: +49 69 798 24636.
E-mail addresses: minor@informatik.uni-frankfurt.de (M. Minor), bergmann@uni-trier.de (R. Bergmann), goergs@uni-trier.de (S. Görg).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.11.011

described, workflow definitions are instantiated repeatedly and the instances are executed in the same manner over a long period of time. However, changes of ongoing work are quite common in areas like media industry or telecommunication. In order to address today's requirements concerning easy and fast adaptation of workflows, a new class of workflow systems has been emerging since the late Nineties: *Adaptive workflow systems* (also called agile workflow systems) [30,37,36,25,31] facilitate structural changes of workflows at run time. Workflow instances can be created (for example based on a workflow definition from a repository [9]) and tailored for a particular demand or business case. Workflow instances can still be adapted after they have been started, for example if some unforeseen events occur [39]. Currently, adaptive workflow technology has reached such a degree of maturity that the adaptive workflow systems are about to be transferred into practice.[1] Hence, the creation and adaptation of workflows has now become an important activity to enable the flexibility of business processes. Though, in real-world scenarios both are nontrivial modeling tasks for which intelligent support is needed.

We put forward the hypothesis that supporting the users "by sample", i.e., by retrieving and automatically transferring past modifications, alleviates the creation and adaptation of workflow instances during their entire life-cycle. *Case-based Reasoning* (CBR) [1] is a problem-solving technology that facilitates the reuse of experience in form of cases. A *case* records a problem situation together with the experiences gained during a problem-solving episode, which includes a solution. CBR provides techniques for representing, storing, indexing, retrieving, and adapting cases. Instead of composing new solutions from scratch, retrieved cases or portions of cases can be reused. We assume that workflow modeling experts who are faced with the problem of how to adapt a workflow instance benefit from solutions and experiences from similar problem situations in the past. The paper addresses a case-based approach supporting the adaptation of workflow instances by automated workflow retrieval and adaptation at build time and run time. This is a contribution towards more flexibility support in workflow management.

The paper is organized as follows. Section 2 gives a brief introduction into agile workflow technology. In Section 3, we present a representation form for adaptation knowledge called adaptation cases. Section 4 is on an automated method for workflow adaptation. In Section 5, we provide evaluation results. Section 6 contains a discussion of related work. Finally, we draw a conclusion in Section 7.

2. Agile workflow technology

Agile workflow technology is the technical prerequisite for adaptation support at run time. We will briefly introduce its core concepts before we deal with the actual adaptation methods. Agile workflow technology addresses structural changes of workflow instances at run time. The changes apply to *workflow elements*, i.e., to atomic parts of the workflow. The types of workflow elements depend on the workflow modeling language. In case of Petri Nets [33], for instance, the types would be places, transitions, and connections while in BPMN [13] they might be activities, gateways, events, connecting objects, and artifacts. The workflow adaptation is carried out by deleting, modifying, or adding one or several workflow elements, or by changing the order of elements.

Two types of structural changes at run time are considered in the approach (for a more detailed discussion of change patterns see [39]): *Ad hoc changes* occur unexpectedly at any time. The workflow elements that will be affected by the change are not known in advance. For instance, a device might be disabled accidentally during a software update process. The workflow supporting the software update process might be adapted by inserting a task for the re-installation of the driver. In contrast to ad hoc changes, *late modeling* refers to structural changes that are projectable to some extent. It is known when the time for the modification will come, but it is not fully foreseeable which workflow elements will be affected. For example, a customer might require to test several variants of a product component before making the decision which variant should actually go into the production process. As a consequence of the decision, the product development process might require changes like further tests or modified integration activities.

We use the Cake Flow Cloud Notation (CFCN, compare [28]) as a sample modeling language to illustrate the work. It has been developed in recent research project at the University of Trier [28,11]. It is a part of the Collaborative Agent-based Knowledge Engine (Cake) [4,26]. Cake provides modeling and enactment support for agile workflows. CFCN consists of several types of workflow elements. A CFCN workflow element can be a task, a data object, a data link (from a data object to a task or vice versa), a start element, an end element, a breakpoint, or a control flow element like an AND-split, AND-join, XOR-split, XOR-join, LOOP-split, or LOOP-join. The control flow elements border block-oriented control flow structures (AND-block, XOR-block, etc.). Blocks cannot be interleaved but they can be nested. For example, the software update workflow depicted in Fig. 1 (I) has the following workflow elements: A start element, an AND-split and AND-join bordering two tasks "Evaluate updates" and "Plan update release", a further task "Deploy updates", and an end element. In part (II) of Fig. 1, an ad hoc change is started by setting a breakpoint. The breakpoint suspends the subsequent workflow elements from execution, so that only the lower branch of the AND-block with the task "Plan update release" can continue execution, while the upper branch is under construction. In part (III), an additional task "Re-install driver" has been inserted. Finally, in part (IV), the breakpoint has been deleted again and the upper branch of the AND-block can continue execution. Each workflow element has a unique identifier (ID) and – except for data objects and data links – an execution state. For instance, the execution state "ACTIVE" is assigned to elements that have already started

[1] See http://www.aristaflow.com for a commercial sample of an adaptive workflow system and http://www.uni-trier.de/index.php?id=21075 for a research prototype.