Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Data & Knowledge Engineering 51 (2004) 223–256

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

AgentWork: a workflow system supporting rule-based workflow adaptation

Robert Müller, Ulrike Greiner *, Erhard Rahm

Department of Computer Science, University of Leipzig, Augustusplatz 10/11, Leipzig 04109, Germany

Received 22 March 2003; received in revised form 12 August 2003; accepted 31 March 2004
Available online 25 May 2004

Abstract

Current workflow management systems still lack support for dynamic and automatic workflow adaptations. However, this functionality is a major requirement for next–generation workflow systems to provide sufficient flexibility to cope with unexpected failure events. We present the concepts and implementation of AgentWork, a workflow management system supporting automated workflow adaptations in a comprehensive way. A rule-based approach is followed to specify exceptions and necessary workflow adaptations. AgentWork uses temporal estimates to determine which remaining parts of running workflows are affected by an exception and is able to predictively perform suitable adaptations. This helps to ensure that necessary adaptations are performed in time with minimal user interaction which is especially valuable in complex applications such as for medical treatments.

Keywords: Workflow management; Adaptive systems; Active rules; Temporal logics; Agents

1. Introduction

Workflow management is widely adopted as a core technology to support long-term application processes in heterogeneous and distributed environments [2,17,21]. Main characteristics include the clear separation of application program code from the overall process logic and the integration of automated and manual activities. Workflow technology is increasingly used to

* Corresponding author. Tel.: +49-341-97-32241; fax: +49-341-97-32209.
E-mail address: greiner@informatik.uni-leipzig.de (U. Greiner).

0169-023X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2004.03.010

manage complex processes in Internet-based e-commerce, virtual enterprises, or medical institutions [33,49]. For example, due to precisely specified treatment procedures in many medical disciplines, workflow management systems can be used to implement diagnostic and therapeutic processes [13,40,49]. Major goals include the improved and timely treatment of patients and a significant workload reduction for the hospital personnel.

However, conventional workflow management systems do not provide sufficient flexibility to cope with the broad range of failures that may occur during workflow execution. In particular, not only *system* failures such as hardware or software crashes need to be dealt with but also *logical* failures or exceptions. These logical failures refer to application-specific exceptional events for which the control and data flow of a workflow is not adequate anymore and thus has to be adapted [54]. The automatic treatment of such logical failures is the main subject of this paper.

In the cancer chemotherapy workflow shown in Fig. 1, assume it is detected just before the administration of drug *C* that the leukocyte count (i.e., the number of white blood cells) has become critically low, so that there is the risk of a serious infection for the patient. As drug *C* is known to reduce the leukocyte count additionally as a negative side effect, the activity "Administer drug *C*" dynamically has to be removed from the workflow while the execution of the other activities can be continued without change. To protect the patient from an infection, it may also be necessary to dynamically *add* an activity supporting the administration of an antibiotic drug after the cancer chemotherapy. Note that explicit conditional routing paths in the workflow definition are not sufficient to deal with such exceptions. For example, checking the condition "leukocyte count < 1000" before the "Administer drug *C*" activity would not help if this condition is violated at different points in time possibly requiring different actions (e.g., dropping drug *A* instead of drug *C*). Inserting conditional branches at any potentially relevant position would significantly reduce workflow readability and maintainability. Thus a more flexible exception handling is required to decide on how to best react to logical failures.

Previous work on dynamic workflow adaptation mostly focused on a *manual* approach where the administrator or an authorized user has to decide which events constitute logical failures and which adaptations have to be performed [45]. However, the manual approach can be time-consuming and error-prone thereby threatening the goals to be achieved with workflow management. For example, during a therapy such as the one shown in Fig. 1, a physician is usually faced with up to 20 patients and 10–30 findings per patient every day. With a manual failure handling, the physician always would have to keep in mind which findings may induce which adaptations, or at

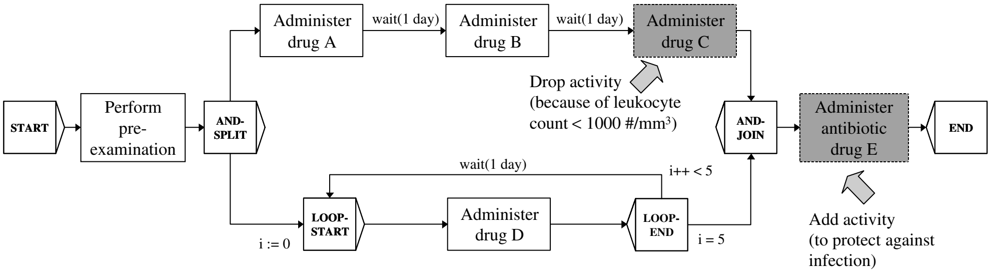

Fig. 1. Workflow adaptation example.