Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 37 (2004) 49–69
Decision Support Systems
www.elsevier.com/locate/dsw

Consulting past exceptions to facilitate workflow exception handling

San-Yih Hwang[a,*], Jian Tang[b]

[a] Department of Information Management, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan, ROC
[b] Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1B 3X5

Accepted 30 September 2002

Abstract

In this paper, we propose an architecture model that deals with both expected and unexpected exceptions in the context of workflow management. Expected exceptions and their handling approaches are specified by ECA rules, while cases of unexpected exceptions are characterized by their features and resolution approaches. The handling of unexpected exceptions is then assisted by the system providing information about how recent similar cases were resolved. The ways in which the previous exception cases were handled provides useful information in determining how to handle the current one. Quantifying the similarity of exception cases is described, and three algorithms for efficiently searching for similar exception cases are proposed and evaluated both theoretically and by experimenting with synthetic data sets.

Keywords: Workflow exceptions; Workflow management; Exception handling; Similarity matching

1. Introduction

Workflow management systems (WFMSs) support the execution of business processes. A business process has a market-centred aim of fulfilling a business contract or satisfying a customer's needs [13], and typically, it is controlled by many factors, including the description of the constituent activities, their control/data flow, the potential participants, the organization model and the referenced data [31]. A WFMS separates the specification of business processes, or so-called workflow types, from their execution and provides a convenient and powerful means of specifying a business process and controlling its executions. However, it is well recognized that defining a workflow that represents all properties of the underlying business process is difficult [10]. Moreover, since the formulation of business processes occurs at a high conceptual level, workflows have to adapt rapidly to a changing environment, resulting in executions that deviate from the predefined plan.

The process model defined at specification simply represents a standard case, and WFMSs should provide the flexibility to support run-time modification to the defined workflows so as to handle non-standard cases, or so-called exceptions. Some types of exceptions are expected because they are known to occur occasionally or periodically, and their char-

* Corresponding author.
E-mail address: syhwang@mis.nsysu.edu.tw (S.-Y. Hwang).

0167-9236/$ - see front matter © 2002 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-9236(02)00194-X

acter and the associated way of handling them can be completely decided at build-time. Other exceptions are *unexpected* since they result from unpredictable changes in the environment, being unable to decide how to handle the exception, or from some other factor that simply cannot be predicted at design-time. It has been observed that exceptions occur rather frequently in real working environments [11,27]. This highlights the importance of exception handling in the context of workflow management.

1.1. Related work

The need for handling workflow exceptions has been identified by several researchers and research projects in recent years (e.g. EXOTICA [1], METEOR [27], ADOME [7,8], ADEPT [23,24], WAMO [11], and WIDE [6]). Most research has focused on the handling of expected exceptions whose character can be anticipated at build-time. To keep the logic of the normal process clean, exceptions and their handling approaches are usually not incorporated into standard workflow types. Instead, another more flexible mechanism is adopted to support explicit modelling of these exceptions. Two approaches that are typically used to implement this mechanism are the *extended transaction model* and *event–condition–action* (*ECA*) *rules*. An extended transaction model requires the workflow designer to specify some properties of the constituent activities and sub-workflows, such as compensatible, retriable, and alternating activities. When an exception occurs, the workflow system handles it according to the given attribute values of the involved activities or sub-workflows. A typical situation would be to roll-back activities to a particular point by executing the corresponding compensating activities in reverse order. An alternative path would then be taken when continuing the execution [1,12,21]. Another approach is to use ECA rules, or so-called triggers [6–8]. The ECA paradigm describes an exception type as a particular ECA rule. The event and the condition of an ECA rule describe the situation under which the associated exceptions occur, and the action part defines the operations that would resolve these exceptions. Possible operations include notifying responsible persons, ignoring exceptions, retrying the activity that causes exceptions, partial rollback followed by forward execution, adding some extra activities, deleting some planned activities, or any change to the part of the workflow definition that is not yet executed. Recently, Hagen and Alonso [14] proposed a framework that integrates rules[1] and an extended transaction model for handling workflow exceptions. Participants in the ADEPT project also proposed an approach which involved evaluating the correctness of changes to workflow schema [23,24]. While previous work has focused on providing flexible mechanisms for defining exceptions at build-time, there is a need to handle exceptions that are not defined at build-time. These exceptions are by no means uncommon and, in some cases, could be substantial.

Cases not specified by the defined workflow types require special treatment. Even though a WFMS may be capable of executing any exception resolution plan that is specified in a particular format, deriving an appropriate solution for handling a given exception is currently conducted in a manual, ad hoc manner, which involves numerous meetings and discussions with authorized and knowledgeable persons. We proposed [29] providing a query interface to enable users to browse the information related to the workflow instance to which an exception occurs. In the proposal, a query is specified in terms of attributes of constituent activities which include, for example, input and output values, the date and time, and details of the performers. By examining the attribute values returned by a number of queries, the user makes an appropriate decision on how to handle the current exception. Chiu et al. [8] proposed the *Human Interface Manager* for handling unexpected exceptions. This device handles exceptions by listing common approaches that serve as suggested resolutions, and allows users to visualize all the recent methods that have been used to resolve exceptions. However, exactly how to provide a list of suitable resolutions for a given exception in a changing environment was not discussed in detail.

[1] In their work, the exception model in C++ or Java, rather than ECA rules, is used to combine with an extended transaction model. However, we consider this exception model to be the same as ECA rules in spirit.