Workflow Recovery Framework for Exception Handling: Involving the User

Hernâni Mourão[1] and Pedro Antunes[2]

[1] College of Business and Administration, Setúbal Polytechnic Institute
Campus do IPS – Estefanilha, 2914–503 Setúbal, Portugal,
and LASIGE (Laboratory of Large Scale Information Systems)
hmourao@esce.ips.pt
[2] Faculty of Science, University of Lisbon, Departamento de Informática
Campo Grande – Edifício C5, 1749–016 Lisboa, Portugal,
and LASIGE (Laboratory of Large Scale Information Systems)
paa@di.fc.ul.pt

Abstract. Unexpected exceptions in WfMS are situations not predicted during the design phase. Human involvement in handling this type of exceptions has been recognized to be a crucial factor. We developed a framework to support the user in handling these situations by redesigning the flow, ad hoc executing the affected tasks, and manipulating engine status. A good characterization of the exception is needed to help the user identifying the best executable solution. The proposed characterization results from integrating operational, tactical and strategic perspectives over unexpected exceptions. An open source platform was selected to establish a test base on which the framework will be tested.

1 Introduction

Workflow exceptions are situations not predicted in the workflow model or when there exists a deviation from the model and the real world [16]. Workflow exceptions have been accounted for almost half of the total working time in an office [21].

Up to now, workflow exceptions have mostly been addressed in a systems perspective, even though a consensus seems to arise on the necessity of involving the user in the recovery of a specific kind of exceptions. Therefore we developed a framework aimed to fulfill this identified gap. The framework is built around the idea that human intervention must be supported with quality information about the problematic situation, status of the workflow engine and possible recovery solutions.

Considering the work in progress, this paper currently offers two contributions to exception handling in WfMS: (1) an integrated perspective over exceptions and exception handling, considering three different organizational levels (strategic, tactic and operational) affected by exceptions and respective human roles in recovery (design changes, ad hoc execution, and engine status manipulation); and (2) a set of WfMS components necessary to help and support these roles: event handler, situation aware

J. Favela and D. Decouchant (Eds.): CRIWG 2003, LNCS 2806, pp. 159–167, 2003.

ness, problem characterization and recovery toolkit. These components are currently being validated on an open source platform.

The paper is organized as follows. Section 2 reviews the literature and establishes the grounds for the proposed framework. Section 3 presents the integrated perspective, where different exception handling approaches are mapped to the organizational levels where they occur and are handled. In Sect. 4 we describe the components that constitute our solution. In Sect. 5 we describe the project current status, identify the main current contributions, and discuss the expected results.

2 Literature Review

In this section we first identify the characterization of exceptions found in the literature and then the handling strategies adopted.

2.1 Characterization of Exceptions

[7] characterize failures and exceptions in a single dimension, encompassing two types of failures and two types of exceptions:

- Basic failures – Associated with failures on the systems supporting the WfMS (e.g., operating system, database management system and network failures);
- Application failures – Failures on the applications invoked to execute tasks (e.g., unexpected data input);
- Expected exceptions – Events that can be predicted during the modelling phase but do not correspond to the "normal" behaviour of the process (e.g., a customer reporting a car accident in a car rental company);
- Unexpected exceptions – When the semantics of the process is not accurately modelled by the system (e.g., changes in rules, or a change in the order processing of an important client).

[8] suggests an escalating concept to transform the failures that cannot be resolved in the level where they occur into exceptions. This way, the classification can be reduced to exceptions.

[4] combines the above view with another orthogonal characteristic described as "exception source." The exception source can either be internal or external to the workflow. A similar classification is adopted by [22] but without any distinction between expected and unexpected exceptions.

Next, we will analyse the expected and unexpected exceptions in more detail, primarily basing our comments on results produced by the WfMS research community. Later, to complete this review, we will present a broader perspective, oriented towards the organizational semantics.

Expected exceptions are cases that can be predicted during the modelling stage but do not correspond to the "normal" process behaviour. Some mechanisms should be

implemented to handle these situations because they can occur frequently [7] and can cause a considerable amount of work to process. In the car accident example, the company has to re-schedule all future rentals for that specific car until the car is re-paired. The "normal" behaviour of the process should have been the returning of the car to the company, as planned, while the accident corresponds to a deviation or an "occasional" behaviour: an expected exception.

[2] identifies four classes of expected exceptions, according to the events that generate them: workflow, data, temporal, and external. Workflow exceptions occur on starting or finishing a task. When workflow relevant data changes a data exception can be fired. Temporal exceptions are related to time stamps, e.g., car not returned on time. Finally, external exceptions are activated by external signals, e.g., the car accident example.

Unexpected exceptions result from inconsistencies between process modelling in the workflow and the actual execution [2]; and result from incomplete or design errors, improvements or changes in the business manoeuvre or quality and customer satisfaction issues not known during the modelling stage. This type of exceptions is frequent in highly complex or dynamic environments and cannot be predicted during the modelling phase. Usually, unexpected exceptions force the process to change to a halt state and require human intervention [13].

In situations where this kind of exceptions occurs frequently, one should consider redesigning the workflow model, if it is out of date, or adopting different technologies based on collaborative work or adaptive workflow systems [3].

[21] proposes a taxonomy based on the organizational semantics associated to exceptions. This work defines a set of base concepts necessary to construct a consistent conceptual framework and fundament the characterization of organizational exceptions. Namely, the concepts of rule, case, event and their relationships are the basis for the framework.

The proposed taxonomy was developed from empirical studies. A special attention was made on the social and financial impacts of exceptions. Six different criteria are used to classify exceptions:

- Exceptionality – Difference between the exceptional and "normal" event;
- Handling delay – Time elapsed between the exception identification and handling;
- Amount of work – Extra work required to handle the exception when compared to the normal event;
- Organizational influence – Number of people involved in the exception;
- Cause – A measure of the importance of the reason for the exception;
- Rule impact – the changes in the organization's rules due to the exception.

Three classes of exceptions were identified according to exceptionality: established exceptions, otherwise exceptions, and true exceptions. Established exceptions occur when rules exist in the organization to handle an event but the right ones cannot be found. Otherwise exceptions occur when the organization has rules to handle the normal event but do not apply completely to the case. Finally, true exceptions occur when

the organization has no rules. According to the organizational influence criteria, exceptions can also be classified at employee, group and organizational level.

2.2 Exception Handling

WfMS systems should deal with failures and exceptions at execution time because predicting any possible cause of failure or exception during design is very difficult or even impossible and makes the system very complex and hard to manage [2; 5; 9; 14].

In a primitive approach, we could rely on the system that supports the WfMS. In fact, most of the commercially available DBMS on the market implement the necessary transaction processing mechanisms to react in case of failure, returning the system to a coherent state and enabling forward execution [2]. On the other hand, some tasks do not run in transactional environments and a typical task can span over a long period of time. This is the complex environment of organizations where typical DBMS solutions are not adequate [22].

Some suggestions to overcome these problems consider relaxing the ACID properties and incorporating compensation mechanisms for backward recovery and forward execution, incorporating the flexibility required for workflow systems [16]. Several researchers are working in these issues [6; 9; 11; 15]. A good survey of this area, which became known as Extended Transaction Models (ETM), is presented by [22].

The error handling semantics of traditional transaction processing systems is also too rigid for workflow [16; 22]. The handling of application failures based on transactional approaches offer, in general, extreme and expensive solutions in terms of lost work and should be avoided. Therefore, some application failures should then be treated as expected exceptions [2].

Some other studies have proposed the adoption of dynamic and adaptive workflow systems to react to exceptions during workflow execution. The operator, on the presence of an exception, could change the system by either creating a new path for the exceptional process or change all the processes running on the system to the newly created path. Some studies have been carried out to keep the system's consistency and correctness during/after the change, e.g., [1; 10; 18; 20].

[5; 6] recognized the fixed control sequence and the rigid compensation policy of the ETM approach and developed the Event Condition Action (ECA) rules to decouple the detection and handling of exceptions from the system itself and to increase flexibility. [3] describes a system to deal with expected exceptions based on ECA rules. The language Chrimera-Exc is used to specify exceptions and augment the WfMS characteristics to automatically detect and handle expected exceptions.

[17] describes a system using a Case Base Reasoning (CBR) scheme to derive patterns for exception handling. The current exception is matched to a knowledge base and the system determines the appropriate action to handle the specific case.

Despite all these efforts to automatically handle exceptions, the majority of authors involved recognize the limits of the proposed solutions. They either recognize the importance of interrupting the process and integrating some manual mechanism

[9; 14], or explicitly state that in some situations the role of humans is crucial to collect process specific data not available to the workflow system [2; 13; 17].

To complete this review, it is important to mention two other approaches introducing a broader perspective over workflow exceptions. [12] proposed an integrated architecture of formal coordinated processes with informal cooperative processes. [21] presents an approach focussed on organizational semantics. Petri nets, outside the scope of the WfMS, define the reactions to the various types of exceptions, and should be interpreted as a global organizational reaction to exceptions.

3 Integrated Perspective

From the above discussion we can assume that accounting for all possible exceptions requires an integrated approach, where different levels of the organizational system affected by exceptions must be involved.

Depending on the cause and impact of the abnormal situation, a suitable recovery mechanism at the most adequate level should be invoked. The operational level provides an environment for handling basic and application failures only, where the traditional transaction processing techniques can be sufficient to bring the system back to a coherent state and continue execution without human intervention. Whenever these techniques are not able to solve the problem, the event is propagated to the tactical level, and the failure may be converted into an external exception.

At the tactical level, the WfMS may automatically handle expected exceptions in many different ways. For instance, using ETM techniques [9]. The extensive work done on adaptive workflow, which falls in this level, should increase the system flexibility and augment its adaptation to real world situations in handling expected exceptions. At this level, human contribution is possible but limited to producing the information necessary to have the system applying the exception handling techniques (e.g. compensation, retry, ignore, etc).

Eventually, if none of the techniques implemented at this level are able to handle the event, it should be propagated to the strategic level, where the attention of a human operator to an unexpected exception is raised.

Even though several authors present some ideas on how to incorporate human involvement in exception handling, we believe that the problem has not yet been completely addressed. [9] describes the idea of a system that incorporates human intervention in two possible modes: (1) ad hoc extension, where the user can suspend the execution of a task and choose alternative paths or change the execution model; and (2) ad hoc refinement, where the user interrupts the task execution to execute one or more activities and later on proceeds with the interrupted task. Even though the authors integrate this model in their framework, we believe that it should be completed in terms of the tools/methodologies available to help and support the user manipulating the environment in which s/he operates.

In [17], a user-interface is provided to assist human intervention. The exceptions tagged as requiring human intervention come to the attention of an expert that can choose one candidate solution within the system proposals, or write a new solution. In

order to react to unexpected exceptions, the system is notified by an external signal and generates an internal exception event. However, it is not clear how this mechanism can be implemented for any particular type of unexpected exception. Moreover, it is not foreseen any assistance mechanism to provide a general view of the situation nor any support tool to change the model.

Note that although human participation is considered at both the tactical and strategic levels, the strategic level envisages a more dramatic intervention in the WfMS. This will be discussed in more detail in the next section.

4 Our Approach to a Solution

In our proposed system, the interaction between the WfMS and the user is supported by the following components:

Event Handler. This component is responsible for launching the recovery process whenever an unexpected exception is detected. It interacts with the user in order to display and manage any upstream and downstream exception propagations related with the exception (e.g., a basic failure previously propagated as an external exception, or an employee exception subsequently propagated as a group exception). This component also interacts with the WfMS to identify the target user. In order to facilitate this identification, by default, unexpected exceptions are classified as "established" and "employee."

Situation Awareness. This component is responsible for gathering and displaying to the user generic data about the workflow and engine status associated with the exception. In particular, it lists all tasks defined by the WfMS, their status and other important details, like task goals. This component also allows the user to identify which tasks are affected by the exception.

Problem Characterization. This component employs the criteria defined by [21] to obtain from the user the information necessary to classify (or reclassify) an exception as a true, established or otherwise exception. It also allows the user characterizing the organizational influence of the exception (employee, group or organizational level) and identifying the relevant participants. A list of participants is gathered from the WfMS.

Recovery Toolkit. Based on the situation awareness and problem characterization, this component offers a collection of pre-defined actions, which can be combined by the user to manipulate the process design, process execution and engine status. The following collection of pre-defined actions is actually considered:

- Engine status – Start/terminate or suspend/continue several tasks; manipulate process relevant data or workflow participants;
- Process execution – Support ad hoc extensions and ad hoc refinements to the process instances affected by the exception; apply design changes according to the specification;

- Design – Change one or several process definitions, either affected by this exception or not; define how and when the modifications are applied (one/all instances, immediately or after completion).

Note that the user can execute, in any order, multiple actions in each one of the above three categories. The heuristic used for proposing the best solutions according to the characterization of exceptions is the following: otherwise exceptions are commonly handled by ad hoc extensions or refinements applied to the affected process instances; established exceptions can be handled by instantiating already available process definitions; and true exceptions require design changes and possibly cascading the exception to other users. If, in the first two cases, the organizational influence of the exception is the organization, the approach must be coordinated with other areas.

5 Project Status and Expected Results

We selected OpenSymphony [19] to implement our solution. OpenSymphony is an open-source platform based on J2EE technology. The prototype consists of Enterprise Java Beans (EJB) for functional components and web-tier components for front end – all components are platform, database and application server independent.

The workflow specification uses a XML file rather than a graphical user interface, which gives more flexibility to cope with changes in the process definitions. The workflow engine is based on the concept of a finite state machine where each state is represented by a combination of StepID and status. The engine supports Java-based functions, BeanShell scripts, and Bean Scripting Framework scripts. Another type of functions is triggered by outside sources and run under the system context.

Figure 3 shows the user interface implementing our exception handling approach. The user interface highlights the four main components necessary to our approach.

The user can analyse the list of exception propagations in the Event Handler area, if any. The interface allows analysing upstream and downstream exceptions, as well as propagating the current exception.

In the Situation Awareness area, the user sees a list of all the tasks running on the system (left side). Selecting the affected ones and pressing the right arrow button, they are transferred to the right side. This way it is possible to identify all the currently running tasks that are affected by the exception. Whenever necessary, the specific task details, including its goals, can be viewed/edited.

The Problem Characterization area enables the selection of the exception characteristics according to our approach. At this stage only the exceptionality and organizational influence are considered. The other four characteristics (handling delay, amount of work, cause, and rule impact) can only be defined at the end of the exception handling, for historical records. In the organizational influence zone there is an area that changes according the selection, allowing to select an employee, group leader or organizational manager, respectively. This way someone is always associated with an exception. There is also the possibility of sending emails to everyone involved.

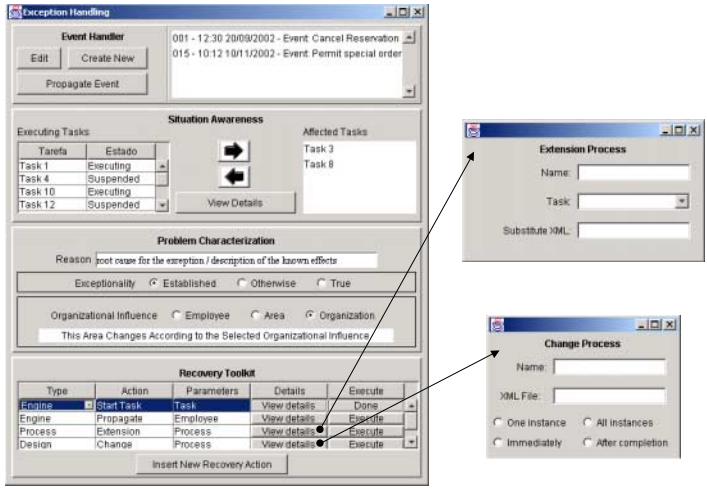

Fig. 1. Interface for exception handling

In the Recovery Toolkit area the user can decide the recovery action(s) to implement on this specific case. As noted before, any combination of the defined above actions can be used for the particular case, although the system suggests some predefined actions. On the type, action, and parameters columns the user selects one option from a list.

A working prototype is currently being developed to implement the described framework. We expect to test this prototype with a collection of simulated situations to test the applicability, flexibility and robustness of the framework. A real world situation should complete the study, raising issues not possible to predict in a simulation (particularly concerning the usability of the prototype).

With this work we also expect to improve the OpenSymphony platform, allowing the workflow components to cope with unexpected exceptions.

References

1. Aalst, W.v.d., 1999. Generic workflow models: how to handle dynamic change and capture management information. Int. Conf. on Cooperative Information Systems, pp. 115–126.
2. Casati, F., 1998. Models, Semantics, and Formal Methods for the Design of Workflows and their Exceptions. PhD Thesis, Politecnico di Milano.
3. Casati, F., Ceri, S., Paraboschi, S., and Pozzi, G., 1999. Specification and Implementation of Exceptions in Workflow Management Systems. ACM Transactions on Database Systems, 24(3): 405–451. ACM Press.
4. Chiu, D.K., 2000. Exception Handling in an Object-oriented Workflow Management System. PhD Thesis, Hong Kong University of Science and Technology.
5. Dayal, U., Hsu, M., and Ladin, R., 1990. Organizing Long-Running Activities with Triggers and Transactions. Int. Conf. on Management of Data SIGMOD'90, NJ, USA.

6. Dayal, U., Hsu, M., and Ladin, R., 1991. A Transactional Model for Long-Running Activities. 17th Int. Conf. on Very Large Data Bases (VLDB'91). Barcelona, Spain.
7. Eder, J., and Liebhart, W., 1995. The Workflow Activity Model WAMO. Int. Conf. on Cooperative Information Systems, Vienna, Austria.
8. Eder, J., and Liebhart, W., 1996. Workflow Recovery. 1st IFCIS Intl. Conf. on Cooperative Information Systems (CoopIS'96). IEEE, Brussels, Belgium, pp. 124–134.
9. Eder, J., and Liebhart, W., 1998. Contributions to Exception Handler in Workflow Management. EDBT'98, Workshop on Workflow Management Systems. Valencia, Spain.
10. Ellis, C., Keddara, K., and Rozenberg, G., 1995. Dynamic change within workflow systems. Proc. of Conf. Organizational Computing Systems, Milpitas, CA, USA, pp. 10–21.
11. Georgakopoulos, D., Hornick, M., and Manola, F., 1996. Customizing Transaction Models and Mechanisms in a Programmable Environment Supporting Reliable Workflow Automation. IEEE Transactions on Knowledge and Data Engineering, 8(4): 630–649
12. Guimarães, N., Antunes, P., and Pereira, A.P., 1997. The Integration of Workflow Systems and Collaboration Tools. In: A.K. Dogaç, Leonid Ozsu (Editor), Advances in Workflow Management Systems and Interoperability. Instambul.
13. Heinl, P., 1998. Exceptions During Workflow Execution. EDBT'98, Workshop on Workflow Management Systems. Valencia, Spain.
14. Klein, M., and Dellarocas, C., 2000. A Knowledge-Based Approach to Handling Exceptions in Workflow Systems, CSCW, 9(3): 399–412. Kluwer Academic Publishers.
15. Krishnakumar, N., and Sheth, A.P., 1995. Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations. Distributed and Parallel Database Systems, 3(2).
16. Luo, Z., 2001. Knowledge sharing, Coordinated Exception Handling, and Intelligent Problem Solving for Cross-Organizational Business Processes. PhD Thesis, Dep. of Computer Sciences, University of Georgia.
17. Luo, Z., Sheth, A.P., Kochut, K., and Arpinar, I., 2002. Exception Handling for Conflict Resolution in Cross-Organizational Workflows, LSDIS Lab, Computer Science, Un. of Georgia.
18. Myers, K.L., and Berry, P.M., 1999. At the Boundary of Workflow and AI. Proc. of the AAAI-99 Workshop on Agent-Based Systems in The Business Context Held.
19. The OpenSymphony project. http://www.opensymphony.com, 2001, 20–04–2003.
20. Reichert, M., and Dadam, P., 1998. ADEPTflex – Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems, 10(2): 93–129.
21. Saastamoinen, H., 1995. On the Handling of Exceptions in Information Systems. University of Jyväskylä, PhD Thesis, University of Jyväskylä.
22. Worah, D., and Sheth, A.P., 1997. Transactions in Transactional Workflows. In: S.K. Jajodia, Larry (Ed.), Advanced Transaction Models and Architectures. Kluwer Academic Publishers.