Chapter 1 **Using WfMS to support unstructured activities**

1.1 Introduction

Workflow Management Systems (WfMS) are based on the premise that procedures are able to define the details of the work carried out in organizations. Since procedures and control data emerge from the overall system, the WfMS is much more flexible than traditional information systems, and any change to the procedure or control data may be easily accomplished. Using a WfMS, the organization is released from the task of routing the process and all related information through the different tasks and affected actors.

This original development of WfMS was biased by the rationalistic view that organizations follow procedures on a rigid way to achieve their goals [Suchman, 1983]. However, organizations also require flexibility when performing their daily operations and procedures do not necessarily contain all the required information to accomplish the work. This clash between the original objectives of WfMS and the concrete organizational requirements lead to a difficult acceptance of these systems by their target market during the nineties [van der Aalst and Berens, 2001].

It has been shown by various ethnographic studies that the idealistic smooth procedural work is not always followed [Suchman, 1983; Bowers et al., 1995]. Often, procedures are only used as guidance, since users adapt to the peculiarities of the situations not completely reflected in procedures. We thus have two different scenarios usually referred as *unstructured,* when users perform unrestricted activities eventually guided by an available procedure, and *structured* when procedures determine the user actions.

These two scenarios should be taken into account when supporting organizational activities. However, WfMS are traditionally algorithm-based and developed with a special focus on the structured scenario. One of the main disadvantages of this approach is the lack of flexibility to adjust to concrete user demand [Abbott and Sarin, 1994]. An *exception* is therefore a situation where the WfMS is not able to support the users performing organizational activities.

Various researchers have addressed this lack of flexibility. However, the majority of the proposed solutions are still biased by the rationalistic approach, where more primitives are inserted on the WfMS to handle exceptions but always under any sort of an algorithm-based control. Even when primitives are inserted to increase adaptability, they have their roots on the original workflow model and therefore do not support totally unstructured activities.

In this chapter we describe a solution developed to address the problem that traditional WfMSs have coping with unstructured activities. We assume there will always be situations where users should be able to decide on what are the most suited activities to fulfil organizational goals, with or without restrictions imposed by the system.

1.2 Adjusting the WfMS to Organizations

The work processes carried out by organizations have been identified to belong to a continuum ranging from totally unstructured to completely structured [Sheth et al., 1996]. It is interesting to note that the majority of the available organizational information systems tend to fall close to both sides of the spectrum boundaries [Sheth et al., 1996], thus leaving a significant gap in between. Unfortunately, traditional WfMS fall into the highly structured boundary and thus contribute to this gap. WfMS emphasize the execution of work models and thus have a normative engagement [Schmidt, 1997]. Closer to the other end of the spectrum limits, Suchman [1987] proposes the notion of maps, which position and guide actors in a space of available actions, providing environmental information necessary to decision making but avoiding the normative trait. Email systems, the newly developed collaborative Web platforms sharing information among users and group support systems are examples of systems that fall close to the unstructured limits of the spectrum. Usually these systems promote interaction and do not have a normative engagement.

Since traditional WfMS fall close to the structured limits of the spectrum, they are inadequate to cope with unstructured processes. To support the continuum of organizational needs, WfMS should cope with the whole spectrum of structured and unstructured activities. This requirement has been identified by different authors [Ellis and Nutt, 1993; Abbott and Sarin, 1994]. In our solution, we propose a system that is able to switch its behaviour from model guidance to map guidance, back and forth. We start this section by discussing the limitations of two definitions of WfMS and propose an extension to the WFMC reference model.

Sheth et al. [1996] enhance the various facets of a WfMS in their definition. The definition starts with the business process: a collection of activities tied together by a set of precedence relations and having a common organizational objective. This involves distributing, scheduling, controlling and coordinating work activities among humans and computers. This definition also embraces the organizational perspective of business processes as a collection of tasks pursuing a common goal, and the system perspective, where the tasks must be coordinated, distributed, scheduled and controlled.

Therefore, Sheth et al. [1996] define workflow management as the automated coordination, control and communication of work task, both of people and computers, as it is required to carry out business processes. This is performed by a workflow enactment service, which is controlled by a computerised representation of the organizational processes and provides the required services on a computing network.

The WFMC's definition [WfMC, 1999] states that a WfMS consists of software components to store and interpret process definitions, create and manage workflow instances as they are executed, and control their interaction with workflow participants and applications.

Both definitions emphasize the computer control of the workflow execution governed by a representation of the organizational processes. This characteristic is inline with our previous discussion of the structured characteristics of the WfMS. However, we established as one of the objectives of WfMS to be able to support unstructured activities, where users should be able to decide on the most suited activity without being restricted by any system contingency. This requirement implies that the system should be able to transfer execution control to the user.

The WFMC reference model [Hollingswoorth, 1995] is also biased by the rationalistic approach and does not foresee any mechanism of transferring control to the users. To account for unstructured activities, we extended the reference model defined by the WFMC with the model represented in Figure 1.1. The main idea behind this extension is not to specify the interface details for WfMS interoperability, as in the WFMC reference model, but to identify the new architecture required to support unstructured activities in a traditional WfMS. Therefore, the interfaces are identified here, but the functionality will be described throughout the chapter. The major goal behind this extension is that the system must be able to switch from model guidance to map guidance. This functionality requires direct interaction with the enactment services of WfMS, represented by interface A. Another required functionality is the capacity to implement model changes on the running instances. These changes require access to the process definition tools (interface B) and to the enactment services (interface A) in order to identify the instances on which the change is to be applied.

In some situations it may also be necessary to suspend the execution of a process model, to reallocate a task or to monitor system evolution using the standard WfMS functions. These features are implemented using interface C with the Administering and Monitoring Tools.

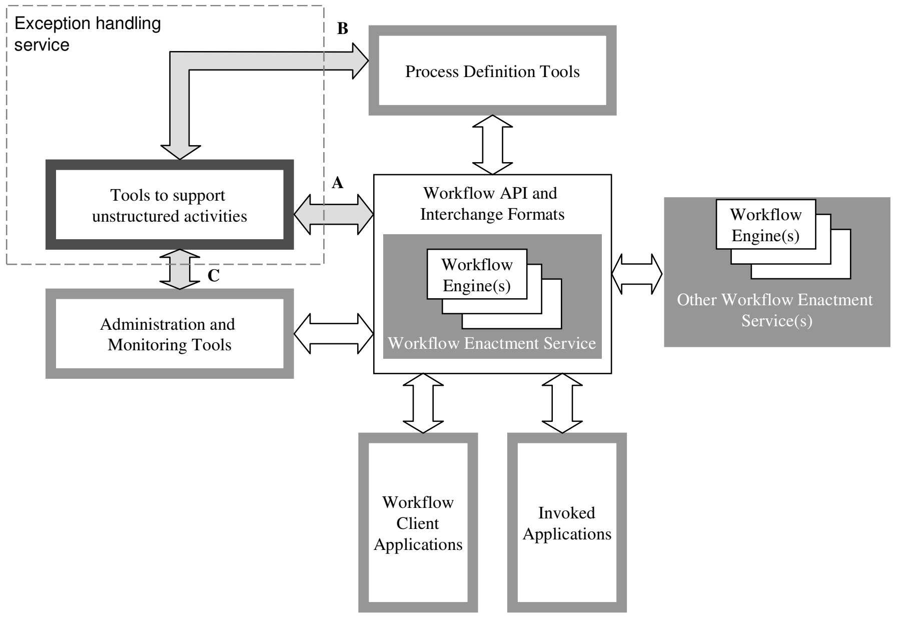

Exception handling service
B
Process Definition Tools
A
Tools to support unstructured activities
Workflow API and Interchange Formats
Workflow Engine(s)
Workflow Engine(s)
C
Other Workflow Enactment Service(s)
Administration and Monitoring Tools
Workflow Enactment Service
Workflow Client Applications
Invoked Applications

Figure 1.1. Extended WfMC's reference model

1.3 Exceptions in WfMS

There are several ways to classify exceptions in WfMS according to the different perspectives that are applied to the problematic situation. In the related literature, some orthogonal criteria for exceptions classification can be found [Saastamoinen, 1995; Eder and Liebhart, 1995; Casati, 1998; Mourão and Antunes, 2003]. Section 0 considers a system perspective and classifies exceptions according to the impact of the event on the system behaviour. In Section 1.3.2 we describe a taxonomy that adopts an organizational perspective. Finally, in Section 1.3.3 we conclude with a proposal for a new taxonomy that classifies exceptions in a set of dimensions helping users deciding on the most adequate strategy to handle the event.

1.3.1. Systems perspective on failures and exceptions

Eder and Liebhart [1995] characterize failures and exceptions according to a single dimension, encompassing two types of failures and two types of exceptions:

- *Basic failures* – associated with failures in the systems underlying the WfMS (e.g., operating system, database management system and network);

- *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;

- *Unexpected exceptions* – when the semantics of the process is not accurately modelled by the system (e.g., changes in business rules or a change in the order processing of an important client.)

The Eder and Liebhart’s [1995] classification distinguishes two major types of deviations from the standard execution of a WfMS: failures and exceptions. The former result from system malfunctions either within the WfMS and the systems that support it or within the applications that implement the various tasks, where the later result from semantic discrepancies between the model and the application environment. The authors recognize that the currently available techniques to solve system and application failures do not overcome every situation and therefore suggest an escalating concept to transform into exceptions the failures that cannot be resolved in the level where they occur.

As defined above, expected exceptions may be predicted during the modelling stage but do not correspond to the “normal” process behaviour. These situations are usually excluded from the work model in order to reduce complexity. However, some authors posit that mechanisms should be implemented to handle these situations because they may occur frequently [Eder and Liebhart, 1995; Casati, 1998; Chiu et al., 2001; Sadiq, 2000; Luo, 2001] and cause a considerable amount of work to handle. For example, consider the example of a client reporting an accident to a car rental company; the company has to reschedule future rentals for that specific car until the car is repaired. The “normal” behaviour should have been the car returning to the company, as planned, while the accident corresponds to a deviation or an “occasional” behaviour: an expected exception.

Chiu et al. [2001] combine the above view with another orthogonal characteristic described as *exception source*. The exception source can be internal, when the exception is triggered by the system, or external when a user reports the exception.

Another taxonomy, classifying the type of event that originated the expected exception, was developed by Casati [1998][1]. We have enriched it distinguishing these classes:

- *Workflow* – triggered when a task or process is started or ended, it refers to the execution of the workflow itself. E.g., a deadlock situation or a loop being executed more times than expected;

- *Data* – identified within the task that generates an error condition. The data events, even though identified within a particular instance can affect a collection of instances (e.g., a trip being booked twice for the same client). These exceptions refer only to workflow relevant data used for workflow evolution. If the error refers to application data operations, they will result into an application failure that is not considered by this class;

- *Temporal* – triggered on the occurrence of a given time stamp (E.g., a rented car not delivered on time);

- *Non-compliance events* – triggered whenever the system cannot handle the intended process due to differences between the tasks and goals.

- *System/application events* – triggered when the system is not able to recover from lower level failures, such as database, network or application failures (lower level failures are propagated as semantic failures [Eder and Liebhart, 1995]).

Finally, the definitions found in the literature for unexpected exceptions state that they result from inconsistencies between process modelling in the workflow and the actual execution [Casati, 1998]; they are mentioned to be consequences of incomplete or design errors, improvements or changes in the business manoeuvre, and issues unknown during the modelling stage [Heinl, 1998]. From our point of view, any situation that is not predicted in the model and requires out of the box activities (unstructured activities) is an unexpected exception.

1.3.2. Organizational perspective on exceptions

Saastamoinen [1995] proposed a taxonomy based on the organizational semantics associated to exceptions. The taxonomy defines a set of base concepts necessary to construct a consistent conceptual framework that fundaments the characterization of organizational exceptions. Then, the taxonomy was developed using these concepts as well as empirical studies carried out in an organization, with a special attention to the social and financial impacts of exceptions.

[1] This classification was developed for expected exceptions, because it assumes that the detection of an unexpected exception is always external to the system. However, any of the above classes may result from an unpredicted situation even though the symptoms are expected.

Six different criteria were proposed to classify exceptions. However, four of these dimensions can only be established after the handling procedure is finished and can not be used to guide the operators overcoming the situation. The two relevant dimensions are:

- Exceptionality – difference between the exceptional and "normal" event;
- Organizational influence – number of people involved in the exception;

Three classes of exceptions were identified according to exceptionality: established exceptions, otherwise exceptions and true exceptions. Established exceptions occur when the handling procedure for the event is defined but the rules in the organization do not support users identifying the correct one. 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. Employee exceptions are situations that affect only the work of one person. Group exceptions affect a group of people working within the same process, in the same kind of job or in the same process. Organization exceptions affect the work of persons in more than one department or project in the organization.

1.3.3. New exception classification

In this section we discuss an extended exception classification focused on the knowledge about the situation processed by the organization and on the planning capacity. We start by the former dimension and finish with the later.

Figure 1.2. shows the expected-unexpected continuum in the Eder and Liebhart's [1995] taxonomy (cf. Section 0) bellow the line. Our taxonomy is shown above the line. In our taxonomy, we propose three exception types. The definition of these types is based on the similarity of the situation with the complete set of rules and past experience that exists in the organization. *True expected* exceptions are at the expected limits of the spectrum and are those for which the handling procedures are entirely defined. For *extended expected* exceptions, which initiate close to the expected limits and extend into the spectrum, some guiding behaviour may be drawn from rules and past experience even though some adjustments are required. Finally, *effective unexpected* exceptions are those for which the organization can not derive any guiding behaviour from the organizational knowledge base. Since the system may not obtain any handling procedure, the user involvement is mandatory. Even further, some exceptional situations can represent a strategic opportunity that will not be recognized by the system. (E.g., a sales representative may identify new opportunities in some minor changes to an existing process.)

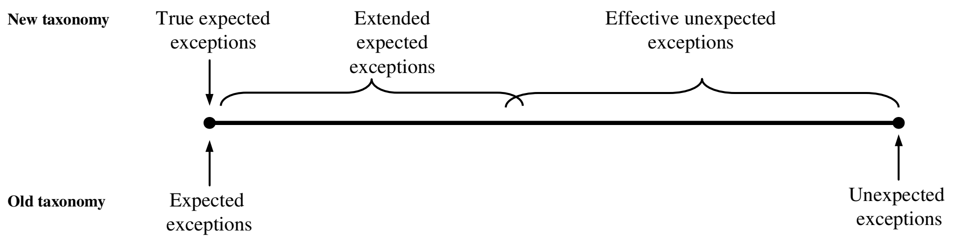


Figure 1.2. Three exception types in the expected-unexpected continuum.

A new dimension will now be used to further classify effective unexpected exceptions: the planning capacity for the handling procedure. In this dimension two classes are identified:

- *planned effective unexpected exceptions*;
- *ad hoc effective unexpected exceptions.*

For *planned effective unexpected exceptions*, a reaction plan can be established before the reaction starts. It usually means the organization has enough knowledge about the situation to establish a reaction plan. (E.g., a new legislation that the company has to comply within a period of time.) For *ad hoc effective unexpected exceptions* no plan can a priori be established. The reason may be that there is not enough knowledge about the event that enables advanced planning of reaction procedures, or the environmental conditions vary so much that no plan can robustly be defined. In these cases the situated characteristics of actions should prevail over prescribed ones and the reaction must be implemented in an ad hoc way (unstructured activities) involving problem solving among participants both for situation diagnosis and recovery. For example, if a truck with a very important delivery is stuck on a traffic jam, the users can not define a priori what is the best action to overcome the situation. It may be the case that traffic just starts to flow and no reaction is necessary, while in some situations another delivery by a different road may be the best solution. Users should collect as much information as they can and react as the situation evolves.

Therefore, ad hoc effective unexpected exceptions require human intervention and an innovative posture from the organization to deal with the situation. As no plan is available, human reaction should be map guided, according to Suchman's definition [1987]. This exception type is the main focus of the present chapter. From now on, they will be referred as *ad hoc effective unexpected exceptions* or simply *unexpected exceptions* when no distinction is necessary.

1.4 Openness and completeness

This section describes the two requirements we consider mandatory to support unstructured activities: openness and completeness.

> *The openness requirement states that the system should be able to collect environmental and workflow status information to support users on their map guided activities. Users should then be able to look for the most relevant information to understand the situation and decide on the most adequate activities to carry out.*

On the other hand, users should not be restricted to the services provided by the exception handling system. The challenge is to manage awareness and consistency with the exception handling activities carried outside the WfMS scope. Our solution integrates environmental information about external activities but will not assume control of those activities. The main idea is that unstructured activities characterize human reactions to ad hoc effective unexpected exceptions.

Since some operations are carried outside system boundaries, it should be possible to maintain information on such activities and register any information useful to the involved actors. This requirement is also important to maintain an update history log of the implemented activities carried out during the exception handling procedure.

> *The completeness requirement states that an exception handling system should consent users to carry out recovery actions without restrictions, i.e., the flexibility of the exception handling system should be on par with the flexibility actors have on their daily activities when working without system control.*

This definition is based on the notion that people tend to solve their problems with all the available means. If any system restrictions are imposed to the users' primary goals, they will overcome the system [Strong and Miller, 1995].

It is important to note the flexibility implied by the completeness requirement should be supported by the WfMS enactment service. When the system is running, the enactment service is responsible for instantiating the work models and guarantee the processes run according to what is defined. Therefore, any deviations from the standard procedures must be implemented at this component.

The consequences of this open perspective on WfMS are profound. For instance, the restrictions to the common model changes found in the literature for adaptive and dynamic systems (see Section 1.5.2) must be relaxed. These restrictions are only applicable if one wants to keep the execution under the specified work models. However, if the objective is, for instance, to graciously abort a workflow instance, no consistency check is necessary. Even further, if the user decides to implement a recovery action that deliberately inserts structural conflicts in the work model, s/he should be advised on potential problems and allowed to proceed.

1.5 Resilience in WfMS

Since WfMS support business processes, it is very important they keep operational during business operations even under unpredictable situations. Their ability to adjust to actual businesses solicitations and to react to different hazardous conditions such as failures and exceptions is a core property for a WfMS to actually support organizations:

> *The resilient property of a WfMS concerns its ability to maintain a coherent state and continue supporting business processes after being subject to any hazardous situations that affect its execution.*

It should be emphasized this is a runtime property of the WfMS, because predicting any possible cause of failure or exception during design is considered very difficult or even impossible and makes the system very complex and hard to manage [Casati, 1998]. The strategy to manage failures and exceptions is to increase system resilience. Resilience requires both robustness, to avoid system crashes due to failures, and flexibility to adjust to deviations on the user and organizational conditions.

This section focuses on techniques to increase robustness and augment flexibility: resilience. The next section is dedicated to analyse the systemic approaches to resilience. The systemic techniques assume the objective to provide the WfMS with the necessary mechanisms to react to basic and application failures, and expected exceptions. The systems handling expected exceptions fall in this group because they do not increase flexibility when users face a new exception at runtime.

The second section proceeds with the human oriented approaches to increase resilience. The various research lines were grouped in two classes according to their approach to the problem (classification inspired by [Han et al., 1998]): metamodels and open-point. Some other approaches to support unstructured activities are also mentioned. The following section compares the different approaches to augment resilience and draws conclusions to use in the remainder of the chapter.

1.5.1. Systemic approaches to increase resilience

We have previously distinguished (cf. Section 0) between system and application failures, where system failures result from malfunctions either within the WfMS and the systems that support it and application failures result from errors in the applications that implement the workflow tasks. Systemic approaches aim to handle this type of events and are defined as:

> *Systemic approaches are designed to handle failures and exceptions without human intervention.*

However, it has been recognised that in some situations it is not possible to handle the event without human intervention [Eder and Liebhart, 1995; Casati, 1998; Chiu et al., 2001]. A propagation mechanism must be foreseen to transform these situations into unexpected exceptions so they can be handled with human support. Systemic approaches are therefore limited by the limited capacity of WfMS to overcome problems without human intervention.

Usually, WfMS use a database management system (DBMS) to persist the workflow relevant data. Transaction processing techniques, developed in the DMBS field, guarantee data integrity and consistency on system failures. 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 [Casati, 1998]. Therefore, on the event of a system failure, the DBMS implements a standard failure handling task by restoring a previous coherent state. The WfMS is then able to proceed with forward execution.

A wide variety of work has been developed to increase the system capacity to recover in the case of an application failure. Advanced transition models with relaxed Atomicity, Consistency, Isolation and Durability (ACID) properties were one of the most adopted research lines. This line tries to handle the long-running characteristics of WfMS activities by relaxing the ACID properties of the transactions and defining compensation activities to recover the system to a coherent state. Nevertheless, as mentioned before, researchers in this field recognize that it will always be necessary to escalate the failure to an exception when the system is not able to handle it.

The systems designed to handle true expected exceptions are supported on special constructs triggered by conditions that identify the presence of an expected exception. These constructs initiate the execution of the procedure designed to handle the expected event. The systems developed by Casati [1998] and by Chiu et al. [2001] are examples developed within this group.

Researchers have tried to increase the applicability of these approaches by applying artificial intelligence techniques and exception mining. The idea behind these approaches is to augment the matching mechanism between the detected event and the existing exception description to verify if the defined procedure can be used in the event handling.

1.5.2. Human-oriented approaches to increase resilience

As mentioned at the beginning of this section, resilience is a system's runtime property. Therefore human-oriented approaches to increase flexibility may be defined as:

> *Human-oriented approaches are designed to support human interventions in business processes at runtime, and increase the systems ' resilience by increasing its flexibility.*

Flexibility is related to the operations not predicted in the model that users carry out to accomplish work [Agostini and De Michelis, 2000; van der Aalst and Basten, 2002; Ellis and Nutt, 1993; Casati, 1998; Mourão and Antunes, 2004]. I.e., when a process is instantiated, the user may implement some operations not predicted in the model but made available by the workflow enactment service. When the intervention requires model adaptation, the process definition tools may be used to design the new model, while the enactment service replaces the old model by the new and continues operation.

Two main research streams can be identified in this area [Han et al., 1998]: metamodel and open-point. Metamodel approaches take into major consideration the structural and dynamic constraints to model adaptations, while open-point approaches define special points in the workflow model where the adaptations can be made. Metamodel approaches offer higher intervention latitude since they are not limited by special points in the model where the intervention can be made. However, they require model consistency checks, while in the open-point approaches the consistency checks are not necessary due to the restrictions in the allowed interventions.

Metamodel approaches are usually referred in the literature as producing dynamic and adaptive WfMS and are actually one of the most important research streams to increase flexibility in WfMS. On the occurrence of exceptions, users should be able to change workflow models at runtime, adapting them to the new situation and migrating running instances to the new model without stopping or breaking the system [Ellis et al., 1995; Reichert and Dadam, 1998; Agostini and De Michelis, 2000; Casati, 1998; Sadiq, 2000; van der Aalst and Basten, 2002; Weske, 2001]. Two types of interventions are identified in the related literature [van der Aalst and Basten, 2002; Rinderle et al., 2004]: ad hoc changes and evolutionary changes.

These interventions may be defined as:

> *Ad hoc changes are typically applied to a small set of instances and are a reaction to a particular situation that affects some specific processes.*

And,

> *Evolutionary changes result in a new version of the workflow model and result from changes in the business processes that the organization is required to implement.*

Both ad hoc and evolutionary changes must be executed under the system control to keep correctness, avoiding the insertion of deadlocks, unreachable states or inconsistencies in the data dependency model. These solutions define *a set of change rules enabling automated correctness checks*. Two correctness criteria must be taken into consideration: structural and state related. The former concerns schema changes and assures the new model is consistent. The state related criterion concerns the state of the instances to be migrated and verifies if they can be propagated to the new model.

It should be emphasized these approaches require that a new model is issued and instances migrated to this new model. Even when ad hoc interventions are allowed, and users may implement special actions in the model they are always restricted by a model, either the original or the changed one. Therefore, users should plan the intervention before any recovering mechanism is started. However, in some situations, especially in ad hoc unplanned effective unexpected exceptions, users are not able to issue any plan and they should start the recovery as soon as they identify the situation.

Some other approaches deserve being mentioned here because they handle the same problem we identified. The system developed by Agostini and De Michelis [2000] is one of the few approaches having the objective of supporting users in a way similar to our approach, allowing collaboration between the involved actors. However, the model limits interventions and users should not insert any inconsistencies.

Guimarães et al. [1997] proposed an integrated architecture of formal coordinated processes with informal cooperative processes. [Saastamoinen, 1995] presents an approach focussed on organizational semantics. Another important research in this area was proposed by Bernstein [2000], who developed a system to support processes in the whole spectrum from structured and unstructured.

1.5.3. Discussion

From the above discussion it is important to realise that systemic approaches are crucial to increase the WfMS robustness regarding failures and expected exceptions. However, when effective unexpected exceptions are raised (cf. Section 1.3.3), human intervention is required.

On the systems aiming to augment flexibility, the metamodel approaches rely on modelling formalisms to support operators implementing changes to the model and migrating the running instances to the new model, while open-point approaches define specific points in the model where interventions are allowed. Open-point approaches are easier to implement because they do not require consistency checks but they have lower latitude for user interventions. Metamodel approaches allow a higher degree of interventions but the users should plan new models before starting the interventions. Even further, the interventions are limited to maintain model consistency.

Figure 1.3 positions the systemic approaches, metamodel and open-point approaches according to the type of control and planning capacity. The arrow bellow indicates the direction for increasing flexibility and shows how the approaches are positioned within the same control type. Three classes were identified in this dimension: systemic, restricted humanistic and unrestricted humanistic. Systems designed to handle failures and expected exceptions have systemic control and planed reaction. In fact, the reaction to these events is pre-planned. Expected exception handling offers higher flexibility because it is easier to plug-in and change the pre-planned reaction to events. Humanistic approaches augment the operators' latitude of intervention and may be applied at runtime, increasing the flexibility to react to unforeseen events. Humanistic approaches were split in the figure into restricted humanistic and unrestricted humanistic. Open point and metamodel approaches are able to support users on both ad hoc and planned interventions. However, they are not able to support unstructured activities because of their limited latitude for interventions (restricted by model consistency).

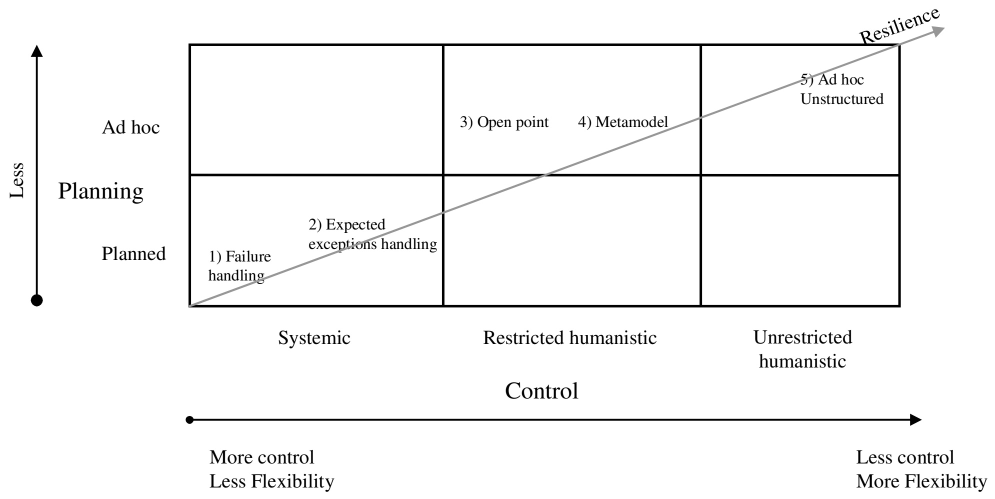


Figure 1.3. Classification of approaches according to control and planning capacity

The characteristics of our system place it on the top right of the figure, identified as *ad hoc unstructured*. Here, flexibility is at its maximum degree and interventions are fully ad hoc since the planning capacity is very low. They are also unrestricted by model consistency.

From the figure above, we may realize that systemic approaches rely on the stage 1) and 2) on the resilience axes and they only provide systemic support. Metamodel and open-point are at stage 3) and 4) since they do not provide unrestricted support to unstructured activities.

1.6 A Solution to Support the Whole Spectrum of Organizational Activities

This section introduces our proposed solution to support the whole spectrum of organizational activities. The system should be able to work under model guidance and adopt map guidance when an unexpected exception is detected. Unstructured activities are carried out until the system is back into a coherent state. Then, the user will either place affected instances under model guidance or abort them. The overall system behaviour is modelled by the state diagram in Figure 1.4.

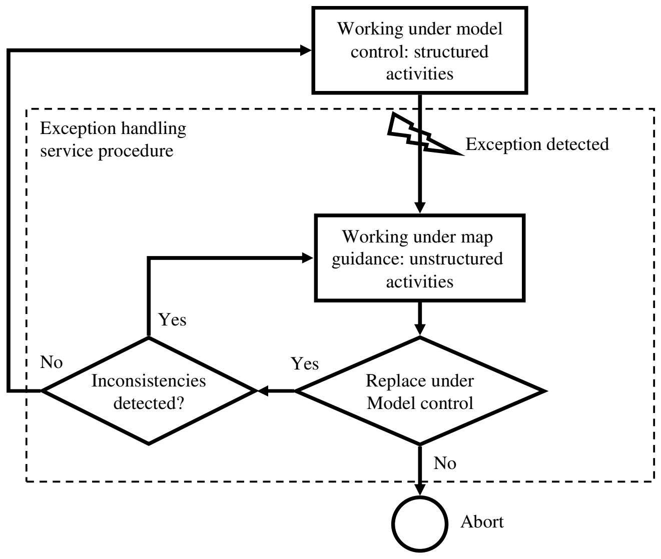

Working under model control: structured activities
Exception handling service procedure
Exception detected
Working under map guidance: unstructured activities
Yes
No
Inconsistencies detected?
Yes
Replace under Model control
No
Abort

Figure 1.4. Solution’s state diagram

In our solution, we implement the extended WfMC reference model (cf. Section 1.2). The developed functionality supporting unstructured activities runs on a workflow engine. A dedicated model implements the components to support unstructured activities.

After this brief introduction, in Section 1.6.1 we present our conceptual approach. The basic exception handling functions are identified in Section 1.6.2: detection, diagnosis, recovery and monitoring. Section 1.6.3 discusses the exception diagnosis and in Section 1.6.4 we discuss the recovery and monitoring functions.

1.6.1. Conceptual approach

Figure 1.5 is our proposal for the extended WfMC reference model[2]. When the system is supporting structured activities, the traditional WfMS has control over activities and the exception handling service is inactive. When an exception is detected, the exception handling service interrupts the WfMS execution and the system starts supporting unstructured activities.

[2] Figure 1.1 is repeated to simplify the reading.

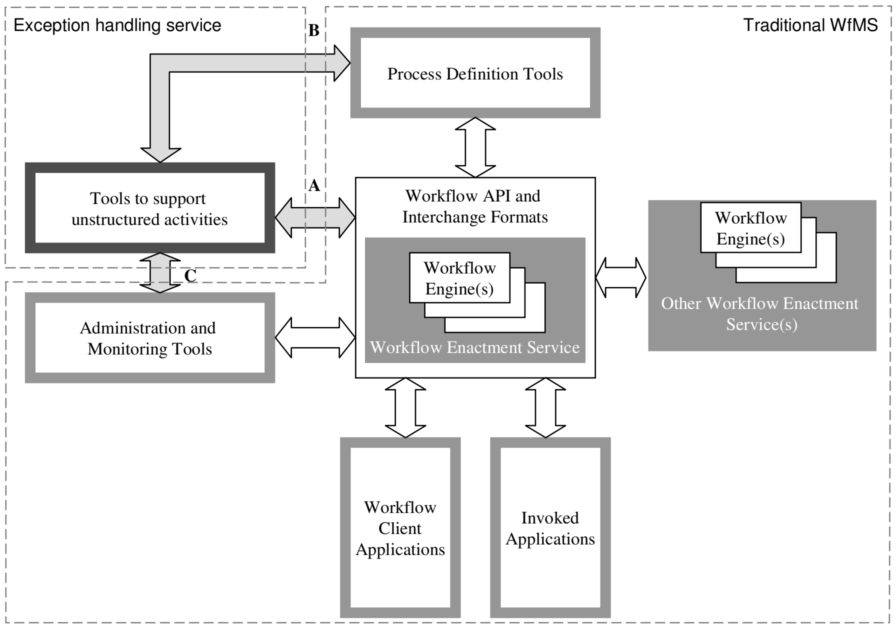

Exception handling service
Traditional WfMS
B
Process Definition Tools
Tools to support unstructured activities
A
Workflow API and Interchange Formats
Workflow Engine(s)
Workflow Enactment Service
Workflow Engine(s)
Other Workflow Enactment Service(s)
C
Administration and Monitoring Tools
Workflow Client Applications
Invoked Applications

Figure 1.5. Extended WFMC's reference model

While supporting unstructured activities, the system offers the following functionality:

(i) Escalation;

(ii) Monitoring;

(iii) Diagnosis;

(iv) Communication;

(v) Collaboration;

(vi) Recovery;

(vii) Coordination;

(viii) Tools to determine the best solution;

(ix) History log.

The relevant organizational actors may have to be involved in the exception handling activities. The organizational levels with adequate decision authority should participate in decision making and action implementation. The *escalation* mechanism allows the involvement of organizational members in this process. On the other hand, to support the group of involved users overcoming the exceptional situation the system must also:

1. Support users understanding the situation – diagnosis;

2. Support users deciding the most adequate actions to overcome the situation – recovery;

To facilitate *diagnosis*, users should be fed with quality information about the peculiarities of the situation at hand. Since information and knowledge about the event are spread throughout the organization and the environment, our solution implements *monitoring* mechanisms to collect relevant data and enable knowledge sharing among the participants (cf. openness requirement in Section 1.4). This shared effort is supported by *collaboration* and *communication* mechanisms facilitating common situation awareness. *Diagnosis* is also supported by a situation description component used to classify the event according to several dimensions. Since the situation may evolve over time, users may change the description.

The recovery process may be characterized as a mutual adjustment and coordination effort. The *collaboration* and *communication* mechanisms support mutual adjustment. (Mutual adjustment requires combined communication, coordination and collaboration). It should be emphasized that during unstructured activities support, the coordination facet implemented by the WfMS is relaxed since users gain control over orchestrating their activities. This unavoidable characteristic of the solution imposes a special focus: users should coordinate their activities under map guidance.

The functionality *tools to determine the best solution* accounts for application environments where special tools may be used to support both the understanding of the situation and the decision process. (E.g., operations research algorithms in a lot manufacturing company may support users calculating the lots that should be manufactured when reacting to an unexpected change in demand.)

Finally, our solution maintains a *history log* for the situation description and all of the implemented activities. When new values are defined for the situation description, the old values are stored in the historical log.

1.6.2. Basic functions

As mentioned before, unexpected exception[3] handling is a problem solving activity that requires understanding the situation and implementing the required activities to overcome the exceptional situation. We distinguish four functions in the process:

- exception detection;
- situation diagnosis;
- exception recovery;
- monitoring actions.

[3] Remember that we use the term "unexpected exceptions" to refer to ad hoc effective unexpected exceptions whenever it is not necessary to distinguish them.

Since detection is only important for triggering the handling procedure and is independent from the other functions, we will not describe it here. This section only describes the other three functions.

The majority of authors identify the first three functions [Sadiq, 2000; Dellarocas and Klein, 1998]. However, as it was discussed before (cf. the openness requirement in Section 1.4) and will be further developed bellow, we posit that monitoring actions play a key role in unexpected exception handling.

In our solution, we advocate an intertwined play between diagnosis, recovery and monitoring until the exception is resolved. That is to say, the diagnosis is not considered complete on the first approach but rather through an iterative process where different actors may collaboratively contribute and information collected from monitoring and recovery actions is used to improve it. It should also be stressed that both the exceptional situation and perception of the situation may change along this iterative process, as new information is made available and processed by humans. Therefore, monitoring actions support this cognitive process.

After diagnosis, users may carry out recovery actions. The open nature of the proposed solution suggests that the recovery actions do not always run in the inner system context and thus some linking mechanism is necessary to bring environmental information to the system.

1.6.3. Exception diagnosis

A good understanding of the exceptional situation is crucial for users to take the right decisions on which recovery actions to adopt. As already mentioned, providing rich context information is critical for convenient map guidance. This information should also support the diagnosis and decision on the best handling strategies. The diagnosis is mostly dependent on a detailed and accurate assessment of the exceptional event.

Using the classifications described in Section 1.3 and some new added characteristics, we propose the following dimensions:

1. *Scope – process specific* when only a set of instances is affected; or *cross specific* when various sets of instances are affected;

2. *Detection – automatic* if the exception is automatically detected by the system; or *manually* if the exception is manually triggered;

3. *Event type* – refers to the event that generates the exception (cf. Section 1.6.2): *data*, *temporal*, *workflow*, *external events*, *non-compliance* or *system/application*;

4. *Organizational impact – employee*, *group* or *organizational,* according to Section 1.3.2;

5. *Difference to the organizational rules – established exceptions*, *otherwise exceptions* or *true exceptions (*cf. Section 1.3.2);

6. *Complexity of the solution* – refers to the complexity associated to obtain an optimal solution: *easy* or *hard*. If the complexity is hard users should consider using a tool to determine the best solution (cf. Section 1.6.1);

7. *Reaction time* – defines the time frame for user's reaction to the event: *quick*, *relaxed* or *long*;

8. *Time frame to achieve solution* – possible values include: *quick*, *relaxed* or *long*.

The information listed above is a general characterization of the exceptional event. It is complemented with some specific information identifying the concrete scenario, e.g., the affected instances and the responsible person. The complete information is not shown here due to space limitations.

1.6.4. Exception handling strategies

The following dimensions to classify exception handling strategies are identified [Mourão and Antunes, 2005]:

(i) *Objective of the intervention* – describes the general goal for the intervention, e.g., to abort the instance;

(ii) *Communication type* – *synchronous* or *asynchronous*. This dimension classifies the way people exchange information to share the situation knowledge/understanding;

(iii) *Collaboration level* – *one person* solves the situation; several persons solve the situation in a *coordinated mode*; or several persons solve the situation in a *collaborative mode*. The involved actors may implement recovery actions in a coordinated mode, meaning that they are aware of each other's activities, while in collaboration mode they only know a general description of the intended objective agreed during the last collaborative session;

(iv) *External monitoring* – there is either *enough information* to achieve the best solution or *additional information* must be collected from the environment;

(v) *Tools to determine the best solution* – either *no external decision aids* are required or there is a need for *advanced support* to achieve the best solution.

This classification affords linking the high-level handling strategies with a specific set of tasks available at the system level. The *communication type* expresses how the collaboration support component will interconnect the persons involved in the handling process. Two types of communication are differentiated: synchronous and asynchronous. In synchronous communication, the involved actors exchange information in real time (in face-to-face interactions or using some electronic means to transfer information), whereas in asynchronous communication information is exchanged in deferred time.

As mentioned in Section 1.6.1, coordinating activities among users is an important aspect of our solution because the coordination facet of traditional WfMS is relaxed. The two modes of operation identified in the *collaboration level* strategy reflect the concern with coordination. In a coordinated mode, users may choose any available tool to coordinate their activities. In a collaborative mode, the coordination aspects are not relevant since users implement their activities in an concerted way.

1.7 Solution Architecture and Implementation

In this section we start by describing the solution's architecture and its integration with the environment, the WfMS and the actors involved in the exception handling process. In Section 1.7.2 the solution implementation by a dedicated workflow is discussed. Section 1.7.3 describes the recovery and monitoring operations that users may implement during unstructured activities.

1.7.1. Architecture

To introduce the solution's architecture, we detail in Figure 1.6 our extended reference model. The figure has been reorganized to place the Exception Handling Service at the top. Other components were readjusted in conformity. Additional detail was also added to the figure. In the architecture design, we have mainly focused on the interface with Workflow Enactment Service because our objective is to control the system behaviour at runtime. We will assume the existence of primitives to implement interfaces C and B in the figure.

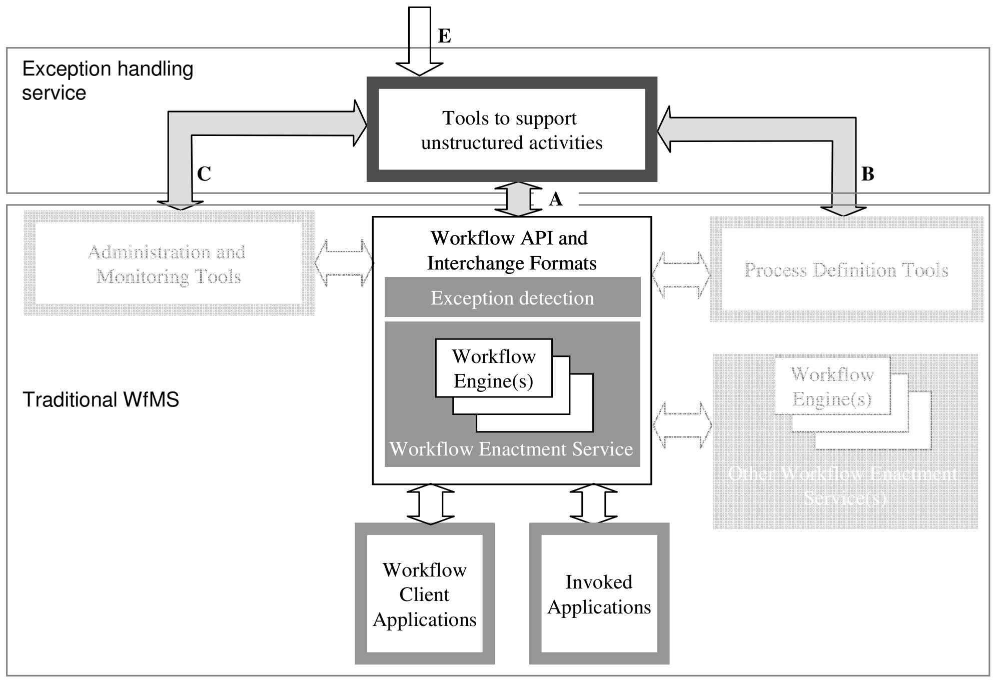

E
Exception handling service
Tools to support unstructured activities
C
B
A
Administration and Monitoring Tools
Workflow API and Interchange Formats
Exception detection
Workflow Engine(s)
Workflow Enactment Service
Process Definition Tools
Workflow Engine(s)
Other Workflow Enactment Service(s)
Traditional WfMS
Workflow Client Applications
Invoked Applications

Figure 1.6. Detailed version of the extended reference model reorganized. The external interface E and an exception detection component placed close to the enactment services were added.

Figure 1.7 gives more detailed on the model components, comprising the Exception Handling Service, Workflow Enactment Services (represented by the workflow engine), Exception Detection Component, workflow client and invoked applications. Interface A and E are also illustrated. Dashed lines represent information flows whereas uninterrupted lines represent control flows.

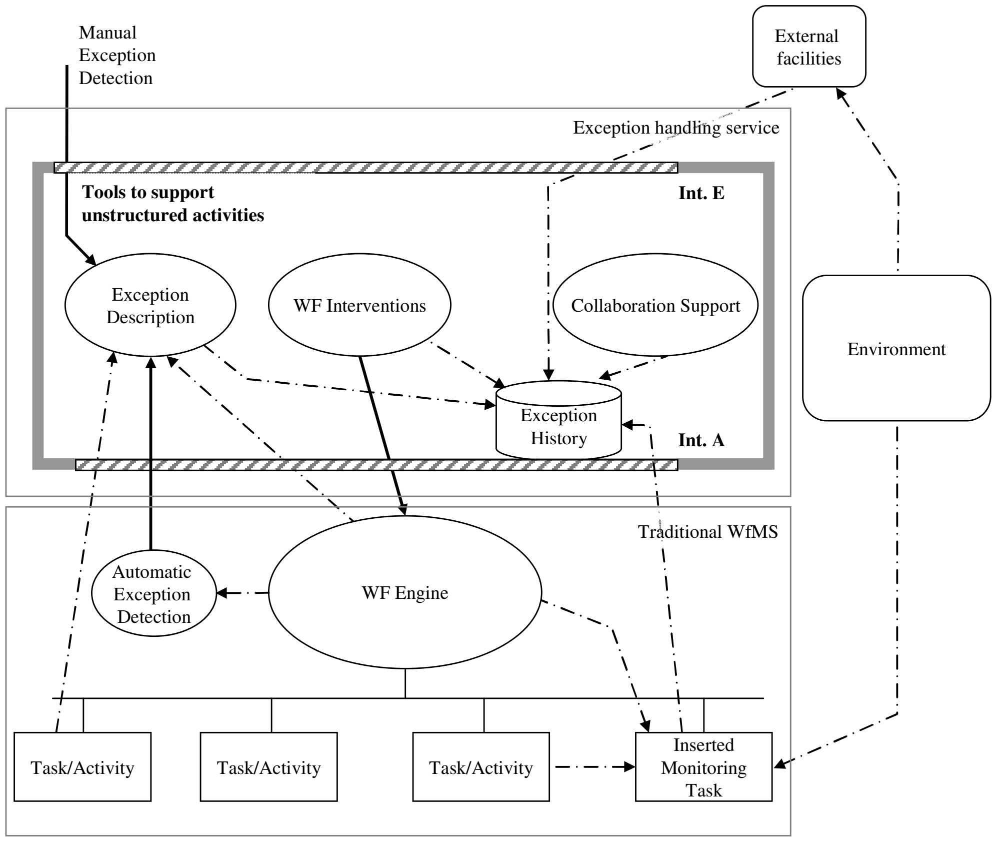


Figure 1.7. The solution's detailed architecture and its integration with the WfMS and the environment.

Four components are identified as belonging to the exception handling service: *Exception Description*, *WF Interventions*, *Collaboration Support* and *Exception History*. Two distinct interfaces are also identified: interface A and E. The External Facilities, illustrated at the top, represent any exception handling activity carried outside the solution's boundary and will be discussed bellow together with the interfaces.

The *Exception Description* component supports the diagnosis process described in Section 1.6.3. The *WF Interventions* component implements the functions associated to objective of the intervention described in Section 1.6.4. The *Collaboration Support* component implements the communication type and collaboration level mechanisms also described in Section 1.6.4. Finally, the *Exception History* component stores all log information associated to the exception handling cycles.

The traditional WfMS supported by the proposed solution is represented at the bottom of Figure 1.7. Naturally, the workflow engine plays a central role. Close to the engine, the *Automatic Exception Detection* component collects information from it and, when an exception is detected, control information is transferred to the exception handling service. The *Inserted Monitoring Task* at the bottom right represents a task to collect information that users decided to insert during the exception handling activities.

Concerning interfaces, the interface A links the exception handling components with the WfMS, while interface E links these components with the users and external environment. Interface A is used to collect information about the WfMS status, to implement low level recovery actions (launch/suspend tasks, etc.), and to automatically detect and signal exceptions.

Interface E connects the exception handling service with users to enable manual exception detection and interaction during exception handling activities. Interface E also supports environmental information gathering about the operations carried outside the framework's scope (cf. Section 1.6.2).

In Figure 1.8, the three remaining operator's basic functions (cf. Section 1.6.1) are added: diagnosis, recovery and monitoring. The diagnosis, recovery, and monitoring functions are carried out by the involved actors with support and orchestration from the components available at Tools to Support Unstructured Activities. Figure 1.8 represents the information and control flow through interface E between the operators and the system and between External Facilities and the system.

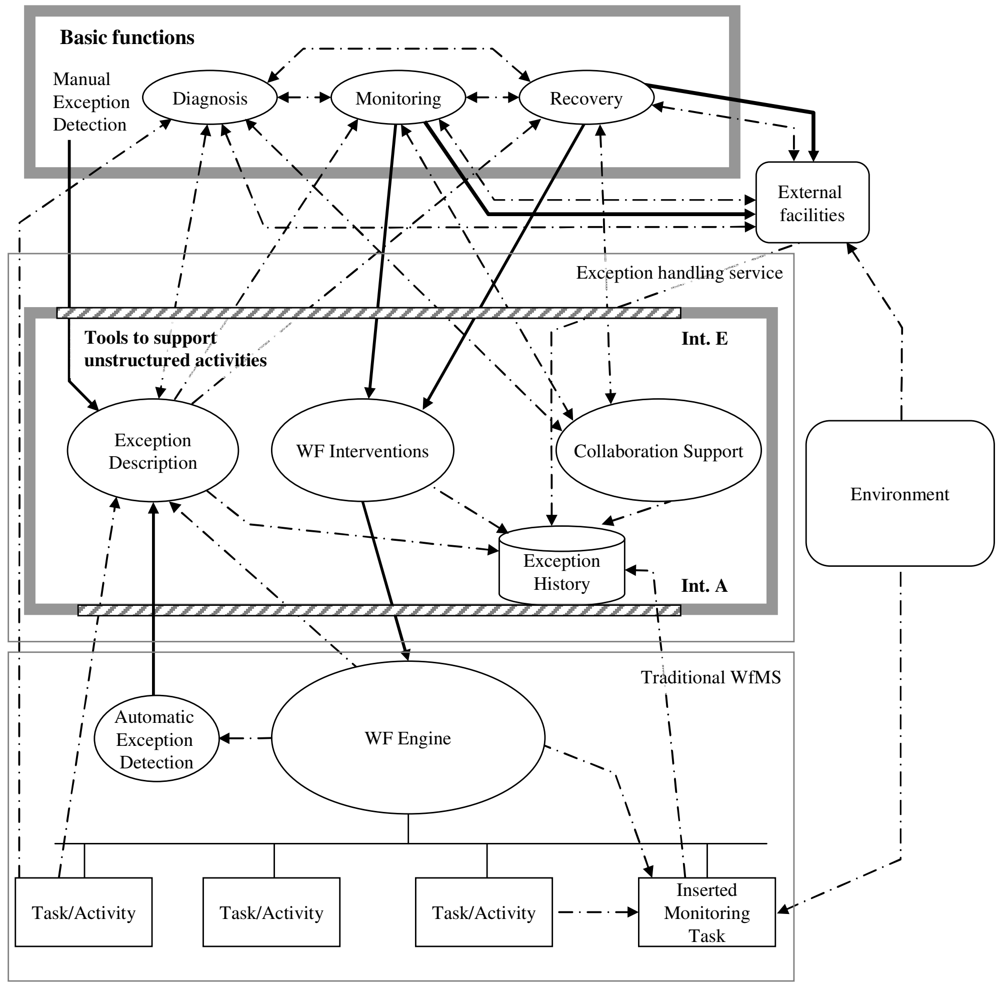


Figure 1.8. Solution's architecture and its integration with the WfMS, environment and operators' basic functions

1.7.2. Exception handling workflow

We also claim it is better to cope with ad hoc effective unexpected exceptions in work models using work models [Sadiq, 2000]. We implemented our solution using the OpenSymphony suite of components [The OpenSymphony project, 2005].

When an exceptional event is triggered, the system instantiates the exception handling workflow process represented in Figure 1.9 and initialises some of the exception description parameters mentioned in Section 1.6.3. There are two alternative ways to instantiate this process: either by system (interface A) or by user detection (interface E). Then, using the Edit Info task, available right after detection, users may refine the exception information that was issued at detection.

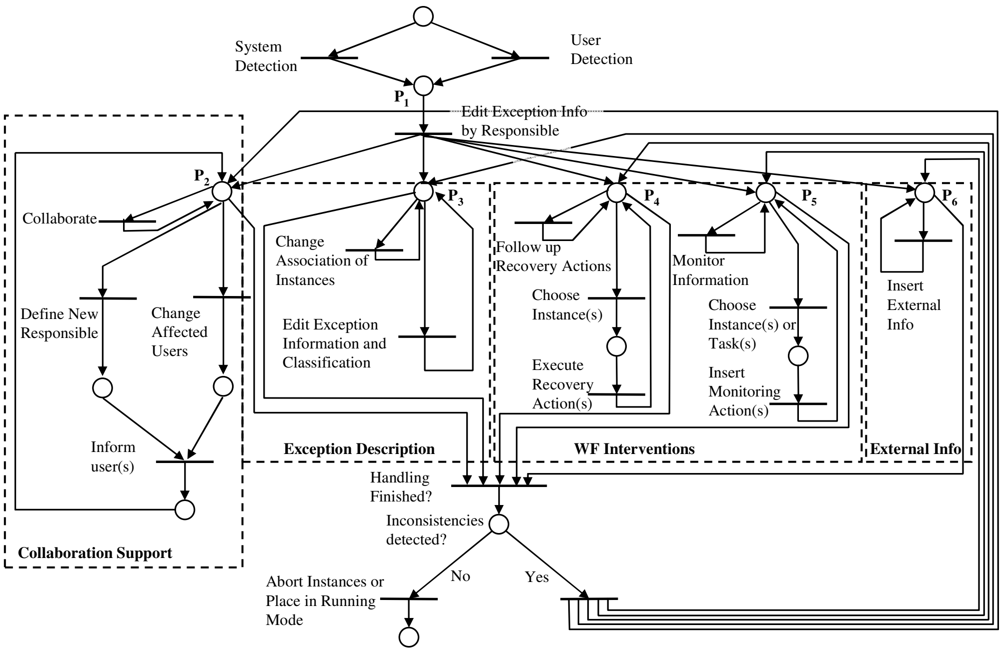

System Detection
User Detection
P1
Edit Exception Info by Responsible
P2
Collaborate
Define New Responsible
Change Affected Users
Inform user(s)
Collaboration Support
P3
Change Association of Instances
Edit Exception Information and Classification
Exception Description
P4
Follow up Recovery Actions
Choose Instance(s)
Execute Recovery Action(s)
WF Interventions
P5
Monitor Information
Choose Instance(s) or Task(s)
Insert Monitoring Action(s)
P6
Insert External Info
External Info
Handling Finished?
Inconsistencies detected?
No
Yes
Abort Instances or Place in Running Mode

Figure 1.9. Exception handling workflow

After this task the system activates four components:

- Collaboration support;
- Exception description;
- WF Interventions;
- Insert external info.

The *WF Interventions* component is implemented by independent recovery and monitoring threads. The interface E, identified in Figure 1.8, is implemented by the thread External Info.

The *Collaboration Support* component supports users specifically collaborating within the scope of an exceptional event. The tasks implemented by this component (see Figure 1.9) enable involving more actors in exception handling and implement the collaboration mechanism. The *Collaborate* task implemented by this component may be synchronous or asynchronous and at any time the users may choose which type to use.

The *WF Interventions component* is implemented with two threads: implement recovery actions and insert monitoring tasks. The specific actions implemented by this component enable users to implement recovery activities to bring the system back into a coherent state. They will be described in Section 1.7.3. The monitoring thread affords users to insert monitoring tasks that store exceptional relevant information in the *Exception History*.

The component *Insert external info* implements interface E, allowing users to insert information associated to the event, either environmental information collected or information about tasks executed outside the workflow scope.

When users identify that coherence has been achieved, they execute the task Handling Finished? (at the bottom of Figure 1.9), removing the marks from places P_2 to P_6 and suspending the support to unstructured activities. Before finalizing the exception handling process, it is necessary to verify whether inconsistencies were inserted into the affected instances (cf. Section 1.6).

1.7.3. Recovery actions, monitoring actions and support users removing inconsistencies

In Section 1.6.4 the high level objectives of the intervention were integrated into the handling strategies. To support users implementing these objectives, a set of quasi-atomic recovery actions are available. The Recovery Actions thread shown in Figure 1.9 affords operators to implement this functionality on the selected workflow instance(s).

The following list of actions is currently available in the implemented solution [Eder and Liebhart, 1995; Agostini and De Michelis, 2000; Reichert et al., 2003; Chiu et al., 2001; Sadiq, 2000]:

- *Suspend/resume instance* – this action involves suspending or resuming instance execution;
- *Abort instance* – abort the instance;
- *Backward jump* – jump to a previous executed location in the work model;
- *Forward jump* – jump forward to a task in the work model;
- *Jump* – jump to another location in the model (this location is neither in the previous executed tasks nor in the upcoming tasks);
- *Move operation* – move one task to another location in the model;
- *Ad hoc refinement* – execute one action from a pre-defined list;
- *Ad hoc extension* – choose a new path or change the model.

Since the *ad hoc refinement* operation inserts threads executing in parallel to the actual instance execution it may also be used in monitoring actions.

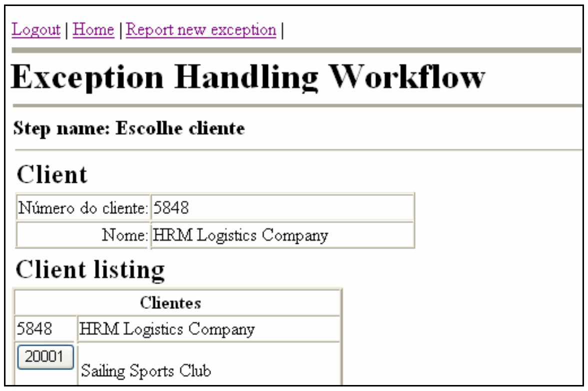
1.8 Evaluation

We have been able to test our solution in a concrete organization: a Port Authority. During the system tests, we were able to follow an exception since it was detected until it got solved. In this section we discuss this event.

The Port Authority has the responsibility to manage all business activities within its jurisdiction that includes the river and the shore side. The Port Authority manages all vessels and cargo transfers to and from ships. All commercial activities installed on the shore are also under the jurisdiction of the Port Authority that issues licences and contracts for the rented places.

The business process modelled by the Port Authority refers to managing space rentals. Companies and individuals rent spaces for their business activities for which they pay a fixed amount in a regular basis. For each rented space, there is a contract between the client and the Port Authority expressing all the conditions governing the business agreement. A department with about 10 employees negotiates all contracts, manages client related information and assures clients pay on time. These administrative processes were modelled using the workflow platform. At the end of every month, the system automatically instantiates a process for every rented space that is supposed to pay its fee. A list of debts and free/occupied zones must be generated at any moment. To describe the exceptional event, user names were changed to preserve anonymity.

Assume that Henry is updating the client's information when he is informed that the client has bankrupted. Figure 1.10 shows the Web page for the client editing task, where the link to manually signal an exception is shown at the top.



Exception Handling Workflow

Step name: Escolhe cliente

Client

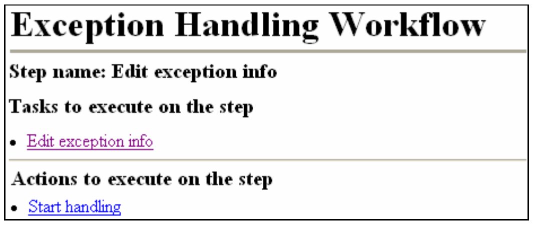
Número do cliente:	5848
Nome:	HRM Logistics Company

Client listing

Clientes	
5848	HRM Logistics Company
20001	Sailing Sports Club

Figure 1.10. Web page for the client edit workflow

After selecting this link, the user is prompted with the EHW page shown in Figure 1.11. From there, the exception classification must be accomplished, as shown in Figure 1.12. Henry realizes that time is not critical and classifies it as relaxed. He also affects John, his direct supervisor, to the exception handling process. He does not define John as responsible because he wants to talk with him first. He inserts a brief description and classifies the exception as an external event with departmental impact. He also defines the exception as a true exception, since it never happened before. The dimensions scope, affected instances, and responsible were automatically defined by the system.



Exception Handling Workflow

Step name: Edit exception info

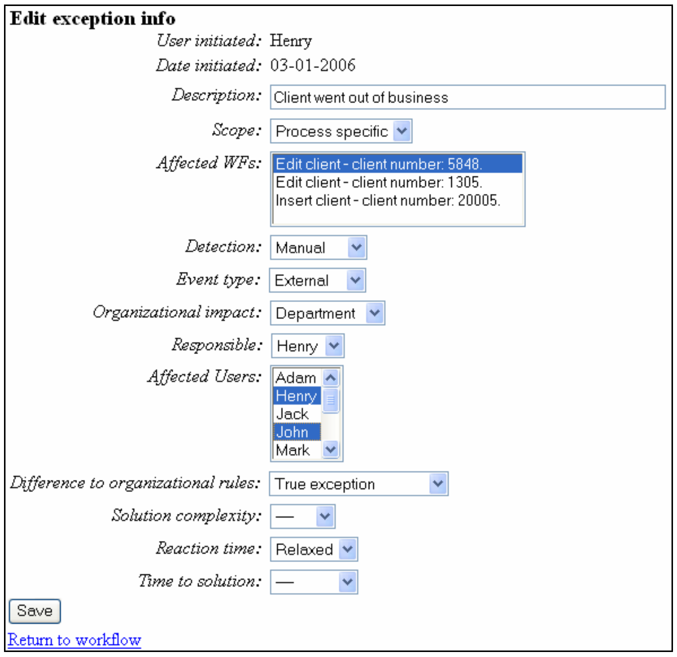
Tasks to execute on the step

- Edit exception info

Actions to execute on the step

- Start handling

Figure 1.11. Exception handling workflow page (EHW)

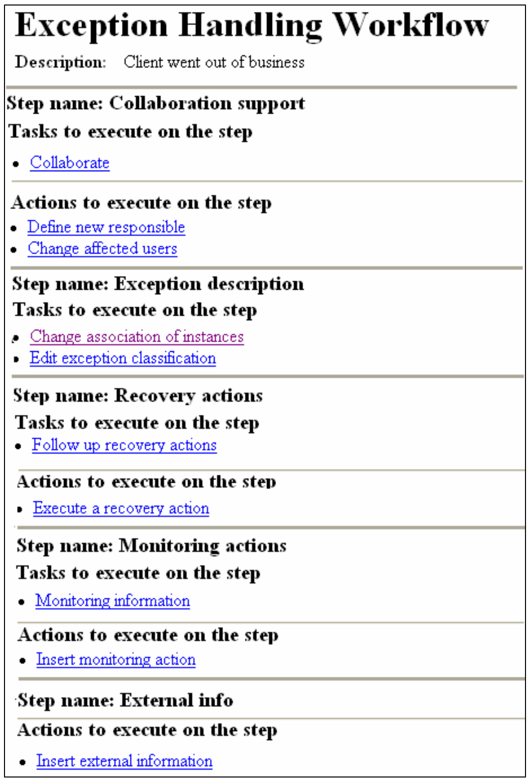


Edit exception info

User initiated:	Henry
Date initiated:	03-01-2006
Description:	Client went out of business
Scope:	Process specific
Affected WFs:	Edit client - client number: 5848. Edit client - client number: 1305. Insert client - client number: 20005.
Detection:	Manual
Event type:	External
Organizational impact:	Department
Responsible:	Henry
Affected Users:	Adam Henry Jack John Mark
Difference to organizational rules:	True exception
Solution complexity:	—
Reaction time:	Relaxed
Time to solution:	—

Figure 1.12. Exception related information

By following the *Start handling* link shown in Figure 1.11, Henry starts handling the exception. An email is generated to John with the exception handling information inserted by Henry and a link to the EHW shown in Figure 1.13.


Exception Handling Workflow

Description: Client went out of business

Step name: Collaboration support

Tasks to execute on the step

- Collaborate

Actions to execute on the step

- Define new responsible
- Change affected users

Step name: Exception description

Tasks to execute on the step

- Change association of instances
- Edit exception classification

Step name: Recovery actions

Tasks to execute on the step

- Follow up recovery actions

Actions to execute on the step

- Execute a recovery action

Step name: Monitoring actions

Tasks to execute on the step

- Monitoring information

Actions to execute on the step

- Insert monitoring action

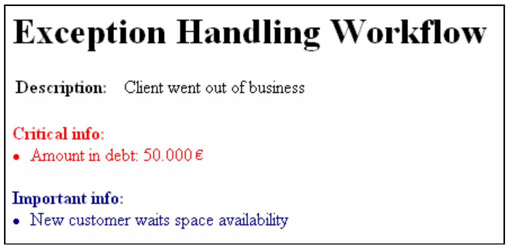
Step name: External info

Actions to execute on the step

- Insert external information

Figure 1.13. EHW page handling the 5 parallel branches of the exception handling workflow

John may then look at the situation in the EHW page and start a collaboration *task* with Henry. He decides using instant messaging. During the conversation, John realizes another company is requesting the space occupied by the company. He also recognizes that the client's debt is 50.000€. John tells Henry to insert this alert in the EHW (cf. Figure 1.14) and involves Philip, from the lawyer department, in the exception handling process. John also decides to insert a monitoring task to identify whether the client has any other debts.


Exception Handling Workflow

Description: Client went out of business

Critical info:

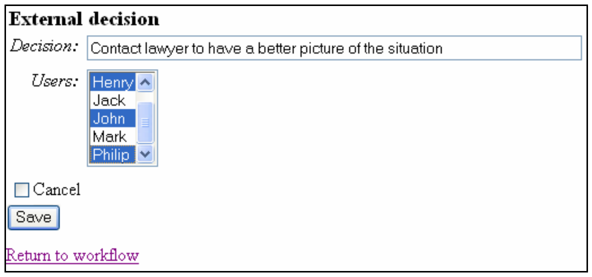
- Amount in debt: 50.000 €

Important info:

- New customer waits space availability

Figure 1.14. EHW displaying alert messages at the top

Philip is informed about the situation by email. After reading the email message, he decides to phone Henry to discuss the details. During the phone conversation, they decide Philip will consult an external expert. Philip inserts a comment about this decision in the external information UI shown in Figure 1.15. Henry will wait for any news.


External decision

Decision: Contact lawyer to have a better picture of the situation

Users: Henry, Jack, John, Mark, Philip

Figure 1.15. Inserting external decision-making information

Philip finds out from the expert that the Port Authority should notify the client by standard mail, giving 5 days to pay the debt. Obtaining no response, they should start a lawsuit action. Philip writes a letter draft and attaches it to the workflow as an entry message in the "edit exception classification" *action*. He then uses the "collaboration support" *step*, to discuss with Henry and John who will send the letter and who will follow this external action. The email mechanism is adopted for that purpose.

Henry will be in charge of the external recovery action. If Henry finds out the company pays the older debts they have to reanalyze the situation. Again, Philip and John are notified about the new events. If the client pays all the old debts they will close the exception handling process.

The system managed the interactions among users to handle this particular case. It was easy to involve an expert from another department in the handling process. Relevant decisions and event related information were easily spread through the involved users improving their knowledge about the situation details and their evolution. The relevant information related to the situation was also attached to the event so it can be used in future events.

1.9 Exercises

1. Consider a situation where a truck with a very important delivery is stuck on a traffic jam. Any delay on the material delivery is charged heavily on the company: it is more expensive than getting a new delivery sent by an airplane.

    a. What challenges to the workflow system emerge from this situation?

    b. What tasks should the operators initiate to follow up the situation and try to get the most adequate solution?

2. A sales representative receives a complaint from a customer about a non-implemented functionality in a good that the company produces during a contact visit requested by the client wanting to by those goods. He then realizes that the functionality is very important and that some of the company most important competitors already implement it. What actions should the

sales representative initiate regarding this process instance? Who should be involved if the production department does not agree on time with the request?

3. A manufacturing company receives an alteration to an already placed order. The company is interested in satisfying the customer because it is a very important one. Nevertheless, the company has to calculate the minimum costs associated to the change and all the production departments must be involved. Describe the tasks that the marketing director responsible for the customer has to develop to satisfy the client.

1.10 Suggested additional reading

Worah and Sheth [1997] present a good overview about systems to handle failures. On the area of expected exceptions the reader may consult the work by Casati et al. [Casati et al., 1999; Casati, 1998] or by Chiu et al. [2001]. Interesting work on the area of extended expected exceptions where Case Base Reasoning is used to extend the matching capabilities for the event identification have been developed by Luo [2001]. Exception mining techniques are explored by Grigori [2001] and Hwang [1999].

A good survey on metamodel approaches is presented by Rinderle et al. [2004]. The work developed by Ellis et al. [1995] is an important milestone because it establishes the notion of dynamic change bug. Relevant readings on this subject are the works from Aalst and Basten [2002], Agostini and De Michelis [2000], Casati [1998], Reichart and Dadam [1998], Sadiq [2000] and from Weske [2001].

On the open point systems there are relevant works from Deiters and Gruhn [1994] and Hsu and Kleissner [1996]. Finally, further details on the proposed solution including an implementation description can be consulted in [Mourão, 2008].

References

Abbott, K. R. and Sarin, S. K., Experiences with workflow management: issues for the next generation, in *CSCW'94*, (Chapel Hill, USA, 1994), 113-120.

Agostini, A. and De Michelis, G., A light workflow management system using simple process models, *Computer Supported Cooperative Work*, 9, 3 (2000), 335-363.

Bernstein, A., How can cooperative work tools support dynamic group process? bridging the specificity frontier, in *CSCW '00*, (Philadelphia, 2000),279-288.

Bowers, J., Button, G., and Sharrock, W., Workflow From Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor, in *ECSCW'95*, (Stockholm, 1995),51-66.

Casati, F., *Models, Semantics, and Formal Methods for the Design of Workflows and their Exceptions*. PhD Thesis, Politecnico di Milano, 1998.

Casati, F., Ceri, S., Paraboschi, S., and Pozzi, G., Specification and Implementation of Exceptions in Workflow Management Systems, *ACM Transactions on Database Systems*, 24, 3 (1999), 405-451.

Chiu, D. K., Li, Q., and Karlapalem, K., WEB Interface-Driven Cooperative Exception Handling in ADOME Workflow Management System, *Information Systems*, 26, 2 (2001), 93-120.

Deiters, W. and Gruhn, V., The Funsoft Net Approach to Software Process Management, *International Journal of Software Engineering and Knowledge Engineering*, 4, 2 (1994), 229-256.

Dellarocas, C. and Klein, M., A Knowledge-based approach for handling exceptions in business processes, in *WITS'98*, (Helsinki, Finland, 1998).

Eder, J. and Liebhart, W., The Workflow Activity Model WAMO, in *Int. Conf. on Cooperative Information Systems*, (Vienna, 1995).

Ellis, C., Keddara, K., and Rozenberg, G., Dynamic change within workflow systems, in *Proc. of conf. on Organizational computing systems*, (Milpitas, USA, 1995), 10-21.

Ellis, C. and Nutt, G. J., Modeling and enactment of workflow systems, in *Application and Theory of Petri Nets*, (Chicago, USA, 1993), Springer-Verlag,1-16.

Grigori, D., Casati, F., Dayal, U., and Shan, M. C., Improving Business Process Quality through Exception Understanding, Prediction, and Prevention, in *VLDB'01*, (Rome, 2001).

Guimarães, N., Antunes, P., and Pereira, A. P. The Integration of Workflow Systems and Collaboration Tools. *Advances in Workflow Management Systems and Interoperability*. Istambul, 1997.

Han, Y., Sheth, A. P., and Bussler, C., A Taxonomy of Adaptive Workflow Management, in *CSCW'98*, (Seattle, USA, 1998).

Heinl, P., Exceptions During Workflow Execution, in *EDBT'98*, (Valencia, Spain, 1998).

Hollingswoorth, D. *Workflow Management Coalition - The Reference Model TC00-1003*. WFMC, 1995.

Hsu, M. and Kleissner, K., ObjectFlow: Towards a process management infrastructure, *Distributed and Parallel Databases*, 2 (1996), 169-194.

Hwang, S. Y., Ho, S. F., and Tang, J., Mining Exception Instances to Facilitate Workflow Exception Handling, in *6th Int. Conf. on Database Systems for Advanced Applications*, (Hsinchu, Taiwan, 1999).

Luo, Z., *Knowledge sharing, Coordinated Exception Handling, and Intelligent Problem Solving for Cross-Organizational Business Processes*. PhD Thesis, Dep. of Computer Sciences, University of Georgia, 2001.

Mourão, H. R., *Supporting Effective Unexpected Exception Handling in Workflow Management Systems Within Organizational Contexts*. PhD Thesis, University of Lisbon, 2008.

Mourão, H. R. and Antunes, P., Supporting Direct User Interventions in Exception Handling in Workflow Management Systems, in *CRIWG 2003*, (Autrans, France, 2003), Springer-Verlag, 159-167.

Mourão, H. R. and Antunes, P., Exception Handling Through a Workflow, in *CoopIS 2004*, (Agia Napa, 2004), Springer-Verlag, 37-54.

Mourão, H. R. and Antunes, P., A Collaborative Framework for Unexpected Exception Handling., in *CRIWG 2005*, (Porto de Galinhas, 2005), Springer-Verlag,168-183.

The OpenSymphony project. (2006, 28/11/06) <Http://www.opensymphony.com>

Reichert, M. and Dadam, P., ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control, *Journal of Intelligent Information Systems*, 10, 2 (1998), 93-129.

Reichert, M., Dadam, P., and Bauer, T., Dealing with Forward and Backward Jumps in Workflow Management Systems, *Software and Systems Modeling*, 2, 1 (2003), 37-58.

Rinderle, S., Reichert, M., and Dadam, P., Correctness criteria for dynamic changes in workflow systems - a survey, *Data and Knowledge Engineering*, 50, 1 (2004), 9-34.

Saastamoinen, H., *On the Handling of Exceptions in Information Systems*. PhD Thesis, University of Jyväskylä, 1995.

Sadiq, S. W., On Capturing Exceptions in Workflow Process Models, in *Proceedings of the 4th International Conference on Business Information Systems*, (Poznan, Poland, 2000).

Schmidt, K., Of maps and scripts - the status of formal constructs in cooperative work, in *GROUP '97*, (Phoenix, USA, 1997),138-147.

Sheth, A. P., Georgakopoulos, D., Joosten, S. M., Rusinkiewicz, M., Scacchi, W., Wileden, J., and Wolf, A. L., *ACM SIGMOD Record*, 25, 4 (1996), 55-67.

Strong, D. M. and Miller, S. M., Exceptions and Exception Handling in Computerized Information Systems, *ACM Transactions on Information Systems*, 13, 2 (1995).

Suchman, L. A., Office procedure as practical action: models of work and system design, *ACM TOIS*, 1, 4 (1983), 320-328.

Suchman, L. A., *Plans and Situated Actions*. MIT Press, 1987.

van der Aalst, W. and Basten, T., Inheritance of workflows: an approach to tackling problems related to change, *Theoretical Computer Science*, 270, 1 (2002), 125-203.

van der Aalst, W. and Berens, P., Beyond Workflow Management: Product-Driven Case Handling, in *GROUP 2001, (*Boulder, USA, 2001).

Weske, M., Formal foundation and conceptual design of dynamic adaptations in a workflow management system, in *International Conference on System Sciences,* (2001), 2579-2588.

Worah, D. and Sheth, A. P. Transactions in Transactional Workflows. In Jajodia, S. K. Larry, ed. *Advanced Transaction Models and Architectures*. Kluwer Academic Publishers, 1997.

WFMC, 1999*, Workflow Management Coalition - Terminology & Glossary TC00-1011*.