A Collaborative Framework for Unexpected Exception Handling

Hernâni Mourão[1*], and Pedro Antunes[2]

[1] Escola Superior de Ciências Empresariais, Instituto Politécnico de Setúbal, Campus do IPS – Estefanilha, 2914-503 Setúbal, Portugal, and
LASIGE (Laboratory of Large Scale Information Systems)
hmourao@esce.ips.pt
[2] Faculdade de Ciências, Universidade de Lisboa, 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. This paper proposes a collaborative framework handling unexpected exceptions in Workflow Management Systems (WfMS). Unexpected exceptions correspond to unpredicted situations for which the system can not suggest any solutions. We introduce the notion that exception recovery is a collaborative problem solving activity that should be addressed through an intertwined play between several actors performing two types of tasks: (1) diagnosing situations; and (2) planning recovery actions. We propose a set of dimensions to classify the exceptional situations and their relations to recovery strategies. We also discuss the importance of monitoring recovery actions within the scope of diagnosis tasks. The proposed solution is implemented through a dedicated workflow.

1. Introduction

The work processes carried out by organizations in their daily operations have been identified to belong to a continuum ranging from totally unstructured to completely structured [33]. The majority of the available organizational information systems fall close to both sides of the spectrum boundaries [33]. In particular, traditional WfMS fall into the highly structured boundary, usually supporting organizational processes through the execution of work models. In the context of Schmidt's work [32], work models play the role of scripts in formal organizational structures and have a normative engagement. Closer to the other end of the spectrum limits, Suchman [35] 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 a normative trait.

* The author is deeply grateful to Ulm's University group for their friendship and scientific support during his staying. Their sharing of ideas and great field experience on Workflows was very important to this work.

Lecture Notes in Computer Science, vol. 3706, pp. 168-183. Groupware: Design, Implementation, and Use. H. Fuks, S. Lukosch and A. Salgado, Eds. Heidelberg, Springer-Verlag.

To support the various organizational needs, WfMS should cope with the whole spectrum of structured and unstructured activities. This requirement has been identified by Ellis and Nutt [18], when they realize that WfMS must be flexible to succeed. Also, Abbot and Sarin [1], based on empirical evidence, claim it is necessary to integrate procedural and nonprocedural work in WfMS. They define nonprocedural work as "unchoreographed interactions between people."

In the WfMS community nonprocedural work has been designated "exception handling," encompassing the set of actions aiming to react to a kind of event that is out of the scope of the work model. Exceptions either can not be predicted during the design phase or, although being predictable, are deliberately excluded from the work model to reduce complexity [6; 10; 14; 24; 31]. The Eder and Liebhart's [14] classification of expected and unexpected exceptions has been widely accepted, since it enables a division between the exceptions that can be predicted in the design phase from those that can not. In our work, we advocate a novel approach to exception classification, assuming a continuum from expected to unexpected exceptions.

This paper is structured in the following way. We start by revising the notion of completeness in an exception handling WfMS. Then, we present a framework to deal with the above mentioned dichotomy: maintain model-based work whenever possible and change to a kind of map guidance whenever the scope is outside the limits under which the work models were designed [2]. A set of guidance mechanisms is proposed to support users dealing with unexpected exceptions, delivered in the form of contextual information about the affected processes. As Bernstein [5] states, emergent actions must be sustained by context information as actors dealing with these environments become overloaded with information.

Our solution is based on a previous developed exception handling workflow [25]. In the present work we expand the exception handling workflow with three functions [31]: detection, diagnosis, and recovery. Most importantly, the recovery phase is now intertwined with the diagnosis phase, as we came to realize that a proper diagnosis is an iterative process requiring harvesting contextual information and collaboration between users.

We also classify exception handling strategies and relate them with exceptional situations. Finally, we present and discuss the implementation details, illustrate the framework usage with an example, and finish with the conclusions and future work.

2. Revising the Completeness Requirement

To be complete, an exception handling system should consent users to carry out recovery activities 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. Several impacts of this definition should be taken into consideration.

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 objective of reaching a solution, they will overcome the system [21; 34].

The consequences of this open perspective on WfMS are profound. The common restrictions to ad hoc model changes, based on structural and dynamic properties, must therefore be relaxed. We believe that 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 but allowed to carry out that action.

On the other hand, users should not be restricted to the services provided by the exception handling system, since they will use everything needed to overcome the situation. The challenge is to implement a comprehensive set of services that may reduce the user's needs to handle exceptions outside the system scope. The framework described in this paper integrates such services while being open to the organizational environment. Thus, some exception handling activities will be partially outside the WfMS scope. The framework integrates environmental information about external activities, thus guiding actors in the course of actions, but will not assume control of those activities. These environmental monitoring tasks collect information necessary to plan the recovery actions or monitor the evolution of actions implemented out of the framework's scope.

3. Related Work and Scope of the Framework

The main appointed reasons for the lack of flexibility in current WfMS are: 1) complex work models where only predictable events are foreseen (expected exceptions, see below) [6; 7; 11]; 2) inability applying model changes to already running instances [17; 28; 36]; 3) difficulties applying ad hoc changes to cope with very small model variations [28; 36]; 4) tight coupling between modeling and enactment [19; 23]; and 5) formal models currently adopted to represent work are inadequate to flexibility support [13].

Various approaches to flexibility can be found in the literature, as different authors understand differently the properties a WfMS should exhibit to effectively deal with office work. We identify two parallel research streams [20]: meta-model and open-point. Meta-model approaches take into major consideration structural and dynamic constraints to model adaptations, while open-point approaches rely on the users' abilities to assure that no inconsistencies are inserted in the system.

The meta-model approaches fundamentally address expected exceptions, coded in special constructs and invoked whenever a predefined exceptional situation is detected [6; 10; 11; 14] – e.g., Event Condition Action (ECA) rules. Several techniques, such as exception mining [10; 22], case base reasoning [24], conversational case base reasoning [37], and knowledge bases [12] have been proposed to expand the system flexibility handling exceptions. If we consider a continuum from expected exceptions to completely unexpected exceptions, all these systems handle events falling close to expected exceptions limits of the spectrum.

On the meta-model approaches to address unexpected exceptions, we find several solutions [7; 17; 26; 36]. The most important distinction from the previous set, is that these solutions support dynamic changes and ad hoc interventions.

Regarding the open-point approaches in more detail, we find interactive enactment [23] and flexible enactment [19]. These approaches assume work models are incompletely specified, allowing users to interactively adapt them, e.g., inserting tasks. This increases the degree of freedom on the user side to cope with deviations between the work models and the real world, although in a more structured way than a totally open-point intervention would afford. In any case, users will be able to insert unidentified inconsistencies, and possibly put the WfMS at risk [20], considering that no dynamic or structural checks are made.

Like Agostini and De Michelis [2], we agree with both research streams delineated above and posit that a WfMS should offer both advantages: being able to work under model guidance and adopt an open-point behavior when model guidance is not applicable. However, after open-point operations, the system should support users bringing instances back to model control, while identifying potential flow and data inconsistencies. A complete discussion of the mechanisms necessary to bring the system under model control is out of the scope of the present paper. It is though important to mention that these mechanisms are highly constrained by the meta-model assumptions. We point to [29] on this issue.

Also studying the integration between meta-model and open-point approaches, Bernstein [5] proposes a stepwise solution with four stages. The handling activities incrementally progress from totally unspecified to totally specified control. Contrary to this solution, which relies on AI techniques to support the incremental steps, our approach relies on user collaboration.

In summary, our main focus is on exceptions that can not be handled in an automatic way, i.e., can not be dealt by any of the solutions enlarging the original notion of expected exceptions (thus moving close to the unexpected limit). We assume that users should be able to flexibly move the system behavior from totally defined to unstructured processes, where open-point operations are carried out while meta-model assumptions are used to check system coherence. This will enable the adoption of the best strategy to the exceptional situation and facilitates the identification of user inserted inconsistencies. Finally, the system should also support the user to identify the necessary actions to bring the system back to a coherent state.

4. Exceptions Handling Framework

We distinguish three functions in exception handling [12; 31]:

- exception detection
- situation diagnosis
- exception recovery actions

Exception detection has been extensively studied in previous works [6; 10; 11; 31; 25]. Detection can be manual or automatic. A detailed description of the automatic detection techniques is out of the scope of this paper as it is focused on user perspective. We assume that an exception detection component is tightly integrated in

the workflow engine, covering the most common situations, such as data, workflow, and temporal events, non-compliance events and application events. Later in this section we will discuss the integration of the detection component. We distinguish manual and automatic detection as they behave differently from user's perspective.

We will rather focus on the other two functions. In our framework we advocate an intertwined play between diagnosis and recovery until the exception is resolved. That is to say, the diagnosis is not considered to be complete on the first approach but rather, through an iterative process where different actors may collaboratively contribute to the solution. We should also stress that both the exceptional situation and perception of the situation may change along this iterative process, as new information is made available. As an example concerning a clinical process, a doctor may decide to insert a new task to collect information on the patient's status and display this information to everyone involved. Also, if the clinical conditions of the patient change, the diagnosis regarding the exceptional situation may also change and some new objectives and tasks may arise.

After diagnosis, users carry out recovery actions. The open nature of the framework indicates 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. This issue will be addressed later in more detail.

Besides the detection, diagnosis and recovery functions, we identify a new function addressing monitoring actions necessary to control the progress of the whole exception handling process. These monitoring actions allow users to collect up to date information related to running instances and tasks. Considering again the open nature of the framework, these monitoring actions may also bring environmental information to the system, e.g., establishing a link to a web site with traffic information may help solving the exception of a truck being stuck on a traffic jam. In other cases monitoring actions may require more sophisticated links to external services, e.g., invoking an existing tool to calculate the minimum impact of a machine break down in a lot manufacturing facility. As shown in Figure 1, the exception handling cycle considers monitoring actions running in parallel with recovery actions.

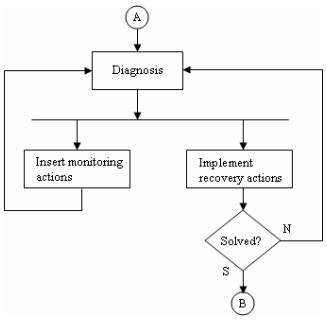

Fig. 1. Exception Handling Cycle

Ellis and Keddara [16] state that a process change is itself a process that can be modeled. Therefore, like Sadik [31], we claim that it is better to cope with unexpected exceptions in work models using a work model. In our framework, the occurrence of

an exception starts an exception handling workflow governed according to the exception handling cycle. The workflow is described in the Implementation section.

Figure 2 illustrates the three different levels of the exception handling framework. Dashed lines represent information flows whereas uninterrupted lines represent control flows. We illustrate the WfMS at the bottom level, including the engine and all running tasks. The mid level represents the system components supporting the exception handling activities.

The top level concerns the users, focused on the two main exception handling functions: diagnosis and recovery/monitoring. The diagnosis and recovery/monitoring functions are carried out by the involved actors with support from the components available in the level below.

Our discussion of completeness in this context requires users not be restricted to the system. Therefore, the "External facilities" component shown in Figure 2 represents what is not under control of the exception handling workflow.

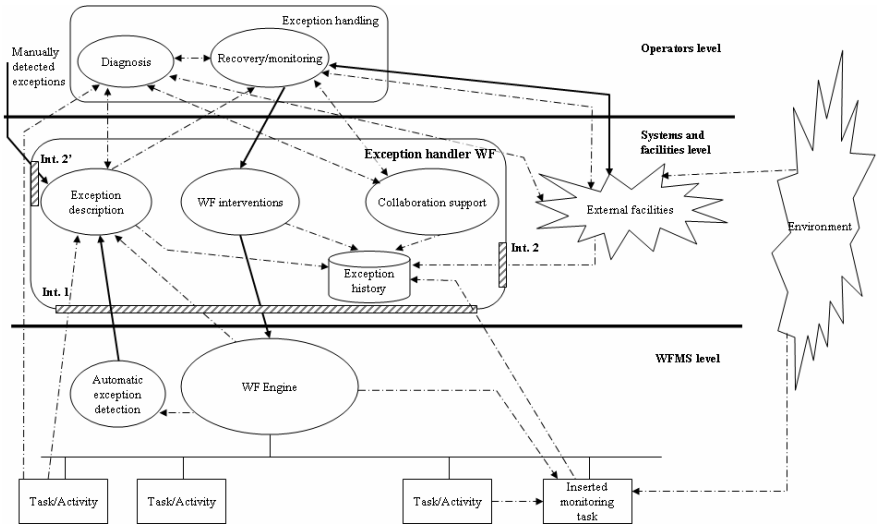

Fig. 2. Operational levels of the exception handling framework

We differentiate two types of activities carried out by the "External facilities": 1) information gathering, collaboration and decision making; and 2) recovery actions. The former group is related to external communication, coordination, collaboration and decision making tools (e.g., meetings, telephone conversations, or operations research techniques). The later group addresses the external recovery actions necessary to resolve the exception. It is our aim that, for any activity executed outside the scope of the exception handling workflow, some environmental information is inserted in the system for monitoring purposes.

The system interfaces are also identified in Figure 2. Interface 1 (Int. 1) interfaces with the WfMS, while interface 2 (Int. 2 and Int 2') interfaces with "External facilities" and implements manual exception signaling. Interface 2 is split to maintain simplicity. Interface 1 is used to collect information about the WfMS status, to implement low level recovery actions (launch/suspend tasks, etc), and to signal

automatically detected exceptions. Through the connection from interface 2 to "External facilities" environmental information about the operations carried outside the framework's scope is collected.

The components *Exception Description*, *WF Interventions*, and *Collaboration support* will be explained later in the Implementation section. Exception detection is also represented in the figure. Manual detected exceptions are inserted by an operator and represented by the uninterrupted line connected to interface 2' on the top left side whereas automatically detection is implemented by the "Automatic exception detection" component located close to the engine. The component uses interface 1 to signal the events to the framework and is located close to the engine because it shares the organization workflow scope.

4.1. Diagnosis

The diagnosis process is mostly dependent on a detailed and accurate assessment of the exceptional event. Using previous classifications [8; 9; 30] and some new added characteristics, we classify exceptional situations using the orthogonal dimensions:

1. Scope – **process specific** when only a small set of instances is affected; or **cross specific** when a large set or different sets of instances are affected. At least one instance must always be associated to the exception;
2. Detection – **automatic** if the exception is automatically detected by the system, or **manually** if the exception is manually triggered;
3. Event type – **data events** related to violation of data rules; **temporal events** when a predefined timestamp occurs; **workflow events** identify special situations at the beginning or ending of tasks or processes, e.g., infinite loops; **external events** are notified by agents or applications external to the WfMS. The assessment of the event type is mandatory, because it directly impacts the handling phase;
4. Impact to organizational goals – **high**, if the particular situation has an important effect on the overall organizational goals; **medium**, if not critical; and **low**, when the organizational goals are not a concern;
5. Organizational impact – **employee**, when only a limited number of employees in the same department are affected by the exception; **group**, when more than one department is affected; and **organizational**, when the overall organization is affected. A responsible person must always be associated to the exception;
6. Difference to the organizational rules – **established exceptions** occur when rules exist in the organization to handle the event but the right ones cannot be found; **otherwise exceptions** occur when the organization has rules to handle the normal event but they do not apply completely to the particular case; and **true exceptions** occur when the organization has no rules to handle the event;
7. Complexity of the solution – **easy**, when the optimal solution can be easily obtained in an acceptable time; **hard**, when the optimal solution is not obtainable within an acceptable time. In this dimension, complexity is not defined as the overall complexity of the handling procedure, but rather an estimation of the possibility to define a cost function based on the available data. Whenever such a function exists, this dimension provides an estimate of the complexity degree to calculate the optimal solution;

8. Reaction time – **quick**, when the reaction to the exception must be as fast as possible; **relaxed**, when the reaction time is not too critical but some decisions must be taken within a time frame imposed by the instance(s); **long**, when the reaction time is not critical. This information is mandatory;
9. Time frame to achieve solution – **quick**, when the situation is expected to be resolved in few working units, normally minutes or hours; **relaxed,** when the time frame is more relaxed, although being a parameter to be taken into consideration, normally measured in working days; and **long** when time is not a critical issue.

Even though some estimates in these dimensions might be available when the event is detected, they can be redefined by users as more information is collected. The old values are kept to maintain an exception history. On the other hand, only the dimensions affected instances, responsible person (organizational impact), event type, and reaction time are mandatory. The user must only insert the most relevant information for the particular situation. This will release the user from inserting information not relevant to handle the concrete situation.

4.2. Recovery

We identified the following dimensions to classify recovery processes:
1. Objective of the intervention – further division presented below;
2. Required type of collaboration – synchronous and asynchronous;
3. Required collaboration level – one person solves the problem; several persons solve the situation in an asynchronous coordinated mode; and several persons solve the situation in a synchronous collaborative mode;
4. External monitoring requirements – there is either enough information to achieve the best solution or additional information must be collected from the environment;
5. Tools to determine the best solution – the solution does not require external decision aids, or there is a need of advanced support to achieve the best solution.

This information is associated to every exception raised. It must be emphasized that, likewise the information necessary to classify the situation, these values may change over time as more information about the exception is obtained. An exception history is kept in the system to be consulted by the involved users.

The *objective of the intervention* is further divided into [3; 10; 15; 27; 31]:

- Abort – further divided in: hard, compensate some tasks, and compensate all tasks;
- Decrease completion time to meet deadline;
- Recover from a system failure condition and replace the system in automatic mode;
- Recover from a task failure and place the system back in automatic mode;
- Recover to achieve the lowest penalty possible, i.e., the exception already impacted negatively on the organizational goals and the objective is to minimize the impact;
- Jump forward to a task in the work model;
- Repeat a previous task that was not executed in the desired way;
- Jump backwards in the work model and compensate some already executed tasks;
- Delay this task. This objective can be useful to release some resources necessary to increase the execution time of another process/instance;
- React to environmental changes. This normally requires a process change.

This classification affords linking the recovery process with a specific set of recovery tasks available at the system level. The *required type of collaboration* expresses how the collaboration support component will interconnect the persons involved in the recovery process. We differentiate between two types of collaboration: synchronous and asynchronous. In synchronous collaboration all of the persons involved intervene at the same time, while in asynchronous collaboration the persons involved are not engaged in the process at the same time.

Concerning the *required collaboration level*, one as to be aware of concurrent changes made to work models. When ad hoc changes are applied in an asynchronous coordinated mode, every change is seen as an independent change and the resulting work model results from the composition of previous changes. Therefore, the structural and dynamic checks are made on the instance with respect to this new model. However, in the case of concurrent ad hoc changes carried in an asynchronous collaborative mode, the work of Rinderle [28] must be taken into consideration because actions carried out by different users without any agreement – asynchronously – may conflict (if they overlap on the same part of the model).

External monitoring requirements specify if environmental information is necessary to resolve the exception. The need to collect information within the system has already been identified by Basil et al [4]. In this approach we suggest that the recovery process may as well require collecting environmental information, from outside the system, e.g. generate an interface to display traffic information because a truck is stuck on a traffic jam.

The item *tools to calculate the best solution* identifies any additional tools necessary to calculate the best recovery solution. This affords linking the framework with external tools supporting the decision process.

5. Relationships Between Diagnosis and Recovery

Some relationships can be established empirically between the diagnosis of the situation and recovery strategies. Although some field trials should be carried out to validate the relations they seem very intuitive and easy to explain. These relations can be used as information to feed a decision support system that helps users on the selection of the most appropriate strategy given a concrete scenario.

We start by identifying the dimensions of the recovery strategy that do not depend on the classification; and the dimensions of the classification that do not have a clear impact on the recovery strategy. Then, the remaining relationships and respective consequences are explored.

On the side of recovery strategies, the *objective of the intervention* is determined by the external environment and does not depend of any characterization of the situation. It is related to the organizational goals regarding a particular event.

The dimensions *scope*, *detection*, and *event type* do not have any direct impact on the recovery strategy. *Detection* is important to know how the event was identified. Since similar events can be automatic or manually detected, this is of minor importance in determining the recovery strategy. The *scope* dimension determines the number of instances affected by the situation but does not affect the recovery strategy. The same strategy can be applied to all instances, or different strategies may be

applied to different instances. Finally, the *event type* dimension does not have a clear relationship to the recovery strategy. For instance, a timeout does not imply that the *reaction time* should be quick. However, the *reaction time* dimension can be used to increase the context awareness of a particular timed event. For instance, a timeout may be classified as critical in some situations and not critical in others.

Table 1 summarizes the identified relationships. The rows refer to diagnosis and the columns to recovery. The table shows two types of relationships: the first letter is the relation between the diagnosis and the need to adopt a particular recovery strategy (if the impact is high then there is a strong impact from the diagnosis row on the necessity to use the recovery strategy in the column); and the second letter shows the relation between the diagnosis and the particular type of recovery strategy within the class (if the impact is high there is a strong relation between the diagnosis row and the type of recovery strategy within the column). This means that in a particular situation, even though the diagnosis might not indicate the necessity to use a particular recovery strategy, if the users decide to use it then the chosen recovery strategy might depend on the diagnosis. E.g., the time dimension on the diagnosis does not affect the decision to use a collaboration type (L on the first letter), but if the users adopt a collaboration type then the time dimension has an impact on the type of collaboration type to choose (H on the second letter).

Table 1. Relation between event classification and handling strategies

	Collaboration type	Collaboration level	Ext. monitoring	Tools to best solution
Time	L/H	L/H	L/H	L/H
Goals impact	H/L	L/L	M/H	M/H
Organizational impact	H/L	H/L	L/L	L/L
Difference to organizational rules	H/L	L/L	M/H	M/H
Complexity	L/L	L/L	M/H	H/H

Legend – L – low; M – medium; H – high

As the time associated with the exception is usually an independent factor, we start by discussing time relations. Also, it is important to note that time restrictions have a strong impact on the way people deal with problems. We have defined two dimensions related with time: *reaction time* and *time frame to achieve solution*. The former is important to specify how the person responsible should be informed about an exception. Then, upon starting the diagnosis phase, that person can define the *time frame to achieve solution* in a different way than reaction time: e.g., some contention action was implemented but the final solution can be implemented in a more relaxed time frame. Therefore, once the parameter *time frame to achieve solution* is defined, it will have a stronger effect on the decision process than the *reaction time*. The *time* row in the table reflects this effect and is obtained from these two dimensions.

One should not expect any impact from the *time* dimension on the usage of any *collaboration type*, i.e., for any value of *time* nothing can be concluded about collaboration among users (L on the first letter). However, if the *time* is *quick* and the user wants to use collaboration the synchronous type should be the choice. On the other hand, if the *time* is not *quick*, one can expect that an asynchronous *collaboration type* may be the choice (H on the second letter). This shows low impact from the *time*

dimension on whether any *collaboration type* should be used, but a strong relationship between the *time* dimension and the *collaboration type* to use. The relation is therefore "L/H".

The relationship between *time* and *cooperation level* is similar, since *time* does not affect the usage of any *collaboration level*, but if one is to be used then asynchronous *cooperation level* should be the choice on situations that require fast responses as autonomous actors react faster. Synchronicity increases the delay on recovery actions.

Similar relationships are found between *time* and *monitoring*, and between *time* and *tools*. The need for monitoring environmental information and using external tools to calculate the best solution depends on a particular case and not on time; but if they are needed the particular ones to choose will be restricted by the time factor.

The *organizational goals* dimension has implications on the *required collaboration type*, since events with high impact should involve the user(s) responsible for the task(s) and their supervisors, but the type of collaboration is not imposed. The *required collaboration level* is not affected in any sense by the *organizational impact* as there is not an indication to use any of the identified collaboration levels due to the type of organizational impact. Even further, the collaboration level to choose is not influenced by the goals impact. There are situations with high (low) organizational goals impact that can be solved with only one user implementing recovery actions and others where more than one person is necessary.

Regarding *monitoring* and *tools*, even though the necessity to use them depends on the particular context, the usage of these mechanisms should deserve more attention on situations with high impact on the organizational goals. The value in the table M/H reflects these considerations where the M is used to signal that the concrete scenario has a higher relation, but the impact of the organizational goals dimension should also be taken into consideration. On the other hand, if the impact on the organization goals is high special care should be placed on the monitoring actions and on the tools to use.

The *organizational impact* has a strong relationship with the *collaboration level* and *type*. However, there is a small relationship with the type within these dimensions. The relationship with *monitoring requirements* and *tools* is also low.

On the *difference to organizational rules* dimension, it is expected that more users are involved when there are rules but the right ones can not be found, or when there are no rules in the organization to handle the event. The involvement of more users is important to find the right rules or define new ones. There is a high relation with the selection of a *collaboration type*, but no restriction is imposed. The *monitoring requirements* and the usage of *tools* are expected to increase in situations that differ from normal procedures, and the type of adopted mechanism will also depend on the degree of difference. No relation is established with the *collaboration type*.

Finally, for situations where it is possible to use a tool to calculate the best solution and the *complexity* is high, the primer relevance is made on the *external monitoring requirements*. As in previous situations, it is expected that *external monitoring requirements* are mainly influenced by the concrete situation, so we expect a medium relation. Nevertheless, if one is to be chosen, special care must be taken about the right one. In these situations, as it is easily justifiable, there is high relation between usage and type of *tools*. No relation is established with the *collaboration type* and *collaboration level*, as it is expected that the solution, even though complex, can be calculated by only one actor.

6. Implementation

Figure 3 represents the proposed exception handling work model, an extended version of our previous work [25], where two new branches were inserted addressing external monitoring actions and collaboration mechanisms; and some minor changes were done to the *collaboration* component. Further details regarding *association of instances* and *edit exception classification* can be consulted in the cited paper.

In the present work we are not concerned with the specific implementation details, in particular about the WfMS engine or model language used. Our main focus is how exception handling is supported by the proposed framework.

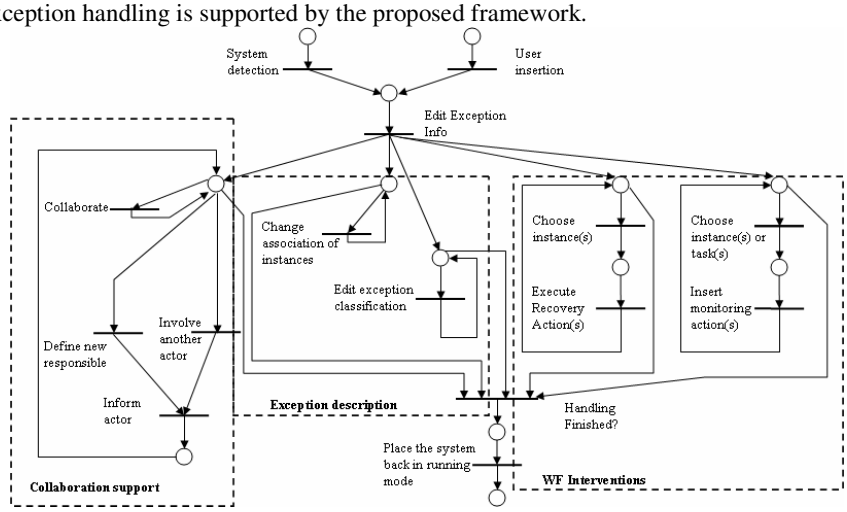

Fig. 3. Exception handling workflow

The *collaboration support* component supports users specifically collaborating within the scope of an exceptional event. The tasks implemented by the component (see figure 3) enable the definition of a new responsible, involve more actors, and implement the collaboration mechanism. The *collaborate* task in the model can be synchronous or asynchronous where at any instant the users can choose the type to use. When asynchronous collaboration is being used any involved actor can send a message to any or all of the colleagues using a developed interface. The company email system is used to inform the user that s/he should check the workflow system. Synchronous collaboration support depends on the application domain and environment as it can be implemented by a phone conversation, chat over a computer or even face-to-face conversation. In both cases the exchanged information is stored using the *exception history* component. If it is not possible to store the conversation the users should insert the conclusions and any special comment. Further developments of collaborative components will be subject to future research.

The *wf interventions* component is implemented using two branches: implement recovery actions; and insert monitoring tasks. Recovery actions are a set of atomic interventions that can be carried out on the specific workflow engine, e.g., suspend an instance or insert a task in the original model.

The *monitoring* branch affords users to insert monitoring tasks that store exception relevant information in *exception history*. Since this information is chronologically stored, the user may monitor the system evolution. Even further, if external environmental information or tools are available, the user may store links, e.g., a link to traffic cameras available through the internet, which may be used to diagnose the situation on a truck that is stuck on a traffic jam.

The detection of a new exception situation is represented in the figure by the first two parallel branches on the top of the figure representing system and manual detection. These tasks will insert all the mandatory information. As mentioned before, a responsible person must always be associated to the exception and will execute the next task "Edit exception info" where the most important information related to the exception is inserted. This task assures that the responsible person is informed on the situation and initiates the exception handling procedure.

The person responsible may then execute any of the actions specified in the six branches of the exception handling workflow. Let us assume that a user decides to involve three more actors in the exceptional event. Then, using the collaboration component (email or chat) s/he informs the other actors that there is an exception to be resolved. The diagnosis phase proceeds using the collaboration component, so the other actors share their views of the present situation. Finally, they decide to insert two monitoring actions in the work model, and two of them will be responsible for the follow up. Once any special event regarding these monitoring actions is triggered, the group is informed and the recovery action may proceed using the execute recovery action branch. The process is repeated until the exceptional situation is overcome.

7. Example

During an operation, the patient's heart monitoring machine breaks down. As soon as the machine breaks, one nurse connects the patient to an available, but less reliable machine, and continues monitoring the patient's data. Meanwhile, she manually instantiates the exception recovery workflow and assigns her name as the person responsible. The time frame to reach a solution is set as *quick*, a brief description of the event is inserted and she joins two other persons to the exception handling procedure: one person will try to fix the machine (maintenance operator) and a technical assistant will try to find another compatible machine for backup. As the time frame is set as quick, both departments will be informed by a flashing light and a buzz sound on their coordination room. The respective department coordinators will look at their computers and find this exceptional situation to handle. They will initiate their own recovery tasks and assign them to an employee in their department (using a form requiring the name, the urgency, and a manually inserted description of the task).

As the maintenance employee will go inside the operations room, face-to-face collaboration with the nurse is assured. However, if the situation in the operations room changes, the technical assistant should be informed. A collaborative reporting task is generated where both the technical assistant coordinator and the nurse can read and write information. Both of them write any status changes. If the maintenance operator fixes the machine, the nurse writes this information and the technical coordinator informs the employee. If, on the other hand, a compatible machine is

found, the technical assistant informs his coordinator, who writes down the predicted available time. If the solution is approved by the nurse the maintenance operator will start to prepare the replacement. When the technical assistant arrives with the machine, the maintenance operator replaces it and the exception is closed.

However, one can imagine a dramatic case where all the machines are allocated to patients and someone responsible has to decide whether they can be removed or not. The technical assistant coordinator informs the nurse and they decide to involve the doctor responsible for the area. They insert a new task in the work model supplying information to the doctor with a high priority: again, a signaling system should be available so the doctor is quickly informed. The doctor analyses the task description and decides, given the patients' situations, if she has enough information to make a decision. If not, she initiates a chat session with the nurse (new task). Let us assume the doctor decides to remove a machine from one of her patients but only after a new nurse is assigned to monitor this patient. She therefore affects another instance to the exception (the instance associated to this new patient) and inserts three new tasks: find a new nurse, remove the machine, and replace the machine once the operation is finished (note that the first and second tasks are in parallel to the original operation sequence, while the last is placed after the operation is finished). The first is assigned to the nurse coordinator and the other two to the technical assistant coordinator. Once the new nurse is monitoring the patient, the first task is completed and the technical assistant can take the machine to the operations room. After the operation is finished, the task to move the machine back to the patient is ready to be executed. Only after the machine is replaced in its original position the exception is completed.

8. Conclusions

The major concern addressed by our framework is the support to unexpected exceptions, defined as situations that can not be handled in an automatic way because the system does not have information about them, nor can infer such information from previous analogous situations. Under these circumstances, collaborative user involvement is crucial to determine the most appropriate solution.

Our analysis highlighted a fundamental system requirement: maintain task execution under model guidance during normal operation and change to unstructured behavior when an unexpected exception occurs, supporting users giving the control back to model guidance after the exceptional situation is overcome.

We developed an exception handling process to support this behavior, comprising collaborative diagnosis, recovery and monitoring tasks. Furthermore, we analyzed in detail the characteristics and relationships between the diagnosis and recovery tasks. The diagnosis task is based on a new classification of unexpected exceptions proposed in this paper. Several dimensions characterizing the recovery tasks, as well as relationships with the classification of unexpected exceptions are proposed as well. The resulting framework helps users collaborating to devise appropriate exception handling strategies for unexpected situations.

9. Bibliography

1. Abbott, K.R., and Sarin, S.K., 1994. Experiences with workflow management: issues for the next generation. Proc. of the 1994 ACM Conference on CSCW. Chapel Hill, North Carolina, United States, pp. 113-120.
2. Agostini, A., and De Michelis, G., 2000. A light workflow management system using simple process models. CSCW, 9(3): 335-363.
3. Agostini, A., De Michelis, G., and Loregian, M., 2003. Undo in Workflow Management Systems. BPM 2003. Springer-Verlag, Netherlands, pp. 321-335.
4. Bassil, S., Rinderle, S., Keller, R., Kropf, P., and Reichert, M., 2005. Preserving the Context of Interrupted Business Process Activities. 7th ICEIS 2005. USA.
5. Bernstein, A., 2000. How can cooperative work tools support dynamic group process? bridging the specificity frontier. CSCW '00: Proceedings of the 2000 ACM Conference on CSCW. ACM Press, Philadelphia, pp. 279-288.
6. Casati, F., 1998. Models, Semantics, and Formal Methods for the Design of Workflows and their Exceptions. PhD Thesis, Politecnico di Milano.
7. Casati, F., Ceri, S., Pernici, B., and Pozzi, G., 1996. Workflow Evolution. Data and Knowledge Engineering, 24(3): 211-238.
8. Casati, F., and Pozzi, G., 1999. Modelling exceptional behaviors in commercial workflow management systems. Proc. IFCIS, International Conference on CoopIS, CoopIS '99. IEEE International, Edinburgh, UK, pp. 127-138.
9. Chiu, D.K., 2000. Exception Handling in an Object-oriented Workflow Management System. PhD Thesis, Hong Kong Univ. of Science and Technology.
10. Chiu, D.K., Li, Q., and Karlapalem, K., 2001. WEB Interface-Driven Cooperative Exception Handling in ADOME Workflow Management System. Information Systems, 26(2): 93-120. Elsevier Publishers.
11. Dayal, U., Hsu, M., and Ladin, R., 1990. Organizing Long-Running Activities with Triggers and Transactions. SIGMOD'90. NJ, USA.
12. Dellarocas, C., and Klein, M., 1998. A Knowledge-based approach for handling exceptions in business processes. WITS'98. Helsinki, Finland.
13. Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., and Zbyslaw, A., 1996. Freeflow: mediating between representation and action in workflow systems. Proc. of the 1996 ACM Conference on CSCW. ACM Press, New York.
14. Eder, J., and Liebhart, W., 1995. The Workflow Activity Model WAMO. Int. Conf. on Cooperative Information Systems. Vienna, Austria.
15. Eder, J., and Liebhart, W., 1996. Workflow Recovery. 1st IFCIS Intl. Conf. on Cooperative Information Systems (CoopIS'96). IEEE, Belgium, pp. 124 - 134.
16. Ellis, C., and Keddara, K., 2000. A Workflow Change is a Workflow. In: W.D. van der Aalst, J. Oberweis (Editor), Business Process Management: Models, Techniques, and Empirical Studies. Springer-Verlag, pp. 201-217.
17. Ellis, C., Keddara, K., and Rozenberg, G., 1995. Dynamic change within workflow systems. Organizational Computing Systems., Milpitas, CA, USA.
18. Ellis, C., and Nutt, G.J., 1993. Modeling and enactement of workflow systems. Application and Theory of Petri Nets. Springer-Verlag, Illinois, USA, pp. 1-16.
19. Faustmann, G., 2000. Configuration for Adaptation - A Human-centered Approach to Flexible Workflow Enactment. CSCW, 9(3): 413-434.

20. Han, Y., Sheth, A.P., and Bussler, C., 1998. A Taxonomy of Adaptive Workflow Management. Conf. on CSCW - Workshop - Towards Adaptive Workflow Systems. Seattle, WA, USA.
21. Hayes, N., 2000. Work-arounds and Boundary Crossing in a High Tech Optronics Company: The Role of Co-operative Workflow Technologies. CSCW, 9(3): 435-455.
22. Hwang, S.Y., Ho, S.F., and Tang, J., 1999. Mining Exception Instances to Facilitate Workflow Exception Handling. 6th Int. Conf. on Database Systems for Advanced Applications. Hsinchu, Taiwan.
23. Jorgensen, H.D., 2001. Interaction as Framework for Flexible Workflow Modelling. Group '01. ACM Press, Boulder, Colorado, USA.
24. 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.
25. Mourão, H.R., and Antunes, P., 2004. Exception Handling Through a Workflow. CoopIS 2004. Springer-Verlag, Agia Napa, Cyprus.
26. Reichert, M., and Dadam, P., 1998. ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems, 10(2): 93-129.
27. Reichert, M., Dadam, P., and Bauer, T., 2003. Dealing with Forward and Backward Jumps in Workflow Management Systems. Software and Systems Modeling, 2(1): 37-58. Springer-Verlag.
28. Rinderle, S., 2004. Schema Evolution in Process Management Systems. PhD Thesis, University of Ulm.
29. Rinderle, S., Reichert, M., and Dadam, P., 2003. Evaluation of Correctness Criteria for Dynamic Workflow Changes. BPM '03. Netherlands, pp. 41-57.
30. Saastamoinen, H., 1995. On the Handling of Exceptions in Information Systems. PhD Thesis, University of Jyväskylä.
31. Sadiq, S.W., 2000. On Capturing Exceptions in Workflow Process Models. Proc. of the 4th Int. Conference on Business Information Systems. Poznan, Poland.
32. Schmidt, K., 1997. Of maps and scripts - the status of formal constructs in cooperative work. GROUP '97: Proc. of the Int. ACM SIGGROUP Conf. on Supporting Group Work: The Integration Challenge. United States, pp. 138-147.
33. Sheth, A.P., Georgakopoulos, D., Joosten, S.M., et al, 1996. Report from the NSF workshop on workflow and process automation in information systems. ACM SIGMOD Record, 25(4): 55-67. ACM Press.
34. Strong, D.M., and Miller, S.M., 1995. Exceptions and Exception Handling in Computerized Information Systems. ACM Trans. on Information Systems, 13(2).
35. Suchman, L.A., 1987. Plans and Situated Actions. MIT Press.
36. van der Aalst, W., and Basten, T., 2002. Inheritance of workflows: an approach to tackling problems related to change. Theoretical Computer Science, 270(1).
37. van der Aalst, W., Basten, T., Verbeek, H., Verkoulen, P., and Voorhoeve, M., 1999. Adaptive Workflow: On the interplay between flexibility and support. Proceedings of the 1st ICEIS. Setúbal, Portugal, pp. 353-360.
38. Weber, B., Wild, W., and Breu, R., 2004. CBRFlow: Enabling Adaptive Workflow Management through Conversational Case-Based Reasoning. European Cof. on Case-Based Reasoning (ECCBR'04). Madrid, Spain.