Mourão, H. and P. Antunes (2007) Supporting Effective Unexpected Exceptions Handling in Workflow Management Systems. Proceedings of the 22nd Annual ACM Symposium on Applied Computing, Special Track on Organizational Engineering, Seoul, Korea. ACM Press, pp. 1242-1249.

Supporting Effective Unexpected Exceptions Handling in Workflow Management Systems

Hernâni Mourão
Escola Sup. de Ciências Empresariais and LASIGE
Campus do IPS – Estefanilha
2914-503 Setúbal, Portugal
+351265709431
hmourao@esce.ips.pt

Pedro Antunes
Faculdade de Ciências and LASIGE
Faculdade de Ciências/UL, Campo Grande – Edifício C6
1749-016 Lisboa, Portugal
+351217500605
paa@di.fc.ul.pt

ABSTRACT
This paper proposes a novel architectural framework handling effective unexpected exceptions in workflow management systems (WfMS). Effective unexpected exceptions are events for which the organizations lack handling strategies. Unstructured human interventions are necessary to overcome these situations, but clash with the type of model control currently exercised by WfMS. The proposed framework uses the notion of map guidance to orchestrate these human interventions. Map guidance empowers users with contextual information about the WfMS and environment, enables the interruption of model control on the affected instances, supports collaborative exception handling and facilitates regaining model control after the exception has been resolved. The framework implementation in the Open Symphony open source platform is also described.

Categories and Subject Descriptors
H.4.1 [**Information Systems Applications**]: Office Automation – *workflow management, groupware, human factors.*

General Terms
Design, Reliability, and Human Factors.

Keywords
workflow management systems; unexpected exceptions; collaboration support; and unstructured organizational activities.

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 [28]. However, the majority of the available organizational information systems tend to fall close to both sides of the spectrum boundaries, thus leaving a significant gap in between. WfMS, based on work models, play the role of scripts falling close to the highly structured boundary [27]. Closer to the other end of the spectrum limits, Suchman [29] proposes the notion of maps, which position and guide actors in a space of available actions, providing the context awareness necessary to make decisions but avoiding the normative trait. To support the continuum of organizational needs, WfMS should cope with the whole spectrum of structured and unstructured activities integrating both procedural and non-procedural work [12]. In the WfMS community, nonprocedural work has been designated "exception handling."

Eder and Liebhart's [9] classification of expected and unexpected exceptions has been widely used, 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 focused on the exceptions that fall close to the unexpected limits of the spectrum, meaning that nothing similar has happened before from which the organization can draw any prearranged behavior. This type of exceptions requires human intervention and an innovative posture from the organization. As no model is available, human reactions should be map guided, according to Suchman's definition. From now on we will refer to this type of exceptions as *effective unexpected exceptions*, or unexpected exceptions when no distinction is necessary.

Our framework is supported on the statement of completeness requirement expressed in [20]. In summary, this 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 paper is structured in the following way. We start by describing the two examples used throughout the paper to motivate and illustrate our approach. Then, in section two, we revise the related work and to identify the scope of the framework. Section three describes the framework, addressing the above mentioned dichotomy: maintain model-based work whenever possible and change to map guidance whenever necessary. We also classify exception handling strategies. In section four we describe the framework's implementation in the Open Symphony platform. Finally, the section five finishes with the conclusions.

1.1 Motivating Examples
An effective unexpected exception must always be brought from real life with proper documentation about the adopted strategies.

Considering this limitation, we use two motivating examples to illustrate our solution: 1) a media report on the 9/11 catastrophic event, as experienced by the USA's air traffic control center; and 2) a non-catastrophic but also unexpected event, where a WfMS must handle for the first time a client that went bankrupt. While this second situation is much less inspiring that the first one, it was indeed experienced by us during the implementation of a space rental management system for a Port Authority.

We fundamentally selected the 9/11 event because very rich information about the adopted exception handling procedures is available to the public [22][*]. The overwhelming impact in society and strong political implications of this unique event were not within the selection criteria and are out of the scope of this research. On the other hand, as discussed in the previous section, an effective unexpected exception is an event for which the organization has no prior knowledge about the resolution. Therefore, this is a good example to motivate the discussion on how WfMS users react to this type of situations.

Considering regular air traffic control, every plane is a process instance and every route is modeled since the plain first checks in the air traffic control on the departing airport and until it checks out on the arriving airport. For instance, AA flight 11 route on 9/11 started at Boston and its model considered driving it to Los Angeles. At approximately 8:15 AM on that day, the air traffic control center in Boston stopped receiving feedback from the airplane pilots and lost the transponder signal. Controllers also reported hearing a man with a strange accent in the cockpit. This combination of events originated an exception. Along with the description of our exception handling solution, facts from this real event will be used to exemplify how the proposed solution could be used to support exception handling.

One of the key decisions taken during this exceptional situation was to land every plane that was flying in the USA and Canada air spaces. According to FAA officials, they "[…] decided not to write a new set of procedures for clearing the skies. They started to but scrapped the idea. They concluded that the FAA was better off relying on the judgment of its controllers and managers." From our perspective, this means that under such extreme conditions procedural control was considered worse than giving people access to the relevant updated information and letting them decide the best reactions to the concrete situation, i.e., map guidance was clearly favored against model guidance.

It is also important to notice that air traffic controllers tried to do whatever they could to overcome the situation. They used any available means to fulfill their goals and established their goals on the fly as they were collecting information about the situation. Furthermore and most important, the plan to overcome the situation was not defined for every control center. According to the available airports and number of planes they had to land, controllers implemented different local strategies. This situation highlights the completeness requirement stated in section 1.

[*] the report was issued by USA Today based on interviews to more than 100 people involved in key decisions and data collected from other sources, such as FAA radar, air traffic control databases, and a special software to analyze plane rerouting.

Finally, we emphasize that although model guidance could not be adopted, map guidance was apparently considered beneficial: the FAA command center, after the second plane crashed, decided to writing on a white board information regarding all planes suspected to be hijacked. This situation also stresses the role of monitoring information and external tools in map guidance.

Another important implication to our research can be drawn out from this quote from USA Today: "landing nearly 4,500 planes was a massive undertaking and a historic achievement. It required intense cooperation, swift decision-making and the unflinching work of thousands of people. Across the nation, controllers searched for alternate airports to land large jets." The mentions to intensive cooperation and swift decision making are crucial to our exception handling approach.

The second example was chosen because it is related with a WfMS that we developed for a Port Authority and a real exceptional event that we had the opportunity to follow. Several data related with this event was collected and the system users were interviewed to identify the adopted handling procedures and their relationships with the WfMS. This second example is used in the paper to discuss the feasibility of our framework.

The businesses processes modeled refer to the activity that manages space rentals within the Port Authority jurisdiction. They are administrative processes and involve 10 persons within the organization. Every month the system must generate invoices for every occupied space and follow up payment. A list of debts and free/occupied zones must be generated at any moment. Client related information is also managed by the system.

2. RELATED WORK AND SCOPE OF THE FRAMEWORK

Various approaches to increase WfMS flexibility during runtime can be found in the literature: 1) special modeling constructs to deal with unexpected exceptions [2,6]; 2) apply model changes to running instances [3,11,24,30]; and 3) using interactive enactment or constraint based workflow [8,13,16]. We identify two parallel research streams [14]: metamodel and open point. Metamodel approaches take into specify, implicit or explicitly, metamodelling constructs to deal with flexibility assuring system correctness, while open-point approaches rely on the users to assure that no inconsistencies are inserted in the system.

Metamodel approaches to deal with expected exceptions rely on special modeling constructs that are invoked whenever a predefined exceptional situation is detected [2,5,6,9]. Several techniques, such as exception mining [5,15], case base reasoning [17,31], and knowledge bases [7] have been proposed to expand the system flexibility handling exceptions. If we consider a continuum from expected to unexpected exceptions, all these systems handle events falling close to the expected limit.

Chiu developed a system to handle expected and unexpected exceptions [5]. The user interface to handle unexpected exceptions allows one user to choose a complete new path. However in our proposed system collaboration among different users and map guidance is a critical issue in effective unexpected exception handling.

Regarding the metamodel approaches and unexpected exceptions, we find several solutions based on dynamic changes and ad hoc

interventions offering correctness criteria to keep system consistency [3,11,24,30].

Regarding the open point approaches in more detail, we find interactive enactment [16], flexible enactment [13], and constraint base modeling [8]. These approaches use incompletely specified models, allowing users to interactively adapt them, e.g., inserting tasks. This increases the users' freedom 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 [14], considering that no dynamic or structural checks are made.

Like Agostini and De Michelis [1], 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.

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 metamodel assumptions are used to check system coherence. This enables 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 identifying the necessary actions to bring the system back to a coherent state.

3. SUPPORTING UNSTRUCTURED ACTIVITIES

We propose a framework supporting the unstructured activities necessary to resolve effective unexpected exceptions. In this section, we start by describing how the framework governs the system changes from model control to human-controlled unstructured activities and then back to model control. The framework enables the system to support the whole spectrum of organizational processes mentioned in section 1. In section 3.1. we introduce the four functions of an exception handling process: detection, diagnosis, recovery, and monitoring. We analyze the main activities carried out on each one of these four functions and their inter-relations as the handling procedure evolves. Section 3.2. analyzes the diagnosis and handling strategies.

On the occurrence of a basic failure, application failure or expected exception we will assume that modeling assumptions and runtime features of the WfMS handle the situation and the system is kept under model control.

On the contrary, when an unexpected exception is detected, the system supports unstructured activities carried outside the consistency boundaries, i.e. the system is beyond model control. When the exception handling is accomplished, the users may decide whether the process instance should be placed under model control, continue outside model control, or be aborted. If model control is the choice, the system will then analyze model inconsistencies, and either redeem model control or notify the users about existing conflicts and continue supporting unstructured activities. The model consistency analysis is accomplished instance by instance.

On the 9/11 motivating example, after the order to land all planes was issued, the Memphis controllers scrapped normal air traffic procedures and decided that every controller should follow their assigned planes until landing. Usually the planes are transferred from a proximity operator to an airport operator when they get close to the airport. But since the number of planes to land was very high, they decided to eliminate these transfers, reducing the synchronization and information overloads. Suddenly, the air traffic controllers started working under a completely new choreography. As reported, all over the country the controllers had to find out the best solution to overcome the problems they faced in their areas. During this period, the air traffic control system in the US was operating with unstructured activities.

When the situation finally got under control, i.e., officials were convinced that no hijacked planes were in the air, they smoothly started rescheduling and allowing commercial airplanes to take off to their destinations. The system therefore was step by step being lead to model control.

3.1 Exception handling functions

We distinguish four functions in exception handling: 1) exception detection; 2) situation diagnosis; 3) exception recovery; and 4) monitoring actions.

The majority of authors identify the first three [7,26]. However, as we already mentioned before and explain in detail below, we believe that monitoring actions play a key role in effective unexpected exception handling.

Exception detection has been extensively studied in previous works [2,5,6,19,26]. Detection can be manual or automatic. A detailed description of the automatic detection techniques is out of the scope of this paper, since we are focused on the user's perspective. We assume that an exception detection component must be tightly integrated with the workflow engine. In section 4.2, we discuss the integration of the detection component with the other exception handling functions.

We will rather focus on the other three functions. In our framework, we advocate an intertwined play between diagnosis, recovery and monitoring until the exception is resolved [18]. 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 and being processed by humans. As an example, the already mentioned white board displaying information about the planes suspected to be hijacked was very important to manage the situation and decide the next steps.

These activities, categorized in the framework as monitoring actions, are necessary to control the progress of the whole exception handling process. They allow users to collect up-to-date information about the exception. Considering again the open

nature of the framework, these monitoring actions may also bring environmental information to the system.

After diagnosis, users may 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 also necessary to bring environmental information to the system. This issue will be addressed later in more detail.

3.2 Diagnosis and handling strategies

A good common understanding of the exceptional situation is crucial to take the right decisions. Providing involved actors with action contexts [33] supports knowledge production and consumption, augmenting the quality of the decision making process. In this section we start by discussing the several dimensions necessary to diagnose the exceptional situation (parameters of the context), and then we proceed with the handling strategies.

3.2.1 Diagnosis

The diagnosis is mostly dependent on a detailed assessment of the exceptional event. Using previous classifications [4,25] and some new added characteristics, we classify exceptional situations using the orthogonal dimensions:

(i) *Scope* – **process specific** when only a set of instances is affected; or **cross specific** when various sets are affected. At least one instance must always be associated;

(ii) *Detection* – **automatic** or **manually** detected exceptions;

(iii) *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. The assessment of the event type is mandatory, because it impacts the handling phase;

(iv) *Organizational impact* – **employee**, when only a limited number of employees in the same department are affected; **group**, when more than one department is affected; and **organizational**, when the overall organization is affected. A responsible must always be associated to the exception;

(v) *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;

(vi) *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. The solution is related to the semantics associated with the event and not with the handling procedure;

(vii) *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;

(viii) *Time frame to achieve solution* – **quick**, when the situation is expected to be resolved in few working units; **relaxed**, when the time frame is more relaxed, although being a parameter to be taken into consideration; and **long** when time is not a critical issue.

Only the *scope*, *organizational impact*, *event type*, and re*action time* dimensions must be set by the detection process. The other dimensions may by set or not by the users, according to their perceptions of the situation. These dimensions may be redefined by users whenever more information is collected, and the old values are always preserved in a chronological record.

Bringing back to the discussion our 9/11 motivating example, and considering the first exceptional event, the detection was manual and occurred when the controller realized that a plane stopped answering calls and the transponder signal disappeared from the radar. This "process-specific" situation affected only one instance. The *time frame to achieve* solution was "relaxed", since the controller had to follow the event realizing if it was a serious trouble or a temporary malfunction. Some other diagnosis information would include: it was a *workflow event type*; the *organizational impact* affected only "one employee" and the *difference to the organizational rules* was an "expected exception", where the controller knows the right procedure to apply. All other dimensions are undefined.

When the controller heard a strange accent in the cockpit saying "we have some planes, just stay quiet and you will be OK," the situation changed and the exceptional event was propagated to the control center in Herndon. This becomes a situation to be followed by the central office with high priority: the *organizational impact* changes to include the national operations manager and the *time frame to achieve solution* is maintained in "relax" mode, because hijacked planes usually follow some course to an airport and thus do not demand fast recovery.

When the second hijacked plane hit the south tower, the diagnosis changed again. The *time frame to achieve solution* had to change to "quick", the *organizational impact* now affected the whole air traffic control organization, the *complexity of the solution* changed to "high" and the *difference to organizational rules* corresponded to a "true exception." As a consequence of the new diagnostic, the national operations manager started wondering how many and what planes were in the hands of the hijackers and collecting more information, e.g. to identify the affected instances.

3.2.2 Exception handling strategies

We identify the following dimensions to classify exception handling strategies:

(i) *Recovery actions* – further division presented below;

(ii) *Communication type* – **synchronous** or **asynchronous**. This dimension classifies the way people exchange information;

(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**. It should be emphasized that this dimension is focused on implementing recovery actions;

(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 of **advanced support** to achieve the best solution.

This information is associated to every exception raised in the system. We emphasize that, likewise diagnosis, this information may change over time as more data about the exception is obtained. A chronological record of the selected values is kept in the system to be consulted by the involved users. The *recovery actions* can be [5,10,23,26]: abort, decrease completion time, recover from a system failure, recover from a task failure, jump forward, repeat a task, jump backwards, delay this task, and react to environmental changes.

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 exception handling process. We differentiate two types of communication: synchronous and asynchronous. In synchronous communication all of the persons involved intervene at the same time, while in asynchronous communication the persons involved are not engaged in the process at the same time.

Concerning the *collaboration level*, one has to be aware of concurrent ad hoc changes made to process instances. There are two different situations to be considered: 1) if the interventions are on disjoint zones of the workflow model no special care is needed and users can implement recovery actions in a collaborative mode; 2) if they are working on overlapping zones of the model they must have a tight synchronization and clear understanding of the interferences each one has on the other and they should implement the actions in a coordinated mode [24].

The *external monitoring* dimension specifies if environmental information is necessary to resolve the exception. In our framework we suggest that not only diagnosis but recovery as well may require referencing external information.

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

4. IMPLEMENTATION

In this section we describe the framework implementation. We start with some relevant details about the system platform. Then, we identify the exception handling components and we finish illustrating system usage on the Port Authority example.

4.1 System platform

The adopted WfMS is the OSWorkflow offered by the Open Symphony (OS) open source platform [21]. The OS platform provides very basic workflow functionality, consisting of a workflow engine (OSWorkflow) integrated with generic Web Services support. The workflow clients are built upon the Model View Controller (MVC) architecture.

Furthermore, we rely on the Java SMS project to send mobile messages supporting asynchronous communication. Finally, we also integrate Wildfire Instant Messenger (IM) client. This component supports exception handling strategies involving the collaboration between users.

4.2 Exception handling components

In Figure 1 we identify the exception handling components and interfaces with the OS components. Four exception handling components and two interfaces are identified. The components are: exception description, WF interventions, exception history and collaboration support.

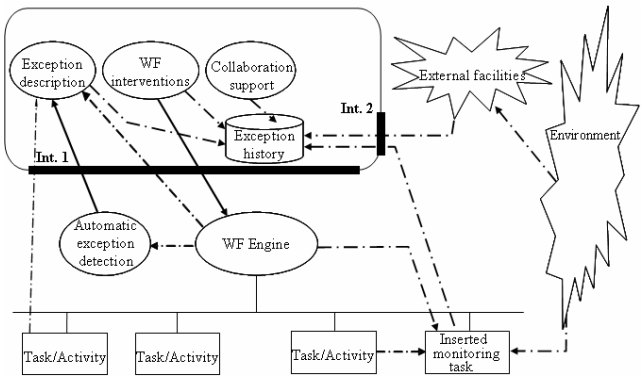

Figure1. Exception handling components and interfaces

The exception description component supports the diagnosis process described in section 3.2.1. The WF interventions component implements the workflow interventions described in section 3.2.2. The collaboration support component implements the communication type and collaboration level mechanisms also described in section 3.2.2. Finally, the exception history component stores all relevant information associated to the exception handling cycles.

Concerning interfaces, the interface 1 links the exception handling components with the OSWorkflow, while interface 2 links these components with the external environment. Interface 1 is used to collect information about the OSWorkflow status, to implement low level recovery actions (launch/suspend tasks, etc), and to automatically detect and signal exceptions.

The interface 2 supports referencing environmental information gathering about the operations carried outside the framework's scope. We differentiate two types of activities carried out in the external context: 1) situation awareness, collaboration and decision making; and 2) recovery actions. The former are related with external communication, coordination, collaboration and decision making tools (e.g., meetings, telephone conversations and operations research techniques). The latter address any external recovery actions necessary to resolve the exception. It is our aim that, for any activity executed outside the framework's scope, some environmental information is inserted in the system for monitoring purposes.

Dashed lines shown in Figure 1 represent information flows whereas uninterrupted lines represent control flows. Automatic exception detection is also represented in the figure, as it involves the workflow engine. Automatic detection is implemented by a specialized component that is highly dependent on the OSWorkflow.

4.3 Detailed functionality of the exception handling components

The functionality of the exception handling components is orchestrated by a workflow model executed by the WF engine[†].

[†] To make this more clear: we use a workflow model to orchestrate exceptions raised by other workflow models.

The model, shown in Figure 2, and the orchestration of the exception handling process is described in the current section.

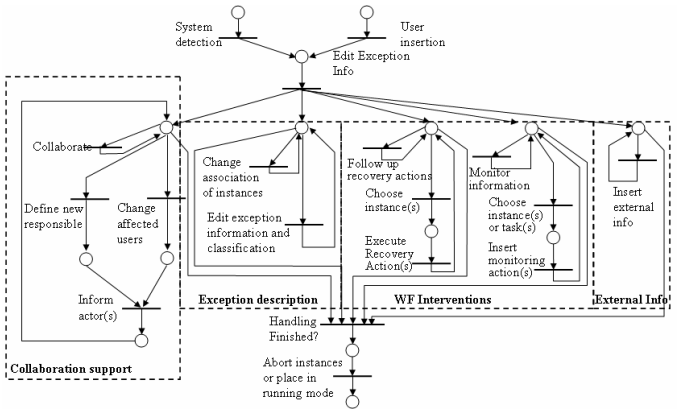

Figure 2. Exception handling components and interfaces

The exception handling process is instantiated either by automatic or manual detection. In both cases one person is always associated to the exception and involved in the handling process. That person is either the one that manually detected the exception or someone involved in the workflow task that automatically generated an exception.

From the users' point of view, the handling process is managed through a Web page, which we designate EHW. Before proceeding with a detailed explanation of the EHW page, some specific OSWorkflow terminology should be introduced. A workflow state is named step in OS. A task in OS is an activity carried out within a step that does not change the workflow state. An action is always associated to a step transition. These definitions are necessary to understand the links between the model shown in Figure 2 and the EHW page displayed in Figures 3 and 5.

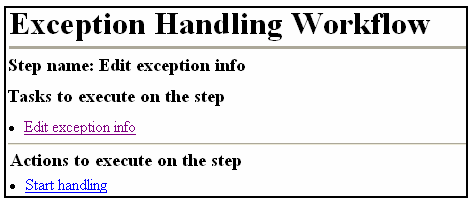

Figure 3. Exception handling workflow page (EHW)

The EHW page reports on the current status and manages the exception handling workflow using the notion of step. In the situation shown in Figure 3, only one step is active: "Edit exception info." If the "Edit exception info" task is executed, the exception information can be edited but no workflow transition will take place.

In Figure 4 we show the details of the "Edit exception info" task, where the user defines the mandatory and optional diagnosis values discussed in section 3.2.1. That person may also define a new responsible and a list of affected users and workflow instances. If a new responsible is assigned, that person is contacted by the collaboration support component on the step transition. This component uses SMS messaging if the time value is quick and an email otherwise.

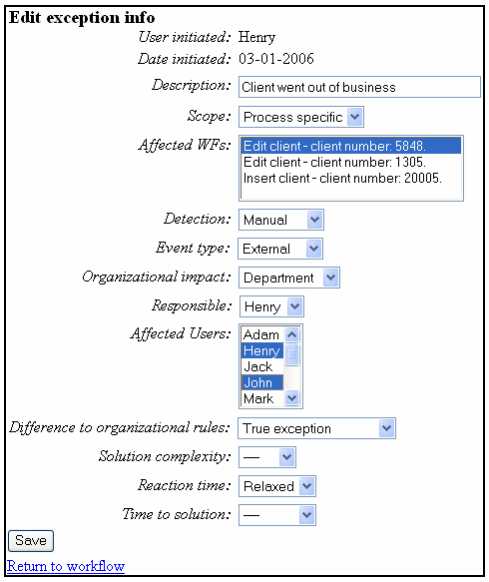

Figure 4. Editing the exception information

Considering again the EHW page, the "Start handling" action initiates the five parallel branches of the exception workflow model. Consequently, the EHW page will look like Figure 5. Observe that several steps are now available, allowing to collaborate with other persons involved, modify the exception description, execute recovery actions, execute monitoring actions or manage external information.

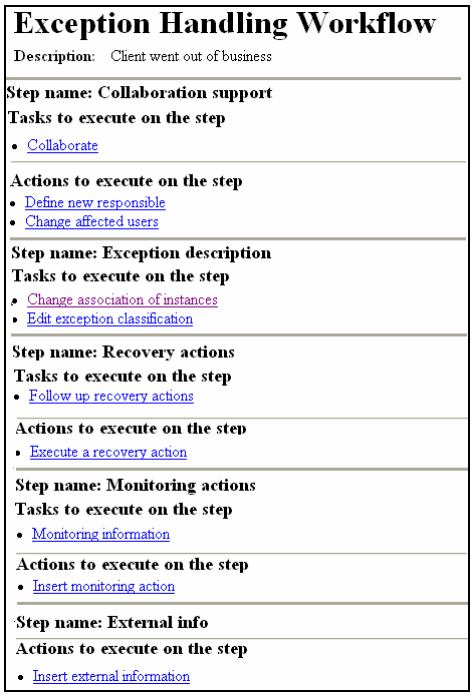

Figure 5. EHW page handling the 5 parallel branches of the exception handling workflow

The "Collaboration support" step offers one task and two actions. The collaborate task can be synchronous or asynchronous, and at any time the users may choose which type to use. When

asynchronous collaboration is selected, the system supports sending email messages between the several persons handling the exceptional event. The generated email messages mixes information provided by the sender with information automatically generated by the collaboration component, which includes at least a link to the EHW page. Concerning the synchronous collaboration, the collaboration component supports IM between the several persons handling the exceptional event and interfaces with the exception history component to preserve the exchanged messages in context.

The "Define new responsible" action allows modifying the person responsible for the exception handling. This action is implemented by a Web page where the user may choose a new responsible by selecting a person from a combo box.

The "Change affected users" action enables the selection of affected users, and is implemented by a Web page similar to the previous one where multiple users can be selected.

Concerning the "Edit exception classification" action in step "Exception description", the Web page utilized to edit the exception classification is similar to the "Edit exception info" page shown in Figure 4 and is not shown. Another functionality is that the user may share alert messages with attached files with the other persons involved. These alert messages may be classified as critical (displayed in red) or important (displayed in blue). Figure 6 illustrates how the alert messages are displayed in the EHW page. If none of these classifications is selected the message is only displayed inside the component. Another web page enables changing the workflow affected instances.

Figure 6. EHW displaying alert messages at the top

Concerning the "Recovery actions" step, the user must first select, among the affected instances, which ones to apply a recovery action. Then, one recovery action may be selected from the list discussed in section 3.2.2. The implementation of these recovery actions requires low-level interventions in the OSWorkflow that will not be described in detail here.

Regarding "Monitoring actions," this step allows users storing relevant external information in the exception history. The user may select among the following information types: application data; workflow relevant data; workflow control data; links to Web resources; and text provided by users. Application data, workflow relevant data, and workflow control data follow the terminology defined by the WFMC [32].

If application data resides on an accessible database, a reference can be inserted in the OSWorkflow configuration file to allow accessing the database. The monitoring action Web page then accesses the database metadata and displays the available tables and fields, so that the user may associate the monitoring action with a database field.

Finally, the "External info" step affords recording into the exception history any external information provided by users.

4.4 Example usage

We will rely on the previously mentioned Port Authority example to illustrate the feasibility of our solution. Assume that Henry is updating the client's information when he is informed that the client has bankrupted. On the Web page to edit client related information there is a link to manually signal a new exception. After selecting this link, the user is prompted with the EHW page shown in Figure 3. From there, the exception classification must be accomplished, as shown in Figure 4. Henry realizes that time is not critical and classifies it as relaxed. He also affects John, his 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 exception 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 are automatically defined by the system.

By following the link shown in Figure 3, 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.

John may then look at the situation in the EHW page and start a collaboration task with Henry. He decides using IM. During the conversation, John realizes that the space occupied by the company is being requested by another company. He also recognizes that the client's debt is 50.000€. John tells Henry to insert this alert in the EHW and then 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.

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

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 with Henry and John to decide on 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 this external recovery action. John will also monitor the evolution of the case in order to decide or not to release the space to another client. 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. They realize the older debt does not allow them to start a law suit; however they decide that John should continuing monitoring this client. If the client pays all his old debts they close the exception handling process.

5. Conclusions

Our analysis on the support of effective unexpected exceptions highlighted a fundamental system requirement: maintain task execution under model guidance during normal operation and change to map guidance when an unexpected exception occurs,

supporting users giving the control back to model guidance after the exceptional situation is overcome. Under these circumstances, collaborative user involvement is also crucial to determine the most appropriate action.

We developed an exception handling process and a set of components to support this functionality, orchestrating the collaborative diagnosis, recovery and monitoring tasks. The diagnosis task is based on a new classification of unexpected exceptions proposed in this paper. Several dimensions characterizing the handling strategies and relationships with the classification of unexpected exceptions are proposed as well.

The major concepts underlying this framework were discussed in the context of two motivating examples. Currently, the several architectural components necessary to implement the framework have been developed, in particular related with interface 1 (interface with the WfMS), including the interventions toolbox and the exception description component. These components were developed in the context of the Open Symphony platform [21].

6. REFERENCES

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

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

[3] Casati, F., Ceri, S., Pernici, B., and Pozzi, G., Workflow Evolution, *Data and K. Engineering*, 24, 3 (1996), 211-238.

[4] Casati, F. and Pozzi, G., Modelling exceptional behaviors in commercial workflow management systems, in*, CoopIS '99*, (Edinburgh, UK, 1999), IEEE International,127-138.

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

[6] Dayal, U., Hsu, M., and Ladin, R., Organizing Long-Running Activities with Triggers and Transactions, in *SIGMOD'90*, (Atlantic City, NJ, USA, 1990).

[7] Dellarocas, C. and Klein, M., A Knowledge-based approach for handling exceptions in business processes, in *WITS'98*.

[8] Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., and Zbyslaw, A., Freeflow: mediating between representation and action in workflow systems, *CSCW*, (1996), ACM Press.

[9] Eder, J. and Liebhart, W., The Workflow Activity Model WAMO, in *CoopIS95*, (Austria, 1995).

[10] Eder, J. and Liebhart, W., Workflow Recovery, in *1st IFCIS CoopIS'96*, (Brussels, Belgium, 1996), IEEE,124 - 134.

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

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

[13] Faustmann, G., Configuration for Adaptation - A Human-centered Approach to Flexible Workflow Enactment, *Comp. Supported Cooperative Work*, 9, 3 (2000), 413-434.

[14] Han, Y., Sheth, A. P., and Bussler, C., A Taxonomy of Adaptive Workflow Management, in *CSCW - Workshop - Towards Adaptive Workflow Systems*, (USA, 1998), ACM.

[15] Hwang, S. Y., Ho, S. F., and Tang, J., Mining Exception Instances to Facilitate Workflow Exception Handling, in *6th Int. Conf. on DASFAA*, (Hsinchu, Taiwan, 1999).

[16] Jorgensen, H. D., Interaction as Framework for Flexible Workflow Modelling, in *Group '01*, (USA, 2001), ACM.

[17] Luo, Z., *Knowledge sharing, Coordinated Exception Handling, and Intelligent Problem Solving for Cross-Organizational Business Processes*. PhD Thesis, Univ. of Georgia, 2001.

[18] Mintzberg, H., *Estrutura e Dinâmica das Organizações*. Publicações Dom Quixote, 1999.

[19] Mourão, H. R. and Antunes, P., Exception Handling Through a Workflow, *CoopIS'04,* (2004), Springer,37-54.

[20] Mourão, H. and Antunes, P., A Collaborative Framework for Unexpected Exception Handling., in *CRIWG'05*, (Brazil, 2005), Springer-Verlag,168-183.

[21] The OpenSymphony project. (2005, 1/11/05) <Http://www.opensymphony.com>

[22] Part I. Terror attacks brought drastic decision: Clear the skies. (2005) <Http://www.usatoday.com/news/sept11/2002-08-12-clearskies_x.htm>

[23] 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.

[24] Rinderle, S., *Schema Evolution in Process Management Systems*. PhD Thesis, University of Ulm, 2004.

[25] Saastamoinen, H., *On the Handling of Exceptions in Information Systems*. PhD Thesis, Un. of Jyväskylä, 1995.

[26] Sadiq, S. W., On Capturing Exceptions in Workflow Process Models, in *Proceedings of the 4thBIS*, (Poland, 2000).

[27] Schmidt, K., Of maps and scripts, in *GROUP '97*, (USA, 1997), ACM Press,138-147.

[28] Sheth, A. P., Georgakopoulos, D., Joosten, S. M., Rusinkiewicz, M., Scacchi, W., Wileden, J., and Wolf, A. L., *Report from the NSF workshop on workflow and process automation in information systems, ACM SIGMOD Record, 25, 4 (1996), 55-67*.

[29] Suchman, L., *Plans and Situated Actions*. MIT Press, 1987.

[30] 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.

[31] Weber, B., Wild, W., and Breu, R., CBRFlow: Enabling Adaptive Workflow Management through Conversational Case-Based Reasoning, *ECCBR'04*, (Madrid, Spain, 2004).

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

[33] Zacarias, M., Caetano, A., Pinto, S., and Tribolet, J. Modeling Contexts for Business Process Oriented Knowledge Support., In. Knowledge Management for Distributed Agile Processes, Springer-Verlag, 2005.