BPM and Exception Handling: Focus on Organizational Resilience

Pedro Antunes

***Abstract*—This paper analyzes exception handling in business process management with a major focus on resilience, i.e. the capability to maintain operations under a wide spectrum of potential breakdowns. The research highlights the need to support various types of exceptions, including expected, planned, unexpected and true exceptions. The developed integrated support contemplates the vital human involvement in exception handling. We propose a specialized component supporting exception diagnosis, escalation to several operators and groups, collaboration support, recovery actions and monitoring the system evolution. The fundamental contribution of this research is the extension of BPM exception handling capabilities to true exceptions.**

***Index Terms*—Business Process Management, Organizational Resilience, True exceptions, Ad-Hoc Interventions.**

I. Introduction

VARIOUS organizations in multiple fields adopted process orientation with the purpose to optimize their businesses and leverage their investments in technology. Business Process Management (BPM) integrates a collection of technologies capable to translate business processes, rules and practices into computer-supported activities, relinquishing routine coordination tasks from humans and empowering complex operations with timely strategic and tactic information. Two other goals often associated with BPM include increasing the level of automation and easing structural changes in organizations through better isolation of functions such as coordination, data and resource management, messaging and service decomposition.

The process-oriented view has however one fundamental drawback. BPM assumes a rationalistic approach where organizations formalize work down to the task-level details required by the underlying technology. Unfortunately, this rationalistic approach is often infeasible or detrimental to organizational behavior. Firstly, we should consider there is a trade-off between responsiveness and formalization. High formalization takes a significant amount of time and effort from business analysts and system designers, turning organizations less responsive to turbulent environments. Less formalization avoids these problems, but challenges the capacity of BPM technology to effectively manage business activities.

Secondly, we should also consider there is a trade-off between detail and ambiguity. Most service-oriented organizations deal with great levels of informality and ambiguity when performing their daily operations. Therefore many process definitions must be kept at very generic levels of detail. On the contrary, BPM requires very detailed specifications about what, how, when, who and where tasks should be executed. So support to the one will negatively impact the other and vice versa.

Furthermore, it has been shown by various ethnographic studies done in organizations that humans do not always act as prescribed [1]. Humans also tend to lose vigilance of routine operations. In many cases the main role of process definitions is helping the operators make their own decisions in sync with the peculiarities of the real-world context.

We thus find two potentially conflicting process-oriented views. One, which is often designated *machine-oriented*, assumes that technology will take control over the business activities. The other one, referred as *human-oriented,* assumes that control depends on human discretion. Many BPM solutions have been developed with a strong focus on the machine-oriented view, thus leading to a difficult acceptance by their hosting organizations [2-4]. We expect the integration between these two views may increase acceptance.

But the consequences of the human/machine conflict extend beyond organizational acceptance. Nowadays one of the main challenges faced by organizations concerns resilience [5]: the capacity to resist major business disruptions due to unforeseeable, unexpected or catastrophic events, leading the organizational systems beyond the planned service limits without serious losses. Many highly reliable organizations in several key sectors, e.g. nuclear power, chemical production, aviation, utilities, banking, etc., already adopted principles, methodologies and mechanisms to preserve themselves when facing major business disruptions. They usually adopt the following behavior [5]: (1) flexibility understanding and acting upon the evolving situation context; (2) deference to knowledge and reliance on the experience of most knowledgeable people; and (3) capability to make decisions lacking full insights about the situation.

The recent studies on organizational resilience clearly emphasize the human-oriented perspective over business processes, since human discretion is considered fundamental to make decisions under unpredictable or uncertain contexts. However, from our point of view, one major challenge associated with organizational resilience concerns exactly the

P. Antunes is Associate Professor (with Habilitation) at the Department of Informatics of the Faculty of Sciences of the University of Lisbon, Portugal (e-mail: paa@di.fc.ul.pt).

integration between the human/machine views:

- Supporting dynamically evolving business processes under emergent situations;
- Whenever necessary, relinquishing control from the technology to support unplanned tasks, deferring control to the most adequate persons;
- Providing process guidance even in contexts where the available process definitions do not comply with the current context;
- Supporting human response to novel, innovative and challenging situations;
- Facilitating the transition from emergent to normal operations.

This paper analyzes in detail the human/machine conflict and the approaches developed to overcome its consequences. We also identify the major BPM constraints associated with this conflict. Then, we propose an approach aiming to increase organizational resilience. Our approach extends traditional BPM with the capability to relinquish control from the technology to humans when facing unforeseeable, unexpected or catastrophic situations, and regain technology control when the operations come back to routine.

The paper is organized as follows. In section II we elaborate the conceptual foundations of the human/machine conflict. Section III discusses the current state of the art in BPM. Section IV presents and discusses the developed approach. Section V is focused on implementation details. Section VI presents a case walkthrough. Section VII provides a preliminary report from the field. Finally, Section VIII discusses the implications and draws some concluding remarks.

II. The Human/Machine Conflict

Sheth et al [6] define *business process* as a collection of activities tied together by a set of precedence relations and pursuing a common organizational goal; and *workflow management* as the automated coordination of activities between workers and computers to carry out a business process. According to the Workflow Management Coalition (WfMC) [7], the infrastructure necessary to manage business processes includes at least: an *enactment service* responsible for managing processes; a *modeling component* responsible for defining business processes and injecting them into the enactment service; and a *client component* responsible for carrying out the activities by controlling the interaction with the workers and other services. This infrastructure is compliant with service oriented architectures and various types of enterprise architectures [8, 9].

We note the WfMC infrastructure is fundamentally *machine-oriented*: it assumes the enactment service controls business processes based on the normative engagement of process definitions [10]. Suchman [11] studied business processes from a sociological standpoint, analyzing the variability of human activities in organizations and the inference, interpretation and contextualization often necessary to carry out the intended goals under variable conditions. According to this *human-oriented* perspective, process definitions guide actors in a space of available actions, providing situation awareness and orientation, although not assuming a normative engagement. In this scenario humans apply their cognitive capabilities to carry out activities informed by business process definitions. The control is thus in the hands of the workers and not the computers.

These distinctions and their implications have fuelled the debate between researchers working in the two sides of the fence, especially after the Suchman's paper "Do Categories have Politics?" [12] criticized the BPM approach proposed by Winograd and Flores [13]. The aftermath of what has been called "the Suchman-Winograd dispute" demonstrates there is not exactly a fence, since both views bring important contributions to systems thinking [14, 15]. Actually, the dispute raised multiple opportunities to research the integration of human/machine views [16-18].

In this paper we address this integration from a systems' design point of view. Perhaps the first theorist to address the issue from this point of view was Simon [19]. Simon characterized design as a process aiming to develop artificial artifacts. Such artifacts must not ignore or violate natural law and should also embody human goals. Thus there is a clear distinction between the inner structure of the artifact and the outer environment where it operates, but also the need to adapt both to avoid failing the design goals. The inner depends on the outer, but the outer is also influenced by the inner.

Vicente [20] enriched Simon's framework by distinguishing different layers of complexity, comprising the technical/ engineering system, workers, organizational/management infrastructure and environmental context. Design might then be regarded as the adaptation of these different layers. Certainly the human layer is one of the most complex to design.

Rasmussen [21] developed a framework considering three levels of human performance: role-based, rule-based and knowledge-based. The first one addresses mechanistic tasks accomplished by humans when facing routine work. We find rule-based performance in situations where work activities have been planned and prescribed, although giving workers some decision latitude. And knowledge-based performance is found whenever workers face novel situations and their decision-making abilities must be fully exercised.

Reason [22] used this framework to differentiate human performance in administrative control: from prescriptive, based on procedures and rules, to discretionary, based on training and experience. In between we find mixed control situations relying on training and procedures. This perspective clearly integrates the human/machine orientations into a prescriptive-discretionary continuum of design possibilities.

Perrow [23] established the link between organizational strategy and the prescriptive-discretionary continuum. The Perrow's framework is based on the notion of *exception*. An exception occurs whenever the organization fails to accomplish the intended business goals, either because the available procedures do not apply to the current context; workers fail to understand, make decisions and act upon the

situation; or the technology creates barriers to the actions necessary to overcome the exception. The study of exceptions in organizations has demonstrated they occur quite frequently [24].

Perrow defines two dimensions of exception: (1) task variability refers to the number of exceptions encountered while performing a task; and (2) task analyzability is the degree to which search activity is needed to overcome the exception. These dimensions highlight four exception-handling strategies adopted by organizations:

- **Routine** (Low variability / High analyzability): Routine working situations have few well-known exceptions that are handled with established procedures.
- **Crafted** (Low variability / Low analyzability): Crafted work has also few exceptions, but they require informal interaction to find out what to do.
- **Engineered** (High variability / High analyzability): The number of exceptions is high but well-known plans and processes exist to handle them.
- **Non-routine** (High variability / Low analyzability): Non-routine working situations require collaboration and decision-making to find out creative responses.

This framework allows us to finally reach the core of the human/machine conflict. The issue is that by focusing solely on the routine/engineered strategies we are neglecting all the situations requiring informality, collaboration and decision-making; and focusing on the crafted/non-routine strategies we are also neglecting all the situations demanding planning and prescriptive actions.

Most highly reliable organizations are compelled to coalesce the above strategies in order to swiftly respond to exceptions, whatever they are. The absence of one single strategy will necessarily be criticized for reducing resilience. As a consequence, the design of organization's artifacts should take into consideration the functionality necessary to sustain and develop the four strategies mentioned above.

III. BPM AND EXCEPTION HANDLING

In the previous section we discussed several exception-handling strategies adopted by organizations to uphold resilience. We will use the same categories to analyze how BPM systems have been supporting exception handling.

A. Type I - Mechanistic

This category includes exceptions with low variability and high analyzability. They may be classified in three classes [25]: (1) basic failures, associated with failures in the underlying technological infrastructure, such as networking, database management and operating system; (2) application failures like unexpected data inputs; and (3) expected exceptions, events that were predicted during the process definition phase but that do not correspond to the "normal" process behavior.

Notice that failures result from system malfunctions while expected exceptions come from semantic discrepancies between the process definitions and the actual running environment. These differences lead to distinctive handling procedures. In the case of failures, the most common handling procedure is to apply transactional mechanisms to return the operations to a coherent state and proceed as planned [26]. Most of commercial BPM implementations are integrated with database management and thus supply this exception handling procedure. Another approach uses failure tolerance techniques, e.g., replicating services [27].

Various solutions have been devised to handle expected exceptions. Some rely on triggers to initiate predefined exception handlers [25, 28-31]. Others adopt special modeling constructs, triggered by conditions that identify the occurrence of a certain exception [28, 29, 32-34]. In all cases the handling procedures are planned during the design stage and applied in an automated way.

It has been suggested that basic and application failures that cannot be handled at the level where they occur should be propagated as expected exceptions and handled as such by the BPM system [29, 35]. This allows encapsulating failure handling within the scope of expected exception handling, the reason why we will not consider further the impact of failures on BPM.

B. Type II - Planned

This category includes exceptions with high variability and high analyzability. They require human involvement to analyze the organizational performance and delineate new working methods. These exceptions typically emerge from incomplete designs, design errors and structural changes in the business environment [36, 37].

The commonly adopted exception handling procedure is to dynamically redesign the affected business processes. This approach requires the capability to adapt processes running in the enactment service without any disruption in the operations [38]. The major problem that has been addressed by the research is to guarantee the system consistency and correctness when migrating process instances [39-42]. Other research lines explore change patterns [4, 43] and exception mining [44, 45].

The dynamic changes must be executed under strict system control, avoiding deadlocks, unreachable states and other inconsistencies. The researchers have developed a set of change rules enabling correctness checks before applying dynamic changes [46, 47].

C. Type III - Informal

This category includes exceptions with low variability and low analyzability. The major difference to the planned strategy is that in this case there is no need, justification or enough time to redesign the business processes.

The informal strategy responds to unexpected exceptions [25]. These exceptions result from lack of knowledge about the business process, its details and variations [36].

BPM technology may deal with unexpected exceptions in two different ways: late binding and ad-hoc changes. Late binding uses loosely specified processes that are only detailed in runtime [45]. Ad-hoc changes are typically applied to a

small set of process instances and have a transient impact on the system behavior [38, 48, 49]. This includes, for instance, delaying an activity, designating another operator to accomplish an activity, and inserting an activity not previously defined by the normal process. In both cases a mixed control policy is necessary, combining the momentary human control crucial to understand the situation and make decisions with the system control required to preserve consistency and correctness.

D. Type IV - Non-routine

This category includes exceptions with high variability and low analyzability. They have been designated true exceptions [24]. True exceptions occur when the organization has no previous plan, rules or knowledge about the exception. True exceptions may be seen as dramatic events (in the case of accidents and emergencies) or strategic opportunities for changing the business. In both cases they demand human judgment under incomplete information.

We should consider there might not be enough time to plan dynamic changes when facing true exceptions. In these situations ad-hoc changes to business processes are necessary, but should not be constrained by the consistency and correctness requirements [47]. If any system barriers are imposed to the organization's primary goal, then the organization will find workarounds to the system [50]. Therefore, true exceptions set challenges different to the ones discussed regarding the informal strategy.

Few approaches have been documented in the literature addressing true exceptions in BPM. One of them integrates BPM with external collaboration tools [51]. The purpose is to normally maintain control in the enactment service but passing it to collaboration tools when a true exception occurs. However, no support was considered to continue with the normal operations after resolving the exception. A mechanism to determine the type of control more adequate to tackle various types of exceptions, including true exceptions, has also been researched [52]. This mechanism was conceived to invoke decision-support tools when true exceptions occur. However, it does not involve humans in the process of determining the best strategy, neither considers changing the strategy according to the evolution of the situation.

Other strategies aim to support decision-making and are only indirectly linked to BPM. One case uses a knowledge base to maintain information regarding past handling procedures and to facilitate linking exceptions to handling procedures [53]. Another case uses data mining to extract relevant information about the exception and support organizational decision making [44].

E. Summary

From the above overview we realize that automated exception handling is crucial to increase the organization's capability to resist expected exceptions. However, when other types of exceptions occur, human intervention is always required and workers become a fundamental component supporting organizational resilience.

Regarding this human role, the dynamic redesign increases resilience by migrating business processes towards new definitions, more capable to accomplish new organizational goals. The late binding and constrained ad-hoc changes further increase resilience by allowing more immediacy and less planning. And finally, the unconstrained ad-hoc interventions provide an increased level of resilience by giving wider latitude of action and decision-making support.

Summarizing the whole scenario, we observe that organizations must integrate various types of exception-handling, covering the path from fully automated to fully human discretionary actions. However, as we have seen above, few approaches integrate fully human discretionary actions in BPM.

Considering this scenario, our research goals are centered on the integrated support to exception handling, covering in particular constrained and unconstrained had-hoc interventions in BPM. The next section details our approach.

IV. An Approach to Support Ad-Hoc interventions

We previously characterized the generic BPM infrastructure with three main components: enactment service, modeling component and client component. Let us now analyze their capability to handle exceptions. The expected exceptions are handled by the enactment service in conjunction with the modeling component. The enactment service implements the runtime mechanisms necessary to detect an exception and invoke the corresponding handling procedure, while the modeling component contributes with the language constructs necessary to define the exception handling procedure.

Regarding planned exceptions, they may also be handled by combining the modeling component and enactment service, the former supporting process redesign and consistency checks, and the later supporting dynamic changes to running process instances.

Unexpected exceptions are more challenging. On the one hand, the enactment service is capable to support and control some changes to process instances, such as delaying or repeating a task. But, on the other hand, the insertion of ad-hoc tasks without a proper process definition and strict control becomes problematic to manage. Often, there is no track record of the tasks effectively accomplished, which may lead the system to a completely ad-hoc venture. We thus need an additional component managing ad-hoc interventions in the enactment service.

True exceptions are beyond reach of most BPM infrastructures. Besides difficulties managing unconstrained ad-hoc interventions, we consider they lack support to decision-making, collaboration and knowledge management.

Our approach extends the BPM infrastructure with a new component designated control switch. The control switch is responsible for moving control out of the enactment service whenever some types of exceptions are detected. The control switch supports ad-hoc interventions in the enactment service and is responsible for orchestrating, monitoring and keeping records of ad-hoc interventions.

In the following we will further detail the control switch. We consider the proposed approach adopts this general behavior:

1. Under normal conditions, we assume the organization operates in a mechanistic way. This means the business processes are managed under the strict control of the enactment service;
2. The occurrence of an expected exception is handled by the enactment service, invoking the corresponding handling procedure. That procedure was predefined at design time and is expected to bring the situation to a normal condition;
3. However, sometimes these procedures are incapable to resolve the situation and fail. Other times there is no handling procedure associated with the occurring exception. In both cases they require human intervention and thus the control switch is invoked;
4. The control switch is invoked whenever three types of exceptions occur: planned exceptions, unplanned exceptions and true exceptions;
5. The control switch requests humans to diagnose the exception. A *diagnosis component* (described later) will support this functionality. Diagnosis is an ongoing activity.
6. Having a diagnosis, the control switch may contemplate the three possible routes described below;
7. If the organization is facing a planned exception, then there is time to redesign business processes. The redesign will involve the modeling component and enactment service, and therefore the control switch will finish its own service;
8. If the organization is facing an unexpected or true exception, then the workers may have to apply recovery actions in the enactment service. A *recovery component* (described later) will support these actions. The distinction between unexpected and true exceptions is related with the level of control assumed by the control switch. In the former case, the control switch restricts the recovery actions to preserve the consistency and correctness of the processes managed by the enactment service, while in the second case no restrictions will be considered.
9. The control switch will continue operating until the workers are able to bring back the organization to a normal condition and declare the exception is resolved.

We note this exception handling process may be modeled and managed as a common work process. Indeed, we have implemented it that way. More implementation details are given later.

A. The control switch components

We will now discuss in more detail the components necessary to implement the control switch. The first one to consider is the *detection component*. It is responsible for detecting exceptions and instantiating exception handling processes. The workers, through a user-interface, or the enactment service using a trigger, may raise exceptions.

TABLE 1
EXCEPTION ATTRIBUTES

Attribute	**Description**	**A**	**M**
Type of detection	*Automatic* or *manual*	●	
Type of exception	*Planned*, *unexpected* or *true*		●
Liable person	Person in charge of the exception handling	● [1]	●
Affected persons	Persons involved in the process		●
Affected instances	Process instances affected by the exception	● [2]	●
Affected tasks	Tasks affected by the exception	● [2]	●
Affected processes	Processes affected by the exception	● [2]	●
Exception description	Textual description of what may have occurred		●
Handling description	Textual description of what should be done to resolve the exception		●

A – Automatically specified; M – Manually specified.

[1]The liable person is automatically specified just after the exception is raised: (1) if manually detected, whoever raised the exception; (2) if automatically detected, the task/process owner.

[2]The affected processes, process instances and tasks are automatically determined by the system just after the exception is raised, but may be redefined afterwards by the liable person.

The *diagnosis component* serves to characterize the exception using the attributes shown in Table 1 [54]. Some of the attributes are automatically determined by querying the enactment service, while humans must explicitly define the other attributes. In any case, it should be emphasized that: (1) the liable person may change the attribute values over time; and (2) the liable person may also designate a new liable person, delegating the capacity to reanalyze the situation and change the exception attributes.

Thus two important functional capabilities associated with the diagnosis component include associating the exception attributes with a timeline and supporting a retrospective view of the exception, from the initial detection to the current time, showing what has changed and who was involved. The whole dynamics of exception diagnosis will be described in more detail in the case walkthrough.

The *recovery component* interfaces with the enactment service to implement recovery actions. This includes a set of quasi-atomic actions, such as cancel, jump forward and backward, repeat and suspend [41].

As implied by the diagnosis, the control switch also needs an *escalation component*. The escalation component is necessary to bring more people to the exception handling process. We consider four levels of escalating human involvement: (1) the *operator*, when one individual is responsible for the necessary recovery actions; (2) the *peers*, when the recovery actions are still accomplished by one individual but multiple co-workers may communicate to analyze and discuss the problem; (3) the *supervisor*, when the responsibility changes from the worker to the supervisor, but the worker is yet able to contribute to analyze and discuss the problem; and finally (4) the *group*, when a worker designates a group of persons to get involved in the recovery actions.

The liable person is the only one that is able to escalate the exception by designating another operator, peer, supervisor or group. Just like diagnosis, escalation is an ongoing activity. The responsibility may reside in one person and in a while escalates to another person, peers, supervisor or group.

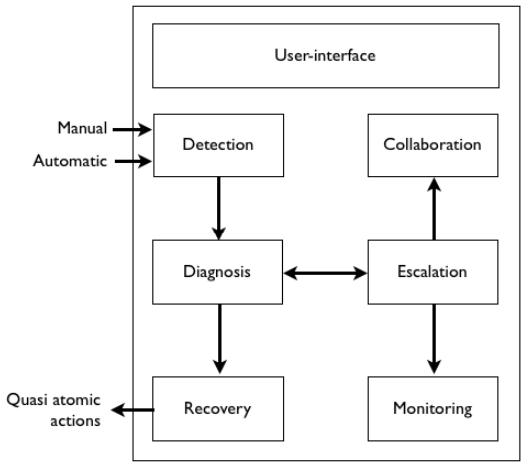

Fig. 1. Control switch components

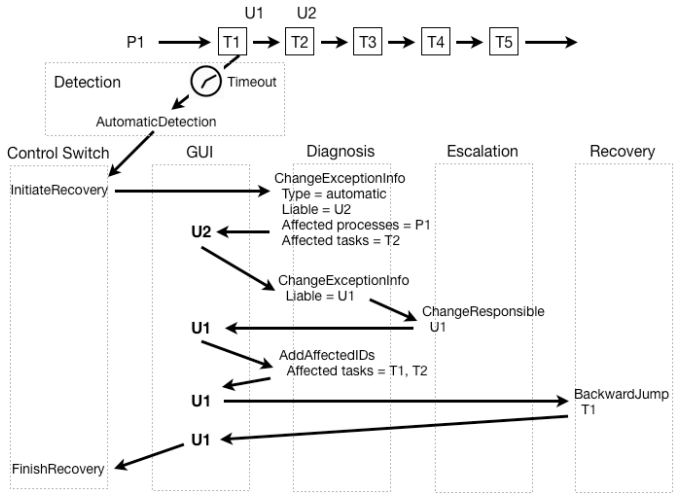

Fig. 2. Implementation details

The *collaboration component* integrates the control switch with any collaboration tools that may be available to support communication, situation awareness and decision-making. It does not implement collaboration, but instead interfaces with a variety of external tools such as e-mail, messaging and chatting. This approach also supports linking the control switch with more complex group support and decision support systems, or even specialized emergency management tools [55, 56]. The collaboration component initiates collaboration between the persons selected by the escalating component.

The *monitoring component* serves to instantiate ad-hoc tasks dedicated to monitor the system evolution. The tasks themselves are managed by the enactment service. The monitoring component might be viewed as a kind of modeling component specifically dedicated and constrained to collect external data related with the exception handling process.

The final component considered by our approach is the *user-interface component*. This component interfaces with the persons involved in the exception handling, giving then access to the services provided by the other components, with the additional preoccupation to facilitate situation awareness, decision-making and action. In Figure 1 we illustrate the several components and relationships that assemble the control switch.

V. Implementation Details

The control switch was implemented in Java and integrated in the OpenSymphony/OSWorkflow open source platform [57-59]. This platform provides a basic enactment service for BPM using XML descriptors for work process specification and HTML for client integration with Web browsers.

In Figure 2 we illustrate the system behavior. The low-level details on how exceptions are triggered in OpenSymphony will not be discussed here. We just point out that after detecting an exception the system requests the control switch to initiate recovery. In the specific case shown in Figure 2 the exception is a timeout of task T2 assigned to user U2.

Currently, only one process instance may be enacted for each detected exception (we have not implemented nested exceptions). OpenSymphony executes that process instance exactly like any other process instances.

After the exception handling process starts, the initial diagnosis information is set using the current details about the exception. These details include: type of detected exception, which is automatic in this case; liable person, who is U2, the user responsible for T2, the current task when the exception was raised; and affected processes and tasks, respectively P1 and T2. All this information is accessible by querying the OSWorkflow object store.

Then the diagnosis component requests U2 to complete or update the diagnosis. In the illustrated example U2 realizes that U1, who is responsible for task T1, should assume the exception handling. U2 designates U1 as the new liable person. Consequently, the escalation component brings U1 to the exception handling process. Note, for the sake of completeness, that U2 could instead have designated U1 as a peer, supervisor or group member, which would have involved the collaboration component.

U1 is then requested by the diagnosis component to analyze the exception and make any necessary changes to the diagnosis. As mentioned before, the diagnosis component allows overviewing the evolution of exception attributes. As shown in Figure 2, U2 decides to affect task T1 to the exception and then requests a recovery action: the process should do a backward jump to task T1. This invokes the corresponding action in the recovery component, which is a low-level interface to the OSWorkflow object store that allows changing the process state and next executable task to T1. The exception handling process finishes with a request from U1.

Figure 2 does not illustrate the use of the collaboration component, which is invoked when the liable person brings other persons to exception handling. This component manages the interaction between the liable person and the other users. Three interaction modes are supported: (1) designating a peer, with whom the liable person will be able to interact but that will not be involved in the exception handling in any other way; (2) moving up the liability to the supervisor, which is similar to the case shown in Figure 2 but allows the supervisor to interact with the subordinate; and (4) involving a group in

the exception handling.

When a group is involved, a parallel recovery task is instantiated for each member, so they may concurrently execute recovery actions. The possible side effects of concurrent recovery actions are not considered in our implementation. We note however such effects may be mitigated through collaboration.

Currently, the collaboration component offers two interaction mechanisms: e-mail and chat. This allows the operators to adopt a synchronous or asynchronous collaboration mode. Regarding the monitoring component, it is capable to instantiate ad-hoc tasks that gather information from an information source after a predefined timeout. The implementation allows gathering information from a database, using a SQL query, and from a URL.

And we finally mention the user-interface component (GUI in Figure 2). OpenSymphony adopts HTML and Web browsers to interface with the users. Therefore the user-interface component consists of a collection of HTML pages invoking functions supported by the control switch API. More details may be found in [59].

VI. Case Walkthrough

In this section we walkthrough the exception handling process from the organization's point of view. The case considers the delivery of goods in the automobile industry from the north to the southeast of Europe; and the exceptional event concerns a strike blocking road traffic in the middle of Europe. The extension of the strike disallows adopting the simple solution of choosing an alternative road. The usual contractual terms in this industry impose financial penalties when the goods are not delivered on time, which pushes the producers to find a creative solution.

TABLE 2
EXCEPTION HANDLING EXAMPLE

Organizational behavior	System use	Comments
Goods packaged for delivery	Goods packaged for delivery	Process evolves according to the specification; control is in the enactment service
Truck arrives	Truck arrives	
Truck leaves with goods	Truck leaves with goods	
Strike blocks truck halfway to destination	Exception occurs; process is suspended	Exception handling process is instantiated; control is passed to the control switch
Driver realizes the delivery will not be on schedule	Driver diagnosis true exception	Diagnosis component
Driver contacts supervisor and asks what to do	Driver brings supervisor to exception	Escalation component
Supervisor discusses with other workers and realizes no alternative road is available	Supervisor brings other workers to the exception and uses chat tool to discuss the problem	Escalation and collaboration components
Supervisor realizes new goods could be delivered by air, but has no authority to do it	Supervisor uses chat tool to realize that goods could be delivered by air	Collaboration component
The CEO is contacted for approval	Supervisor brings CEO to the exception and uses chat tool to obtain approval	Escalation and collaboration components
CEO approves	CEO uses chat tool to approve	Collaboration component
Worker is assigned by the supervisor to proceed with plane rental	Supervisor creates ad-hoc task to deploy solution; supervisor also instantiates monitoring task to detect goods delivery	Recovery and monitoring components; the worker is assigned to the ad-hoc task
Find plane rental company	Ad-hoc task	
Rent plane	Ad-hoc task	
New goods packaged for delivery	Ad-hoc task	
Goods sent to airport	Ad-hoc task	
Plane leaves	Ad-hoc task	
Plane arrives to destination	Ad-hoc task	
Goods delivered	Monitoring task triggered	
Process complete	Supervisor executes recovery task to finish the process	Recovery component

Table 2 describes the case walkthrough. On the left we show the typical response from the organization to the exception. The case illustrates the integration between the routine (e.g., the first 3 tasks), crafted (e.g., the driver contacting the supervisor by phone) and non-routine strategies (e.g., supervisor contacting CEO for support to the plane rental solution).

The case also illustrates what in these circumstances would happen to the enactment service in the absence of the control switch. The most probable behavior would have the process suspended after the "truck leaves with goods" task triggers a timeout. This would require a maintenance operation in the enactment service after the events to properly terminate the process.

In the middle column we illustrate the system behavior proposed in this paper. The case illustrates the extended system support after the occurrence of the exception and throughout the decision-making activities necessary to overcome the exception. The system support comprehends not only escalating the exception to the right people but also supporting collaboration, monitoring exception handling and recovering normal operations. Additionally, we observe the proposed approach allows keeping records of the activities, thus contributing to build organizational memory.

VII. Report From the Field

In this section we report the field tests of our approach. The main driver for these field tests was the deployment of a BPM system for a Port Authority [58]. The control switch was an add-on to the BPM system.

We defined two goals for the field tests: (1) to assess, in real-world conditions, the viability of the control switch, considering in particular its integration with BPM; and (2) to obtain preliminary indications about the organization's behavior during exception handling. We start discussing the

first goal.

The Port Authority has a concession to manage the river and shore activities within a coastal jurisdiction assigned by the government. The Port Authority controls the river traffic and cargo transfers to and from ships. A large number of companies and individuals (e.g., fisherman) operate within the port area. The Port Authority licenses designated spaces on the shore for these activities, while the licensees assume contractual obligations and pay fees. But managing contracts and obligations in this context is rather complex, mostly because the port combines large-scale industrial activities (e.g., exporting automobiles and shipyard maintenance) with small-scale, traditional fishing activities.

The Port Authority set up a project to develop a BPM system improving license management and control. The system was specifically developed for the Space Rentals Department. It supports several administrative tasks and, in particular, automatically verifies that every client pays its fee every month. One important requirement concerns having a permanently updated list of debts and free/occupied zones in the port.

The BPM system delivered to the Port Authority not only supports license management and control but also exception handling using the control switch described in this paper. The results of this integration were quite satisfactory. The OpenSymphony platform offers a Web front-end for task management that was easily integrated in the workers' environment. Exception handling was tightly coupled with the other functionality. For instance, each task interface displayed to the users integrates a button that serves to trigger an exception. The users manage the exception handling tasks in the same way as the other tasks, although without the rules imposed by the modeled business process. The integration with e-mail and chat also revealed easy because the workers already used the technology.

Regarding the OpenSymphony implementation, the lack of support to high-level process specification tools and a sophisticated enactment service were compensated by the free access to the platform's object store, which facilitated the implementation of recovery actions. Thus our goal to assess the viability of the control switch was achieved.

Our second goal was quite more difficult to achieve. The integration of an exception handing mechanism in an organization requires training and commitment, which were not available because of the project's constraints already mentioned. Furthermore, administrative units such as the Space Rentals Department do not deal with many unexpected/true exceptions, at least within the timeframe we had to develop the project.

Nevertheless, we could follow one such event. We will describe in detail that particular exception, showing how the system and workers reacted to the situation. The real names were changed to preserve anonymity.

The observed exception evolved according to the following events. Henry, working on the Space Rentals Department, was updating the client's database record when someone told him informally the client had in fact bankrupted. This was important information, with obvious impact on the Port Authority. However, the modeled business process did not consider any specific provisions for handling such an event. Clearly, this seemed to be an unexpected exception.

In other circumstances, the event would certainly continue its path through the rumor mill, but in this case the system offered the users the chance to trigger an exception. In fact, every task displayed in the browser offered the option to trigger an exception. Henry used that option.

As a consequence of this action, a task was assigned to Henry to diagnose the exception. The task appeared on Henry's browser along with the other tasks assigned. Then Henry inserted a brief description of the event in the corresponding field and classified the exception as unexpected. The affected instances were automatically defined by the system: it affected the monthly payment task for which Henry was the liable person.

Henry did not know the whole implications of the bankruptcy and lacked authority to make any necessary provisions, and therefore decided to involve John, his supervisor, to whom he would delegate the problem. After being involved, John became the liable person. The diagnosis task appeared on John's browser.

Since the situation was unclear, John decided to invoke the chat tool to discuss with Henry. The collaboration component launched the chat tool with two users: Henry and John.

During the chat, John realized that another company was requesting the space, coming to understand the full extent of the raised problem. Henry also informed John the client's debt was about 50.000€. John then decided to change the exception to a true exception. He also involved Philip from the legal department in the process.

The collaboration component established a chat between Henry, John and Philip. After discussing the situation, they decided that Philip should consult with an external lawyer. Henry should wait for any news from Henry. The chat session was then closed.

Philip was later on counseled to notify the client by registered mail, giving 5 days to pay the debt. Obtaining no response, they should start a lawsuit action. He then invoked the chat tool to discuss with Henry and John who should send the letter and who should follow up the process.

They agreed that Henry would be responsible for sending the letter. If the client pays the debt within 5 days they would close the exception handling process. Otherwise they should advance with the lawsuit action. An ad-hoc monitoring task was instantiated by John to remember Henry after 5 days.

After 5 days Henry was notified. He recognized the debt was paid and established again a chat to inform John. John decided to finish the exception handling process.

The exception handling process managed the interactions necessary to handle this particular case. We observed it was easy for the participants to involve an expert from another department in the process and keep records of what was going on. The relevant decisions were spread among the participants. The exception handling process allowed users to improve overall situation awareness while resolving a problem with the

business process but without affecting the organization behavior. The problem was resolved in an ad-hoc way, although orchestrated by the control switch.

Thus our preliminary report from the field indicates the control switch integrates with current BPM technology and supports exception handling activities, while improving situation awareness and organizational memory. Though more long-term experiments are necessary to evaluate the overall impact on the organization.

VIII. Discussion and Conclusions

The Reason's [22] perspective over the organizational strategies necessary to overcome exceptions is the fundamental key to understand the limitations and possibilities of BPM. Many of these systems have been developed under the Low variability / High analyzability assumption. Several developments aimed at improving the flexibility of BPM extended them to the Low variability / Low analyzability and High variability / High analyzability strategies. The High variability / Low analyzability strategy has however been under developed [60].

But High variability / Low analyzability strategies are fundamental to improve organizational resilience. Our research not only shows that these strategies may be integrated with BPM systems but also that they improve situation awareness and organizational memory.

Also, these strategies give the operators more latitude for intervening in the system with immediate ad-hoc recovery actions. But such recovery actions may introduce inconsistencies in the system. Our solution addresses this problem by supervising the system evolution during exception handling through the control switch. The control switch supports exception diagnosis, ad-hoc recovery actions (constrained and unconstrained), monitoring tasks, escalating exceptions to several workers and collaboration between them.

We described the detailed functionality of the control switch at three different levels: conceptual, organizational and infrastructural. At the conceptual level, we discussed the combined operation of the various components that set up the control switch. The diagnosis component classifies the exception according to a set of attributes. It supports dynamic changes to diagnosis and retrospective analysis of the exception evolution.

The escalation component orchestrates workers in the exception handling process, thus addressing a fundamental resilience principle: involving the most knowledgeable persons to overcome the exceptional situations. The collaboration component relies on the escalation component to establish synchronous and asynchronous communication between the workers involved in the exception handling. The collaboration component addresses another important resilience principle: empowering perception, awareness and decision-making through participation and collaboration. The recovery component implements the recovery actions determined by the operators.

Regarding the organizational level, we described in detail how the operators might use the exception handling process to respond to non-routine situations. The proposed solution integrates BPM with several collaborative tools already common in organizations, such as email and chat tools.

And finally, regarding the infrastructural level, we illustrated in detail the orchestration of events necessary to manage the system evolution towards the “normal” behavior after the occurrence on an exception. The overall functionality combines end-user interaction with low-level access to the data structures managed by the enactment service.

The preliminary report from the field indicates the proposed solution integrated well with OpenSymphony. One fundamental contribution of this research is tracking the operators’ communications, interactions, collaborations and recovery actions, this way building organizational memory. Of course longitude studies are necessary to further validate this integration. In particular, future research should analyze the impact on the system evolution of: (1) increasing the number of workers involved in the exception handling process; (2) having a large number of workers concurrently applying unconstrained ad-hoc interventions in the system; and (3) having multiple concurrent exceptions triggered by the system. Also, since we have not implemented consistency checks, we have not fully explored the differences between unexpected and true exceptions.

The proposed solution was implemented in a real-world organization. Nevertheless, more extensive studies are necessary to understand how to support ad-hoc interventions in organizations. Future studies should also focus on broader organizational issues such as organizational learning, decision-making and collaboration. Another research line, which is necessary to continue studies in this area, concerns the development of objective measures of exception handling.

References

[1] J. Bowers, G. Button, and W. Sharrock, "Workflow From Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor," in *Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden, 1995, pp. 51-66.

[2] B. Mutschler, M. Reichert, and J. Bumiller, "Unleashing the Effectiveness of Process-Oriented Information Systems: Problem Analysis, Critical Success Factors, and Implications," *IEEE TRansactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 38, pp. 280-291, 2008.

[3] W. van der Aalst and P. Berens, "Beyond Workflow Management: Product-Driven Case Handling," in *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, Boulder, Colorado, USA, 2001, pp. 42-51.

[4] B. Weber, M. Reichert, and S. Rinderle, "Change patterns and change support features – Enhancing flexibility in process-aware information systems," *Data & Knowledge Engineering,* vol. 66, pp. 438-466, 2008.

[5] E. Hollnagel, D. Woods, and N. Levenson, *Resilience Engineering: Concepts and Precepts*. Hampshire, England: Hashgate, 2006.

[6] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf, "Report from the NSF Workshop on Workflow and Process Automation in Information Systems," *ACM SIGMOD Record,* vol. 25, pp. 55-67, 1996.

[7] WfMC, "Workflow Management Coalition - Terminology & Glossary " WfMC 1999.

[8] F. Leymann, "Web services and business process management," *IBM Systems Journal,* vol. 41, p. 198, 2002.

[9] T. Arora and A. Nirpase, "Next Generation Business Process Management: A Paradigm Shift," in *Proceedings of the 2008 IEEE Congress on Services - Part I - Volume 00*, Washington, DC, 2008, pp. 81-82.

[10] K. Schmidt, "Of maps and scripts," in *GROUP 97 International Conference on Supporting Group Work*, Phoenix, Arizona, 1997, pp. 138-147.

[11] L. Suchman, *Plans and Situated Actions: The problem of human-machine communication*. New York, NY: MIT Press, 1987.

[12] L. Suchman, "Do categories have politics?," *Computer Supported Cooperative Work,* vol. 2, pp. 177-190, 1993.

[13] T. Winograd and F. Flores, *Understanding computers and cognition: A new foundation for design*. Norwood, New Jersey: Ablex, 1986.

[14] CSCW, *Computer Supported Cooperative Work,* vol. 3, 1994.

[15] T. Winograd, "Designing a new foundation for design," *Communications of ACM,* vol. 49, pp. 71-74, 2006.

[16] J. Taylor and S. Virgili, "Why ERPs Disappoint: the Importance of Getting the Organisational Text Right," in *ERP Systems and Organisational Change* London: Springer, 2008, pp. 59-84.

[17] T. Herrmann and K. Loser, "Vagueness in models of socio-technical systems," *Behaviour & Information Technology,* vol. 18, pp. 313-323, 1999.

[18] R. Grinter, "Workflow Systems: Occasions for Success and Failure," *Computer Supported Cooperative Work,* vol. 9, pp. 189-214, 2000.

[19] H. Simon, *The Sciences of the Artificial*. Cambridge, USA: The MIT Press, 1996.

[20] K. Vicente, *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work*: Lawrence Erlbaum Associates, Inc., 1999.

[21] J. Rasmussen and A. Jensen, "Mental procedures in real-life tasks : a case-study of electronic trouble shooting," *Ergonomics,* vol. 17, pp. 293-307, 1974.

[22] J. Reason, *Managing the risks of Organizational Accidents*. England: Ashgate, 1997.

[23] C. Perrow, *Organizational Analysis: A Sociological View*. United Kingdom Tavistock Publications, 1970.

[24] H. Saastamoinen, "On the Handling of Exceptions in Information Systems," University of Jyväskylä, PhD Thesis, 1995.

[25] J. Eder and W. Liebhart, "The Workflow Activity Model WAMO," in *Proceedings of the Third International Conference on Cooperative Information Systems*, Vienna, Austria, 1995, pp. 87-98.

[26] D. Worah and A. Sheth, "Transactions in Transactional Workflows," in *Advanced Transaction Models and Architectures*: Kluwer, 1997.

[27] G. Alonso, C. Hagen, D. Agrawal, A. El Abbadi, and C. Mohan, "Enhancing the fault tolerance of workflow management systems," *IEEE Concurrency,* vol. 8, pp. 74 -81, 2000.

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

[29] D. Chiu, Q. Li, and K. Karlapalem, "WEB Interface-Driven Cooperative Exception Handling in ADOME Workflow Management System," *Information Systems,* vol. 26, pp. 93-120, 2001.

[30] S. Sadiq, "On Capturing Exceptions in Workflow Process Models," in *Proceedings of the 4th International Conference on Business Information Systems*, Poznan, Poland, 2000.

[31] Z. Luo, "Knowledge sharing, Coordinated Exception Handling, and Intelligent Problem Solving for Cross-Organizational Business Processes," Department of Computer Sciences, University of Georgia, PhD Thesis, 2001.

[32] R. Muller, U. Greiner, and E. Rahm, "AgentWork: a workflow system supporting rule-based workflow adaptation," *Data & Knowledge Engineering,* vol. 51, pp. 223-256, 2004.

[33] Q. Chen and U. Dayal, "A transactional nested process management system," in *Proceedings of the Twelfth International Conference on Data Engineering*, New Orleans, Louisiana, 1996, pp. 566-573.

[34] C. Combi, F. Daniel, and G. Pozzi, "A Portable Approach to Exception Handling in Workflow Management Systems," in *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006*. vol. 4275, R. Meersman and T. Zahir, Eds. Heidelber: Springer, 2006, pp. 201-218.

[35] J. Eder and W. Liebhart, "Workflow Recovery," in *first IFCIS International Conference on Cooperative Information Systems (CoopIS)*, Brussels, Belgium, 1996, pp. 124-134.

[36] P. Heinl, "Exceptions during Workflow Execution," in *Proceedings of the EDBT Workshop on Workflow Management Systems*, Valencia, Spain, 1998.

[37] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi, "Specification and Implementation of Exceptions in Workflow Management Systems," *ACM Transactions on Database Systems,* vol. 24, pp. 405-451, 1999.

[38] M. Adams, A. Hofstede, D. Edmond, and W. Van der Aalst, "Worklets: a service-oriented implementation of dynamic flexibility in workflows," in *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006*. vol. 4275, R. Meersman and T. Zahir, Eds. Heidelber: Springer-Verlag, 2006, pp. 291-308.

[39] Z. Li, M. Zhou, and N. Wu, "A Survey and Comparison of Petri Net-Based Deadlock Prevention Policies for Flexible Manufacturing Systems," *IEEE TRansactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 38, pp. 173-188, 2008.

[40] C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic change within workflow systems," in *Proceedings of Conference on Organizational Computing Systems*, Milpitas, CA, 1995, pp. 10-21.

[41] M. Reichert, P. Dadam, and T. Bauer, "Dealing with Forward and Backward Jumps in Workflow Management Systems," *Software and Systems Modeling,* vol. 2, pp. 37-58, 2003.

[42] M. Weske, "Formal foundation and conceptual design of dynamic adaptations in a workflow management system," in *Proceedings of the 34th Annual Hawaii International Conference on International Conference on System Sciences*, 2001, pp. 2579-2588.

[43] M. Adams, A. Hofstede, W. Van der Aals, and D. Edmond, "Dynamic, Extensible and Context-Aware Exception Handling for Workflows," in *Proceedings of the OTM Conference on Cooperative information Systems* vol. 4803, F. Curbera, F. Leymann, and M. Weske, Eds. Heidelberg: Springer-Verlag, 2007, pp. 113-130.

[44] D. Grigori, F. Casati, U. Dayal, and M. Shan, "Improving Business Process Quality through Exception Understanding, Prediction, and Prevention," in *Proceedings of the 27th international Conference on Very Large Data Bases*, Rome, Italy, 2001, pp. 159-168.

[45] B. Weber and W. Wild, "An Agile Approach to Workflow Management," in *Proceedings of Modellierung 2004*, Marburg, Germany, 2004, pp. 187-201.

[46] W. van der Aalst and T. Basten, "Inheritance of workflows: an approach to tackling problems related to change," *Theoretical Computer Science,* vol. 200, pp. 125-203, 2002.

[47] S. Rinderle, M. Reichert, and P. Dadam, "Evaluation of Correctness Criteria for Dynamic Workflow Changes," in *Conference on Business Process Management 2003*, Eindhoven, The Netherlands, 2003, pp. 41-57.

[48] H. Mourão and P. Antunes, "Exception handling through a workflow," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*. vol. 3290, R. Meersman and Z. Tari, Eds. Heidelberg: Springer-Verlag, 2004, pp. 37-54.

[49] A. Agostini and G. De Michelis, "A light Workflow Management System Using Simple Process Models," *Computer Supported Cooperative Work,* vol. 9, pp. 335-363, 2000.

[50] N. Hayes, "Work-Arounds and Boundary Crossing in a High Tech Optronic Company: The Role of Co-Operative Workflow Technologies," *Computer Supported Cooperative Work,* vol. 9, pp. 435-455, 2000.

[51] N. Guimarães, P. Antunes, and A. Pereira, "The integration of workflow systems and collaboration tools," in *Workflow*

Management Issues and Interoperability. vol. 164, A. Dogac, L. Kalinichenko, M. Ozsu, and A. Sheth, Eds. Heidelberg: Springer Verlag, 1997, pp. 222-245.

[52] A. Bernstein, "How can cooperative work tools support dynamic group process? bridging the specificity frontier," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, Philadelphia, 2000, pp. 279-288.

[53] M. Klein and C. Dellarocas, "A Knowledge-Based Approach to Handling Exceptions in Workflow Systems," *Computer Supported Cooperative Work,* vol. 9, pp. 399-412, 2000.

[54] H. Mourão and P. Antunes, "A Collaborative Framework for Unexpected Exception Handling," in *Groupware: Design, Implementation, and Use*. vol. 3706, H. Fuks, S. Lukosch, and A. Salgado, Eds. Heidelberg: Springer-Verlag, 2005, pp. 168-183.

[55] J. Wang, W. Tepfenhart, and D. Rosca, "Emergency Response Workflow Resource Requirements Modeling and Analysis," *IEEE TRansactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 39, pp. 270-283, 2009.

[56] C. Sapateiro and P. Antunes "An Emergency Response Model Toward Situational Awareness Improvement," in *International Conference on Information Systems for Crisis Response and Management*, Göteborg, Sweden, 2009.

[57] OpenSymphony, "The OpenSymphony Project."

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

[59] H. Mourão, "Supporting Effective Unexpected Exception Handling in Workflow Management Systems within Organizational Contexts," University of Lisboa, Doctoral Dissertation, 2008.

[60] W. van der Aalst, M. Weske, and D. Grunbauer, "Case handling: A new paradigm for business process support," *Data & Knowledge Engineering,* vol. 53, pp. 129-162, 2005.