Adding a Resilience-Enhanced Component to the WfMC Reference Architecture

Pedro Antunes[1], Hernâni Mourão[2]
[1]*Faculty of Sciences of the University of Lisboa, Portugal*
paa@di.fc.ul.pt
[2]*Escola Superior de Ciências Empresariais, Instituto Politécnico de Setúbal, Portugal*
hrmourao@gmail.com

Abstract

This paper discusses the extension of the Workflow Management Coalition (WfMC) reference architecture to support organizational resilience. We propose a workflow component responsible for moving control between the system and the operators. The internal architecture of this component is discussed in detail, exposing the fundamental resilience properties: diagnosis, escalation, collaboration, monitoring and recovery.

Keywords: Workflow Management, Organizational Resilience, Resilient Workflow.

1. Introduction

Various organizations optimize their business through process orientation. Workflow Management Systems (WfMSs) support this process view, relinquishing routine coordination tasks from humans, increasing automation and easing structural changes in the organizations through better service decomposition. This process orientation has however one fundamental problem: assuming that organizations formalize business processes down to the task-level details required by WfMSs. But that is often not the case [1]. Firstly, there is a trade-off between responsiveness and formalization. High formalization makes organizations less responsive to change, while low formalization increases responsiveness but challenges the systems' capability to control the business.

Secondly, many service-oriented organizations deal with high levels of informality and ambiguity, and their business processes must avoid the details required by WfMSs. We thus find two conflicting process-oriented views, usually referred as *human-oriented,* when human discretion is assumed; and *machine-oriented*, when it is assumed that the technology will take control over the business. These views should be taken into account when building the business process management system. WfMSs have been developed with a strong focus on the machine-oriented view leading to a difficult acceptance by organizations [2].

But the problem goes beyond this human versus machine conflict. Nowadays one major challenge faced by organizations concerns resilience [3]: the capability to resist major business disruptions due to unforeseeable, unexpected or catastrophic reasons, leading the systems beyond the limits without serious human, physical or financial losses. Many organizations, e.g. aviation and banking, are adopting mechanisms to preserve themselves when facing major business disruptions. These mechanisms usually posit [3]: (1) flexibility understanding and acting upon the evolving context; (2) reliance on the experience of most knowledgeable people; and (3) capability to make decisions lacking full insights about the situation.

The development of resilient WfMSs emphasizes the human oriented perspective, since human judgement is fundamental to make decisions under unpredictable contexts. From our point of view, the major technical problems concern the integration between the human and machine views:

- Providing process guidance in contexts where work models fail to comply with the reality;
- Supporting dynamically evolving processes under emergency situations, though facilitating the transition to normal operations;
- Moving control from the technology to the operators whenever necessary;
- Supporting unpredictable actors' roles and work contexts.

This paper proposes an architectural approach aiming to reconcile the human and machine views and to increase organizational resilience. The paper also discusses in detail the components and mechanisms necessary to incorporate resilience properties into WfMSs. The paper is organized as follows. In the next section we elaborate the conceptual foundations of this research. Section 3 discusses in detail the architectural approach. Section 4 highlights some implementation details. And section 5 discusses the main implications of the proposed approach and draws some conclusions.

2. Conceptual Foundations

A *business process* is defined in [4] as a collection of activities tied together by a common goal and precedence relations; and *workflow management* as the automated coordination of these activities to carry out the business process. According to the Workflow Management Coalition (WfMC) [5], the WfMS infrastructure is composed by an enactment service, which manages workflow instances; client components, which control the user interactions; and other components necessary to store and interpret work models, to interoperate with other workflow services, and to monitor the system behaviour. The notion of workflow management is *machine-oriented*: it assumes the computer control of the business process based on the normative engagement of a work model [6].

But we should also analyze business processes according to the contextualization necessary to carry out the intended goals under different task conditions [7]. According to this *human-oriented* perspective, work models guide actors in a space of available actions but do not assume a normative engagement. The control is in the hands of the humans and not the technology.

And finally, we have to consider business processes as one bit of a larger organizational management strategy that may be best understood using the notion of *exception* [8]. Exceptions occur whenever the available repeatable processes fail to respond to a dynamic business context, humans fail to act upon the situations, or the technology constrains the operators conducting the necessary actions to overcome the situations.

The organizations tend to overcome exceptions using three strategies related with the exceptions' frequency and degree of exceptionality:

- Low frequency / Low exceptionality – Redesigning procedures and rules.
- Low frequency / High exceptionality – Mixed strategy relying on training and procedures.
- High frequency / Low exceptionality – Mixed strategy relying on training and procedures.
- High frequency / High exceptionality – Responding with discretionary actions based on training and experience.

Organizations thus operate with a continuum of *structured*, *mixed* and *unstructured* processes [4]. Several researchers studied the impact of exceptions on WfMSs. The major problem is that most WfMSs adopt the machine-oriented view over business processes and therefore are adapted to the Low frequency / Low exceptionality strategy. This means that WfMSs overcome exceptions if the handling strategy is restricted to the redesign of work models under strict control from the enactment service. This also means that organizations must look elsewhere to overcome situations falling outside this strategy.

We typify the Low frequency / Low exceptionality exceptions in [9]: (1) *basic failures*, related to failures in the underlying technology; (2) *application failures*, related with task execution; and (3) *expected exceptions*, events that may be predicted during the modelling phase but that do not correspond to the "normal" behaviour. WfMSs may automatically handle basic failures. Several techniques exist to recover from application failures; most of them are based on advanced database transaction techniques that guarantee data integrity and recovery. However, [9] points out that not every possible exception may be resolved this way and suggest escalating such failures into expected exceptions.

Researchers also developed mechanisms to automatically handle expected exceptions. Some rely on triggers to initiate predefined handling procedures [10, 11]. Others apply Artificial Intelligence techniques to identify prior handling procedures that could be automatically applied to similar situations. Naturally, it has also been suggested the expected exceptions that cannot be handled with these techniques should be transformed into unexpected exceptions.

The exceptions related with Low frequency / High exceptionality and High frequency / Low exceptionality require human involvement. These exceptions may be categorized as *unexpected exceptions* [9]. The unexpected exceptions result from incomplete or erroneous designs, changes in the business manoeuvre, and issues too complex to be handled by the work models [12]. Handling unexpected exception requires a mixed control strategy, combining the human control necessary to understand the situation and make decisions, with the control from the workflow enactment service necessary to preserve the work models correctness.

Several techniques have been developed to support human interventions in business processes at runtime [13, 14]. For instance, many WfMSs allow the operators to insert activities not specified in the work model but permitted by the enactment service. When more extensive changes are necessary, several WfMSs allow replacing whole work models in runtime. These techniques lead to dynamic and adaptive WfMS and represent one important approach to increase WfMS resilience. On the occurrence of unexpected exceptions, the operators may migrate running workflows to new work models without stopping or breaking the system [15].

Two types of interventions are considered [13]: *evolutionary* changes and *ad-hoc structured* changes. Evolutionary changes enact a new work model and thus have a broad and permanent impact in the WfMS. The ad-hoc structured changes are typically applied to a small set of workflow instances and thus have a transient impact. But these types of changes must be executed under the system control to preserve model correctness (avoid deadlocks, unreachable states and other problems). These techniques apply a set of automated checks to avoid violating the work model correctness.

The High frequency / High exceptionality situations emphasize the limitations mentioned above. The constraints imposed by correctness checks are beneficial to the WfMS operation but limit the operators. High frequency / High exceptionality situations usually involve human judgement under time pressure and incomplete information. In these situations *ad-hoc unstructured* changes to the work models may be necessary to overcome the situation, even at the cost of leading the WfMS to an incorrect state. But few techniques have been developed to integrate ad-hoc unstructured changes in WfMSs. [16] integrated a WfMS with collaborative tools, allowing to pass control from the WfMS to collaborative tools when an unexpected exception occurs. But no support is considered to continue with the normal WfMS operation after the exception is handled. Another contribution was proposed by [17], who developed a mechanism to determine which type of control is more suitable to the type of exception that has occurred and invoke adequate tools, including tools supporting ad-hoc unstructured changes.

2.1. Exception handling strategies

From the above discussion we realise that automated exception handling techniques are crucial to increase resilience. However, when unexpected exceptions occur, humans become a necessary component supporting resilience. Regarding the human role, the evolutionary techniques increase resilience by allowing migrating workflow instances towards new work models. The ad-hoc structured techniques further increase resilience by allowing more immediacy and less planning in the actions taken by the operators. Finally, the ad-hoc unstructured activities provide an increased level of resilience by removing the model correctness constraints and giving wider latitude of action to the operators.

In Figure 1 we classify the existing exception handling techniques according to two criteria: control and planning. The first criterion classifies the type of control. Three categories are defined: mechanistic, restricted humanistic and unrestricted humanistic. The second criterion concerns the immediacy of response to exceptions. Two categories are identified: planned and ad-hoc.

The systems that automatically handle (basic and application) failures are classified as mechanistic, since control is totally exerted by the technology. Furthermore, they require the reactions be planned in the design stage. Considering these constraints, they offer low resilience levels. The expected exception handling techniques offer more flexible reaction mechanisms, still these reactions have to be planned and control is totally exerted by the technology.

The evolutionary techniques allow the operators to dynamically change the planned organizational behaviour. The interventions are restricted by the model correctness criteria, but there is a significant increase in human control. Considering that the evolutionary techniques usually have broad impact on the business process, such reactions must be carefully planned, although fewer restrictions are imposed when compared with the previous techniques. The ad-hoc structured techniques, having less impact on the business process, are also less demanding in terms of planning and offer increased human control and flexibility to react to unforeseen events. Nevertheless, human intervention is to some extent limited by the technology, the reason why we classify these techniques in the restricted humanistic category.

We finally find the ad-hoc unstructured techniques. Here, resilience is at its maximum degree since interventions may be fully ad-hoc: non-planned and unrestricted by the technology.

Summarizing the whole scenario, we observe that, to increase resilience, organizations must integrate exception-handling techniques covering the path leading towards unrestricted humanistic and ad-hoc reactions. However, as we have seen, few techniques currently support ad-hoc unstructured reactions in WfMSs. Considering this scenario, our research goals are centred on the integrated support to exception handling, covering not only mechanistic and restricted humanistic control but also unrestricted humanistic control. In the next section we describe in detail our approach.

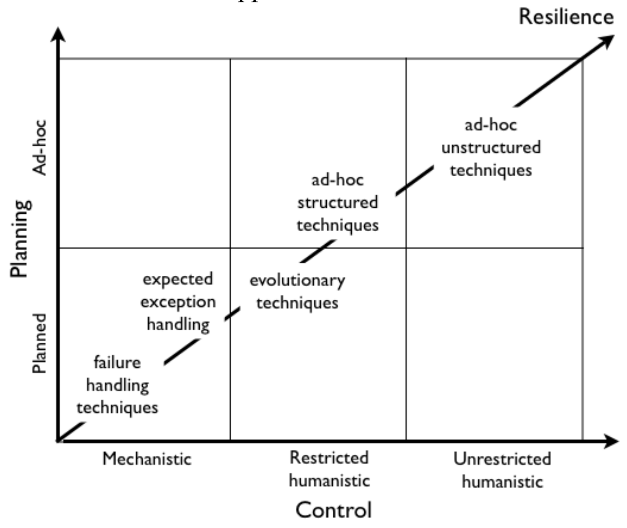

Figure 1. Classification of exception handling techniques according to resilience criteria

3. Architectural Approach

The WfMC reference model [18] has been widely adopted in industry. It fails however to integrate the level of resilience described in the previous section, since it does not foresee to move away from the restricted towards the unrestricted humanistic control.

To support unrestricted humanistic control, we extended the reference model in the way represented in Figure 2. The change concerns one new component, designated *control switch*, responsible for moving control between the workflow engine and the operators whenever an unexpected event is detected. This capability requires direct interaction with the workflow enactment service and the process definition tools to support runtime changes in workflow instances. In some situations it may also be necessary to monitor the evolution of the enactment service using the available administration and monitoring tools. In the following we will further detail the control switch.

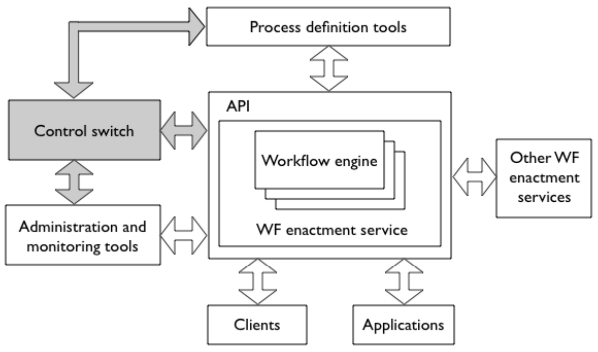

Fig. 2: Extended reference model

The proposed architectural change is based on the following general behaviour:

1. Under normal conditions, the workflow instances are managed by the enactment service;

2. The occurrence of basic and application failures is handled by the enactment service, using transaction-processing techniques. Whenever these techniques fail, the exception is propagated as an expected exception;
3. The occurrence of an expected exception triggers the corresponding handling procedures, managed by the enactment service. These procedures are predefined at design time. If a trigger is not available or is incapable to resolve the situation, the exception is propagated as unexpected exception;
4. The occurrence of an unexpected exception invokes the control switch;
5. The control switch will determine which workflow instances will continue to be managed by the enactment service and which instances the operators will manage themselves;
6. The control switch will continue its operation until the operators are able to bring back all workflow instances to a correct state (which requires performing correctness checks).

Regarding the step 5 mentioned above, the control switch must also determine what type of intervention is necessary: (1) redesign the work models with support from the process definition tools (evolutionary changes) and constrained by the workflow engine; (2) apply changes to a set of workflow instances (ad-hoc structured changes) constrained by the workflow engine; or (3) apply ad-hoc unstructured changes independently from any constraints. This behaviour is controlled by a special workflow managed by the enactment service. I.e., the WfMS manages the organization's business processes plus an *exception handling workflow* responsible for coordinating the exception handling. More concrete details about this workflow will be given later.

The control switch requires several components to execute the following services (see Figure 3): (1) detect the occurrence of exceptions, which may be raised by the WfMS or by an operator; (2) support recovery actions, including quasi-atomic actions (cancel, jump, etc.) and dynamic model changes, all of them performed by the enactment service; (2) maintain a log of all the relevant events occurring in the control switch; and (3) interface with the users.

3.1. User interface component

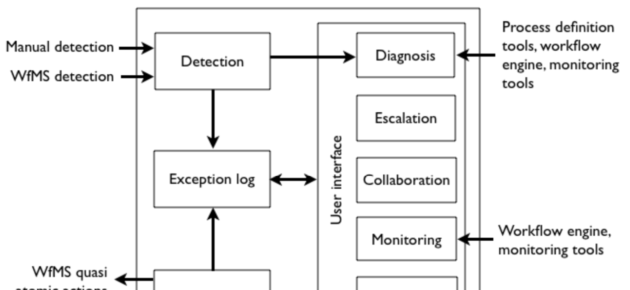
Manual detection
WfMS detection
Detection
Diagnosis
Process definition tools, workflow engine, monitoring tools
Escalation
Exception log
Collaboration
User interface
Monitoring
Workflow engine, monitoring tools
WfMS quasi atomic actions
WfMS dynamic model changes
Recovery
Recovery

Fig. 3: Control switch components

The user interface component is specified in more detail in Figure 3, including: (1) diagnosis component, collecting relevant data about the situation; (2) escalation component, allowing to involve other users in the process; (3) collaboration component, supporting collaboration and decision making; (4) monitoring component, allowing users to follow events occurring in the system; and (5) recovery component, supporting the users invoking recovery actions.

The diagnosis component collects relevant data from the users and the process definition tools, workflow engine and monitoring tools [19]:

- Detection – *automatic* or *manual*;
- Event type – the type of event that generated the exception (e.g. temporal event);
- Scope – *instance* when only a set of workflow instances was affected; or *model* when the business model was affected;
- Affected instances – list of affected instances;
- Affected tasks – list of affected tasks;
- Affected models – list of affected work models;
- Exception description – short description of what occurred, written by the users;
- Type of intervention – *evolutionary, ad-hoc structured* or *ad-hoc unstructured*;
- Intervention details – short description of what should be done to resolve the exception;
- Organizational impact – *employee*, *peer*, *supervisor* or *group*;

The users concurrently do the diagnosis of the situation. The escalation component serves to bring more people to the exception handling process. The escalation considers four decision levels: (1) the operator; (2) the peer-level allows multiple peers to concurrently handle the exception; (3) the supervisor; and (4) the group, where multiple actors collaborate in the process.

The collaboration component aims to increase the situation awareness by using adequate collaboration tools. The monitoring component serves to specify and instantiate ad-hoc tasks aiming to collect specific information about the system and deliver it to the users, thus allowing them to follow the system evolution. Finally, the recovery component provides a user-interface to the services available in the control switch to operate the enactment service.

4. Implementation

The proposed architecture was implemented in OpenSymphony [20]. The details regarding this implementation have been divided in two sections, one related with the architecture implementation and the other with the exception handling workflow.

4.1. Architecture implementation

In Figure 4 we overview the integration between the control switch and the exception handling service. The following core services are available: (1) concurrent management of exception detection triggers; (2)

instantiation of the exception handling workflow; (3) notification of users; (4) escalating the exception to other users; (5) integration of tools supporting collaborative diagnosis, awareness and decision making; (6) invocation of monitoring actions to obtain feedback about the system behaviour; (7) concurrent users' recovery interventions; (8) and recovery actions.

The exception handling workflow was implemented as an OpenSymphony workflow. The adopted external collaboration tools were e-mail and chat tools, which may be selected according to the users' preferences.

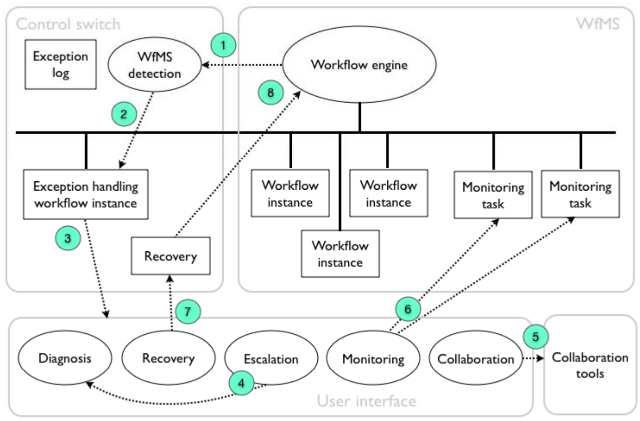

Fig. 4: Detailed architecture and core services

4.2. Exception handling workflow

When an exception is detected, the OpenSymphony instantiates the exception handling workflow. By default, the exception is always associated to a specific workflow instance and task, for which there is always a user known by the system. This user is therefore the one that is contacted with a request to characterize the exception and apply any necessary recovery actions. The diagnosis task demands the user to classify the exception according to several categories (type of intervention, description, etc.). Some details about the exception are automatically given by the system whenever possible, e.g. affected instances and tasks). Immediately after this task, the exception handling workflow activates the following parallel tasks:

- Diagnosis, which may be continuously and concurrently updated by several users;
- Escalating the exception to other users, which results in additional running instances of the exception handling workflow requiring assistance from the selected users;
- Collaborating, allowing the users affected by the same exception to use the chat or mail tools;
- Recovery, allowing users to invoke quasi-atomic recovery actions or applying dynamic changes to the workflow engine;
- Monitoring; allowing to instantiate ad-hoc data acquisition and notification tasks in the workflow engine.

When any user considers that the exceptional situation may have been resolved, he may execute an additional task, which suspends the control switch operation and verifies model correctness. If any inconsistency is detected, the exception handling workflow will continue operation and the users will be kept involved in the process. When no inconsistency is detected the exception handling workflow is terminated.

More details about the exception handling workflow and its implementation in OpenSymphony can be found in [14, 21]. What we would like to emphasise here is that the exception handling workflow is implemented at the same system level and with the same tools as any other workflow. The same comment applies to most of the core services. For instance, the diagnosis task is implemented with a set of queries to the workflow database. The only exception regards the recovery service, as it has to interface at low-level with the workflow engine to initiate tasks, suspend tasks, etc.

5. Discussion and Conclusions

The [8] perspective over the organizational strategies necessary to overcome exceptions is the fundamental key to understand the limitations and possibilities of WfMS. These systems have been mostly developed under the Low frequency / Low exceptionality assumption. Several developments, coined as flexible workflow, may extend the WfMS functionality to the Low frequency / High exceptionality and High frequency / Low exceptionality strategies. The High frequency / High exceptionality strategy has unfortunately been under developed. To increase organizational resilience, the developers must figure out strategies capable to maintain WfMS functionality under the High frequency / High exceptionality events.

This paper shows that High frequency / High exceptionality events should be tackled with humanistic unrestricted control and had-hoc planning. Under these circumstances, the operators should be able to apply immediate actions while avoiding any control imposed by the WfMS to preserve model correctness.

Of course we find here a trade-off between flexibility and responsiveness on the one hand; and model correctness on the other hand. In this paper we propose an architectural scheme to manage the system evolution and trajectory during exception handling that affords dealing with the trade-off in runtime.

Our architectural solution defines a control switch aiming to move control between the workflow engine and the system operators. This switch is fundamental to support unrestricted humanistic interventions whenever the operators perceive them as necessary. Independently from the specific behaviour of the control switch, the developers should always consider extending WfMSs with this important component.

In this paper we also propose a detailed functionality for the control switch. This includes the combined operation of an exception handling workflow with several components responsible for interfacing with the workflow engine and the operators. Regarding the interface with the operators, the proposed solution involves five components responsible, respectively, for the exception diagnosis, escalation and monitoring; and the support to collaboration and recovery actions.

The developers should regard the escalation and collaboration components as most important. The escalation component is responsible for involving additional operators in the exception handling process, thus addressing a fundamental resilience principle: involving the most knowledgeable persons to overcome the exceptional situations. The escalation component relies on the WfMS itself to involve and orchestrate these persons. The collaboration component addresses another important resilience principle: empowering perception, awareness and decision-making abilities through participation and collaboration. The developers may consider implementing this component with several tool already common in organizations, such as email and chat tools.

In summary, the proposed architecture integrates the WfMS infrastructure with other organizational systems and tools, with the fundamental goal to orchestrate the human interventions necessary to overcome the exceptional situations. Of course many issues remain to be addressed by the developers. In particular, they should carefully consider that releasing control from the operators back to the workflow engine may take some time and, during that period, the operators may apply wrong actions that may put the system at risk. Therefore, additional risk management procedures may have to be considered, including training, undo-redo functionality and the development of awareness mechanisms.

Acknowledgements

This paper was partially supported by: the Portuguese Foundation for Science and Technology, Project FCT (PTDC/EIA/67589/2006).

References

[1] L. Suchman, "Office procedure as practical action: models of work and system design," *ACM Transactions on Office Information Systems,* vol. 1, pp. 320-328, 1983.

[2] 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.

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

[4] 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.

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

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

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

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

[9] J. Eder and W. Liebhart, "The Workflow Activity Model WAMO," in *International Conference on Cooperative Information Systems (CoopIS '95)*, Austria, 1995.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] D. Hollingsworth, "Workflow Management Coalition: The Workflow Reference Model," Workflow Management Coalition, Hampshire, UK 1995.

[19] 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.

[20] OpenSymphony, "The OpenSymphony Project."

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